

Journal of
**Language
Modelling**

VOLUME 3 ISSUE 1
JUNE 2015



*Institute of Computer Science
Polish Academy of Sciences
Warsaw*

Journal of Language Modelling

VOLUME 3 ISSUE 1
JUNE 2015

Editorials

A word from the editors 1

*Adam Przepiórkowski, Elżbieta Hajnicz,
Agnieszka Mykowiecka, Marcin Woliński*

High-level methodologies for grammar engineering.

Introduction to the special issue 5

Denys Duchier, Yannick Parmentier

Tools and resources

The CoreGram project: theoretical linguistics,
theory development, and verification 21

Stefan Müller

Articles

A logical approach to grammar description 87

Lionel Clément, Jérôme Kirman, Sylvain Salvati

A syntactic component for Vietnamese language processing 145

Phuong Le-Hong, Azim Roussanaly, Thi Minh Huyen Nguyen

Implementing semantic frames as typed feature structures with XMG 185

Timm Lichte, Simon Petitjean

A type-logical treebank for French 229

Richard Moot

FRIGRAM: a French Interaction Grammar 265

Guy Perrier, Bruno Guillaume

JOURNAL OF
LANGUAGE MODELLING

ISSN 2299-8470 (electronic version)

ISSN 2299-856X (printed version)

<http://jlm.ipipan.waw.pl/>

MANAGING EDITOR

Adam Przepiórkowski IPI PAN

SECTION EDITORS

Elżbieta Hajnicz IPI PAN

Agnieszka Mykowiecka IPI PAN

Marcin Woliński IPI PAN

STATISTICS EDITOR

Łukasz Dębowski IPI PAN



Published by IPI PAN

Instytut Podstaw Informatyki

Polskiej Akademii Nauk

Institute of Computer Science

Polish Academy of Sciences

ul. Jana Kazimierza 5, 01-248 Warszawa, Poland

Circulation: 100 + print on demand

Layout designed by Adam Twardoch.

Typeset in X_YL^AT_EX using the typefaces: *Playfair Display*
by Claus Eggers Sørensen, *Charis SIL* by SIL International,

JLM monogram by Łukasz Dziedzic.

*All content is licensed under
the Creative Commons Attribution 3.0 Unported License.*
<http://creativecommons.org/licenses/by/3.0/>



EDITORIAL BOARD

Steven Abney University of Michigan, USA

Ash Asudeh Carleton University, CANADA;
University of Oxford, UNITED KINGDOM

Chris Biemann Technische Universität Darmstadt, GERMANY

Igor Boguslavsky Technical University of Madrid, SPAIN;
Institute for Information Transmission Problems,
Russian Academy of Sciences, Moscow, RUSSIA

António Branco University of Lisbon, PORTUGAL

David Chiang University of Southern California, Los Angeles, USA

Greville Corbett University of Surrey, UNITED KINGDOM

Dan Cristea University of Iași, ROMANIA

Jan Daciuk Gdańsk University of Technology, POLAND

Mary Dalrymple University of Oxford, UNITED KINGDOM

Darja Fišer University of Ljubljana, SLOVENIA

Anette Frank Universität Heidelberg, GERMANY

Claire Gardent CNRS/LORIA, Nancy, FRANCE

Jonathan Ginzburg Université Paris-Diderot, FRANCE

Stefan Th. Gries University of California, Santa Barbara, USA

Heiki-Jaan Kaalep University of Tartu, ESTONIA

Laura Kallmeyer Heinrich-Heine-Universität Düsseldorf, GERMANY

Jong-Bok Kim Kyung Hee University, Seoul, KOREA

Kimmo Koskenniemi University of Helsinki, FINLAND

Jonas Kuhn Universität Stuttgart, GERMANY

Alessandro Lenci University of Pisa, ITALY

Ján Mačutek Comenius University in Bratislava, SLOVAKIA

Igor Mel'čuk University of Montreal, CANADA

Glyn Morrill Technical University of Catalonia, Barcelona, SPAIN

Stefan Müller Freie Universität Berlin, GERMANY
Reinhard Muskens Tilburg University, NETHERLANDS
Mark-Jan Nederhof University of St Andrews, UNITED KINGDOM
Petya Osenova Sofia University, BULGARIA
David Pesetsky Massachusetts Institute of Technology, USA
Maciej Piasecki Wrocław University of Technology, POLAND
Christopher Potts Stanford University, USA
Louisa Sadler University of Essex, UNITED KINGDOM
Ivan A. Sag † Stanford University, USA
Agata Savary Université François Rabelais Tours, FRANCE
Sabine Schulte im Walde Universität Stuttgart, GERMANY
Stuart M. Shieber Harvard University, USA
Mark Steedman University of Edinburgh, UNITED KINGDOM
Stan Szpakowicz School of Electrical Engineering
and Computer Science, University of Ottawa, CANADA
Shravan Vasishth Universität Potsdam, GERMANY
Zygmunt Vetulani Adam Mickiewicz University, Poznań, POLAND
Aline Villavicencio Federal University of Rio Grande do Sul,
Porto Alegre, BRAZIL
Veronika Vincze University of Szeged, HUNGARY
Yorick Wilks Florida Institute of Human and Machine Cognition, USA
Shuly Wintner University of Haifa, ISRAEL
Zdeněk Žabokrtský Charles University in Prague, CZECH REPUBLIC

A word from the editors

*Adam Przepiórkowski, Elżbieta Hajnicz,
Agnieszka Mykowiecka, and Marcin Woliński*
Institute of Computer Science,
Polish Academy of Sciences,
Warsaw, Poland

Welcome to the sixth issue of the Journal of Language Modelling, which is the first special issue of JLM. We, the Editors, are very happy that this first special issue is devoted to grammar engineering, which is right in the centre of the scope of JLM, as stated on its WWW page:

Journal of Language Modelling is a free (for readers and authors alike) open-access peer-reviewed journal aiming to bridge the gap between theoretical linguistics and natural language processing. Although typical articles are concerned with linguistic generalisations – either with their application in natural language processing, or with their discovery in language corpora – possible topics range from linguistic analyses which are sufficiently precise to be implementable to mathematical models of aspects of language, and further to computational systems making non-trivial use of linguistic insights.

<http://jlm.ipipan.waw.pl/>

The scope and content of this issue are described in the introduction by the Guest Editors: Denys Duchier and Yannick Parmentier. While the whole issue is devoted to high-level methodologies for grammar engineering, we envisage future “special issues” as possibly taking only a part of an issue which may also contain regular papers.

So far – apart from *Editorials* like this one and *Acknowledgements* to reviewers – only regular peer-reviewed *Articles* have been published in JLM. This issue introduces a new section type, *Tools and Resources* (see the contribution by Stefan Müller). Typical papers appearing in this section will contain descriptions of linguistic tools or resources, normally authored by their developers. Just as regular articles, such

papers will be peer-reviewed, but referees will be asked to evaluate the linguistic sophistication of the described tool or resource and its envisaged impact in theoretical, computational or mathematical linguistics, rather than just the theoretical novelty and validity. We invite submissions describing novel formal descriptions of morphological systems, linguistically rich syntactic and semantic lexica, corpora annotated with new linguistic data types, “electronic descriptions of natural language” (to use the current Guest Editors' preferred term for implemented grammars), parsers producing syntactic, semantic and pragmatic representations of utterances, etc. – this list is certainly not meant to be exhaustive.

Apart from *Tools and Resources*, three other types of sections are planned to be added in future issues, all containing peer-reviewed papers, but with different reviewing criteria. The *Overviews* section will consist of articles synthetically describing a given field or area, with emphasis on recent developments. The *Squibs and Discussion* section will contain smaller contributions making a point within the space of a few pages or briefly responding to a paper previously published in JLM. Finally, *Reviews* will contain evaluations of recently published monographs.

Let us finish this editorial on a historical note and with a plea. In the inaugural issue 1(0) we wrote:

Launching a new journal is a high-risk business. Many people have already invested much of their time in this initiative, and returns are far from certain – the community may or may not accept it. We, the Managing Editors, are cautiously optimistic...

Now, five issues later, our optimism is less cautious: JLM has managed to attract a reasonable number of good submissions, including many from already well-established researchers, and – apart from such regular submissions – there are specific plans for two more special issues with carefully reviewed and selected papers. We also receive a lot of positive feedback from, both, authors and readers. It seems that JLM has indeed answered a need of the formal and computational linguistics community.

Despite offers from major scientific publishers, JLM remains independent, community-driven and free for all – readers and authors

A word from the editors

alike. We, the Editors, invest our time in this enterprise and we rely on similarly voluntary help from the reviewers and other people involved in the production of the journal. Of these “other people”, Copy Editors are undoubtedly most important and most skilled: a good JLM Copy Editor should not only be a native speaker of English (or perhaps a near-native graduate of English Philology) with a good command of the scientific style, but should also know some basics of L^AT_EX or X_YL^AT_EX and – preferably – have interest in theoretical or computational linguistics. So far, JLM Copy Editors have been doing an excellent job; to cite one of the authors of the current issue, the author of many papers published by more established scientific publishers: “I never got such a good copy editing in my whole scientific life...”. So we would like to heartily thank our current and past Copy Editors for their devotion to the good cause: Jette Viethen, Dave Carter, Filip Skwarski, Agnieszka Patejuk, Natalia Kocyba, Alina Wróblewska and Chris M. Fournier.

However, given the increased number of JLM submissions, and in order to keep the workload of each Copy Editor reasonably small, we badly need the help of a few more Copy Editors. Hence, the plea: if you are a PhD student in Linguistics, Natural Language Processing, or a related field, and you satisfy the above description, please help us for a year or two (or longer!). Similarly, if you are a professor with PhD students, please consider encouraging them to help us – it is in the best interest of the community to keep JLM independent and free for all.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



High-level methodologies for grammar engineering. Introduction to the special issue

Denys Duchier and Yannick Parmentier
Laboratoire d'Informatique Fondamentale d'Orléans,
Université d'Orléans, France

ABSTRACT

Grammar engineering is the task of designing and implementing linguistically motivated electronic descriptions of natural language (so-called grammars). These grammars are expressed within well-defined theoretical frameworks, and offer a fine-grained description of natural language. While grammars were first used to describe syntax, that is to say, the relations between constituents in a sentence, they often go beyond syntax and include semantic information. Grammar engineering provides precise descriptions which can be used for natural language understanding and generation, making these valuable resources for various natural language applications, including textual entailment, dialogue systems, or machine translation. The first attempts at designing large-scale resource grammars were costly because of the complexity of the task (Erbach 1990) and of the number of persons that were needed (see e.g. Doran *et al.* 1997). Advances in the field have led to the development of environments for semi-automatic grammar engineering, borrowing ideas from compilation (grammar engineering is compared with software development) and machine learning. This special issue reports on new trends in the field, where grammar engineering benefits from elaborate high-level methodologies and techniques, dealing with various issues (both theoretical and practical).

Keywords:
grammar
engineering,
formal language,
syntax,
semantics

Grammar engineering, the task of designing and implementing linguistically motivated electronic grammars, has been an active field for decades, following seminal work by Chomsky (1957) on formal languages. The kind of deep structures produced in this field contain rich information, which makes them a valuable resource for various NLP applications, including natural language parsing / generation, textual entailment, or dialogue systems. Among the reasons which make grammar engineering a complex task, one may cite the variety of theoretical frameworks that are used to represent linguistic information, and the intrinsic complexity coming from interactions between rules within large grammars.

In this context, the classical model of hand-crafted grammar has been replaced with more advanced techniques, which we may call semi-automatic grammar production. These techniques vary depending on the target formalism, the target language, or the target linguistic dimensions (e.g. syntax, semantics, morphology, etc.). In the following, we first report on the production of formal grammars (Section 1.1). We then report on the main resource grammars that are available (Section 1.2). We then give a brief overview of the current issues in grammar engineering (Section 1.3). Finally, in Section 2, we summarise the contributions of this special issue, and in Section 3, we conclude about the current status of grammar engineering.

1.1 *Semi-automatic production of formal grammars*

Over the last decades, several approaches to formally describing natural language syntax have been proposed, starting with relatively basic string rewriting systems such as Context-Free Grammar (Chomsky 1956), to continue with more elaborate constraint-based systems such as Head-driven Phrase Structure Grammar (Sag and Pollard 1987). These formal grammars differ in terms of expressive power and computational complexity. While theoretical research on formal grammar addresses the question of what expressive power is needed to describe natural language syntax (and of what computational cost it implies),¹ more practical research is concerned with issues arising from building large grammars.

¹ See e.g. Joshi 1985.

Large grammars often exhibit a high structural redundancy, especially when using *lexicalised* formalisms where each grammar rule is associated with at least one lexical item.² This high redundancy heavily affects grammar production and maintenance. Indeed, some representation choice applies to many grammatical structures. Should it be modified, a costly revision of the grammar would be required.

In order to facilitate grammar engineering, two main approaches have been considered. The first approach, which we may call knowledge-driven grammar engineering, aims at formally describing the structures belonging to a grammar (which in turn describes natural language syntax). Such a formal description is defined by linguists using a description language.³ Examples of such description languages include the PATR II (Shieber 1984), the DATR (Evans and Gazdar 1996), and more recently the XMG (Crabbé *et al.* 2013) languages.

Such languages offer a well-defined syntax and semantics to express the relations between grammar structures. These relations are then automatically processed to build a set of structures (i.e., the target grammar). While working with description languages, grammar engineering becomes similar to software engineering. Indeed, both rely on developers working together on a source code, which can be processed to produce some information (e.g. some binary code in one case, or some syntactic structures in the other case).

The second approach to grammar engineering, which we may call data-driven, aims at acquiring the structures belonging to a grammar from annotated corpora (so-called treebanks) (Abeillé 2003). The complexity of grammar engineering is moved from designing grammar rules to designing learning algorithms. Examples of such grammar induction include for instance work by Charniak (1994, Chapter 7), Villavicencio (2002), or Cahill *et al.* (2005). As is the case with statistical approaches in general, grammar learning suffers from the sensitivity to the corpus used to infer the grammar, not mentioning the fact that it requires large annotated corpora which may be lacking when

²Such lexicalised formalisms are particularly interesting for the lexicon and can be seen as a mapping between a word and its various uses in a sentence, and parsing complexity is reduced since only the grammar rules associated with the input words need to be considered.

³In this respect, one may consider that the grammar description itself is a linguistically motivated description of natural language.

working on under-resourced languages. Still, data-driven approaches to grammar learning showed promising results for English, in particular in terms of coverage (Cahill *et al.* 2008).

These two approaches can also be seen as complementary. While knowledge-based methods make it possible to design precision grammars where one can integrate various extra information (e.g. semantic structures), they often hardly scale up so far as describing unrestricted text. On the other hand, while data-driven methods allow to build robust grammars, which can achieve very good results in terms of coverage, these automatically acquired grammars sometimes fail at describing linguistic phenomena which are very infrequent.

1.2 *Resource grammars*

Among the many formalisms which have been proposed to describe natural language syntax, some have been used in practice to develop core or large grammars for a wide range of languages. Formalisms for which there exist available electronic grammars include (by chronological order of publication): Tree Adjoining Grammar (TAG, Joshi *et al.* 1975), Lexical Functional Grammar (LFG, Kaplan and Bresnan 1982), Head-driven Phrase Structure Grammar (HPSG, Sag and Polard 1987), Combinatory Categorical Grammar (CCG, Steedman 1987), Interaction Grammar (IG, Perrier 2000), or Property Grammar (PG, Blache 2005).

Many efficient description-language-based integrated grammar development environments have been created for these formalisms, such as XLE (Butt *et al.* 1999) for LFG, ALE (Carpenter and Penn 1999), TRALE (Meurers *et al.* 2002) and LKB (Copestake 2002) for HPSG, or DotCCG (Baldrige *et al.* 2007) and GF (Ranta 2011) for CCG. Such environments made it possible to develop large grammars for several languages, see Table 1. Such grammars have been used in practical applications such as machine translation (Lønning and Oepen 2006), textual adventure games (Benotti 2009), or second language learning (Perez-Beltrachini *et al.* 2012).

At the same time, efficient learning algorithms have been developed to induce large grammars from annotated corpora for some of these formalisms, see e.g. Xia (1999) for TAG, Cahill *et al.* (2002) for LFG, Miyao *et al.* (2005) for HPSG, or Hockenmaier and Steedman (2002) for CCG. These automatically learned grammars have been

High-level methodologies for grammar engineering

Table 1: Available electronic grammars (non-exhaustive)

Type	Grammar	Reference
TAG	XTAG (English)	http://www.cis.upenn.edu/~xtag/
	XTAG using XMG (English)	http://homepages.inf.ed.ac.uk/s0896251/XMG-basedXTAG/titlepage.html
	FrenchTAG	https://sourcesup.renater.fr/scm/viewvc.php/trunk/METAGRAMMARS/FrenchTAG/?root=xmg
	GerTT (German)	http://www.sfs.uni-tuebingen.de/emmy/res-en.html
LFG	Parallel Grammar (Norwegian, Japanese, etc.)	http://pargram.b.uib.no/
	HunGram (Hungarian)	http://hungram.unideb.hu
	Urdu ParGram	http://ling.uni-konstanz.de/pages/home/pargram_urdu
	POLFIE (Polish)	http://zil.ipipan.waw.pl/LFG
HPSG	English Resource Grammar	http://lingo.stanford.edu/erg.html
	GG (German Grammar)	http://gg.dfki.de/
	JaCY Grammar (Japanese)	http://jacy.opendfki.de/
	Korean Resource Grammar	http://web.khu.ac.kr/~jongbok/projects/krgram.html
	Modern Greek Resource Grammar	http://www.delph-in.net/mgrg/
	NorSourceGrammar (Norwegian)	http://wiki.delph-in.net/moin/NorsourceTop
	Spanish Resource Grammar	http://svn.emmtee.net/trunk/upf/srg/
	Berligram (German), Danish, Chinese, Persian	https://hpsg.fu-berlin.de/Software/
CCG	openCCG (English)	http://www.utcompling.com/wiki/openccg/openccg-grammar-writing
	Grail (French, Dutch)	http://www.labri.fr/perso/moot/Corpus/
	GF (29 languages)	http://www.grammaticalframework.org/lib/doc/synopsis.html
IG	FriGram	http://wikilligramme.loria.fr/doku.php/frig:frig
PG	FrenchPG	http://prost.jeanphilippe.free.fr/resources/grammaireGP13.JPP.these.xml

evaluated on test suites, and often used in practical applications such as semantic construction (Bos *et al.* 2004), dialogue systems (Foster *et al.* 2005), or machine translation (Birch *et al.* 2007).⁴

1.3 Current issues

As mentioned above, the field of grammar engineering has been active for several decades. It succeeded in providing the research community with both large resources for a wide range of languages, and techniques for efficient grammar production. By efficient, it is meant that knowledge-based approaches now offer expressive and modular description languages, together with tools for computer-aided grammar design.⁵ In the same spirit, data-driven approaches now offer generic algorithms and frameworks which can be applied to the induction of grammars for many formalisms and languages (provided there exist available treebanks for these languages).

Still, the field has a lot more to offer, on-going projects aim at removing existing barriers in grammar engineering, such as *the lack of enhanced grammar development techniques and tools*, which would facilitate grammar debugging, grammar evaluation, or collaborative grammar design. Attempts at providing such techniques include work by Gardent and Kruszewski (2012) on debugging and by Hoetmer (2005) and Sygal and Wintner (2011) on grammar design.

Another current issue in grammar engineering concerns *parsing efficiency*. Indeed parsing complexity depends not only on the length of the input sentence, but also on the grammar size. In order to parse sentences using large grammars, several options have been considered, including the on-line (symbolic or probabilistic) selection of a sub-part of the grammar (Zhang *et al.* 2009; Gardent *et al.* 2014),⁶ or parsing using factorised grammars (Carroll *et al.* 2000; Villemonte De La Clergerie 2010).

Other challenges include *multilingual and cross-framework grammar engineering*. While there exist several projects aiming at building

⁴ Some of these automatically acquired grammars are available on-line, see e.g. <http://web.engr.illinois.edu/~juliahr/CCGlexicon/index.html> for CCG, or <http://lfg-demo.computing.dcu.ie/lfgparser.html> for LFG.

⁵ See e.g. the Matrix for HPSG (Bender *et al.* 2010).

⁶ Following seminal work by Bangalore and Joshi (1999), this selection is often called supertagging.

parallel grammars (see e.g. Butt *et al.* 2002; Flickinger *et al.* 2012), cross-framework grammar engineering did not (yet) achieve the same results. One may cite seminal work by Clément and Kinyon (2003) on the description of parallel TAG-LFG from a common abstract description (called metagrammar), or more recent work by Crabbé *et al.* (2014) on the design of a constraint-based description language which could be applied to the description of grammars belonging to distinct formalisms. In the latter, the authors show how to enrich the description language to support several target formalisms, while in the former the authors show how to project a common description to several target formalisms (the metagrammar could be seen to some extent as a universal grammar). Without going as far as designing a universal grammar, *grammar reusability* (i.e., sharing information between grammars) remains an important challenge.

Another interesting topic concerns *grammar interfaces*. One of the motivations behind grammar engineering is the possibility to build rich semantic representations. The definition and implementation of a syntax / semantics interface within large grammars is an active field (see e.g. Gardent 2008; Kallmeyer and Osswald 2013), for both theoretical (definition / selection of an adequate semantic formalism) and technical (limited grammar readability and extensibility) reasons.

Describing under-resourced languages is also an active field within the grammar engineering community. The objective is twofold. Grammar engineering can help to (i) better understand e.g. minority languages (by implementing linguistic theories and checking how this implementation compares with field data), and also (ii) provide electronic grammars (which would make it possible to develop NLP applications for these languages, and/or build core treebanks, which could in turn be beneficial to grammar engineering). Recent knowledge-based attempts at creating linguistic resources for under-resourced languages include work by Bender (2008) and Duchier *et al.* (2012).

Last but not least, in order to improve *grammar coverage*, novel ideas are needed. As mentioned above, a first step towards a better grammar coverage was to automatically learn the grammar from annotated corpora. In order to get grammars with a better coverage while keeping a high precision, hybrid techniques involving both knowledge-based and data-driven methods are needed. Seminal work by Baldwin *et al.* (2005) expresses the same concerns.

In order to improve grammar coverage, one major issue needs to be addressed, namely, *Multi-Word Expressions*. Such expressions are often ignored when designing core grammars, while they frequently appear in unrestricted text. Work on MWE detection for enhancing parsing with HPSG has been done by Zhang *et al.* (2006), where authors use parsing error mining techniques to detect whether unknown words belong to some MWE which is in turn included in the lexicon.⁷ Further work in this field is needed to improve grammar precision and coverage.⁸

2

CONTRIBUTIONS TO THIS SPECIAL ISSUE

This special issue contains contributions dealing with several aspects of grammar engineering, namely description languages (Clément *et al.*), grammar extraction (Le-Hong *et al.*), syntax / semantics interface (Lichte and Petitjean), grammar coverage (Moot), multilingual grammars (Müller), and grammar development and maintenance (Perrier and Guillaume).

Clément, Kirman, and Salvati present a logic-based grammar description formalism. They use this formalism to describe both mildly context-sensitive grammars and their semantic interpretation. As an illustration, this formalism is applied to the (syntactic and semantic) description of several linguistic phenomena related to extraction in Dutch, English, and German.

Le-Hong, Roussanaly and Nguyen present the development of a linguistic resource for Vietnamese using the TAG formalism. The authors first show how to semi-automatically extract such a grammar from a treebank of Vietnamese. In a second step, they use this grammar for deep parsing. In particular, they present a complete pipeline for parsing Vietnamese sentences to produce constituent and dependency structures.

⁷ As is the case in lexicalised formalisms, the term *lexicon* is used here to refer to the grammar entries.

⁸ Better support of MWEs within lexicons, grammars, and applications is among the topics of the current PARSEME international initiative, EU COST Action IC1207, see <http://www.parseme.eu>.

Lichte and Petitjean present an extension of the XMG description language with a new linguistic dimension based on semantic frames. In their approach, the authors aim at offering a description language, which can be used to express various constraints on types. They apply this formalism to the definition of a syntax / semantics interface within an English TAG.

Moot reports on the development of a type-logical treebank for French, and its use for wide-coverage syntactic and semantic parsing. This article contains information about the various tasks involved in the development of a competitive type-logical parser for French using an automatically-extracted broad-coverage type-logical grammar.

Müller presents the CoreGram project, which aims at providing HPSG grammars for various typologically distinct languages. In this approach, a multilingual grammar is used to represent a common core shared by these languages. This article gives theoretical linguistic motivations behind multilingual grammars, along with theoretical grammar development concepts, and information about the concrete implementation of the corresponding HPSG grammars.

Perrier and Guillaume present FriGram, a broad-coverage French IG, which relies on a modular architecture and can be interfaced with various lexicons. This article also addresses grammar design and maintenance issues by presenting grammar-consistency principles which are implemented within FriGram. The authors also report on the current status of the grammar (coverage, comparison with other resource grammars for French, evaluation).

In this introduction, we gave an overview of past and recent advances in the field of grammar engineering. We presented the main approaches for semi-automatic grammar production, namely knowledge-based approaches, which rely on linguistically motivated descriptions of formal grammar designed by experts, and data-driven approaches, which rely on robust broad-coverage grammars extracted from large annotated corpora.

We also reported on existing available resource grammars for various languages and grammar formalisms, and summarised current issues in grammar engineering. These issues include the lack of tech-

niques and tools for easier grammar extension and maintenance (e.g. debugging facilities), a sometimes low parsing efficiency when dealing with large grammars, the limited coverage of hand-crafted grammars (especially regarding multi-word expressions), the difficulties to interface syntax with other linguistic dimensions, and a weak reusability between grammars belonging to different formalisms or describing different languages.

We finally gave a brief overview of the contributions to this special issue, which cover both knowledge-based and data-driven approaches, along with several grammar formalisms (namely CCG, HPSG, TAG, IG), several linguistic dimensions (syntax and semantics), and several languages (including English, Dutch, German, French, Danish, Persian, etc.).

ACKNOWLEDGEMENTS

As guest editors of this special issue, we would like to thank the members of the editorial board and editorial team of the *Journal of Language Modelling* for their work regarding the reviewing, copy-editing, and typesetting of the articles submitted to this issue. We are also grateful to the members of the guest editorial board. This issue would not have been possible without their valuable contribution to the reviewing process. Finally, we would like to express our gratitude to Adam Przepiórkowski and Agnieszka Mykowiecka for their support throughout the editorial process.

REFERENCES

- Anne ABEILLÉ (2003), *Treebanks: Building and Using Parsed Corpora*, Text, Speech and Language Technology, Springer.
- Jason BALDRIDGE, Sudipta CHATTERJEE, Alexis PALMER, and Ben WING (2007), DotCCG and VisCCG: Wiki and Programming Paradigms for Improved Grammar Engineering with OpenCCG, in Tracy Holloway KING and Emily M. BENDER, editors, *Proceedings of the GEAF07 Workshop*, pp. 5–25, Center for the Study of Language and Information (CSLI), Stanford, California, <http://csli-publications.stanford.edu/GEAF/2007/geaf07-toc.html>.
- Timothy BALDWIN, John BEAVERS, Emily M. BENDER, Dan FLICKINGER, Ara KIM, and Stephan OEPEN (2005), Beauty and the Beast: What running a broad-coverage precision grammar over the BNC taught us about the grammar

— and the corpus, in Stephan KEPSEK and Marga REIS, editors, *Linguistic Evidence: Empirical, Theoretical, and Computational Perspectives*, pp. 49–70, Mouton de Gruyter, Berlin.

Srinivas BANGALORE and Aravind K. JOSHI (1999), Supertagging: An Approach to Almost Parsing, *Computational Linguistics*, 25(2):237–262.

Emily M. BENDER (2008), Evaluating a Crosslinguistic Grammar Resource: A Case Study of Wambaya, in *Proceedings of ACL-08: HLT*, pp. 977–985, Association for Computational Linguistics, Columbus, Ohio.

Emily M. BENDER, Scott DRELLISHAK, Antske FOKKENS, Michael Wayne GOODMAN, Daniel P. MILLS, Laurie POULSON, and Safiyah SALEEM (2010), Grammar Prototyping and Testing with the LinGO Grammar Matrix Customization System, in *Proceedings of the ACL 2010 System Demonstrations*, pp. 1–6, Association for Computational Linguistics, Uppsala, Sweden.

Luciana BENOTTI (2009), Frolog: an Accommodating Text-Adventure Game, in *Proceedings of the Demonstrations Session at EAACL 2009*, pp. 1–4, Association for Computational Linguistics, Athens, Greece.

Alexandra BIRCH, Miles OSBORNE, and Philipp KOEHN (2007), CCG Supertags in Factored Statistical Machine Translation, in *Proceedings of the Second Workshop on Statistical Machine Translation*, pp. 9–16, Association for Computational Linguistics, Prague, Czech Republic.

Philippe BLACHE (2005), Property Grammars: A Fully Constraint-Based Theory, in Henning CHRISTIANSEN, Peter ROSSEN SKADHAUGE, and Jørgen VILLADSEN, editors, *Constraint Solving and Language Processing*, volume 3438 of *Lecture Notes in Computer Science*, pp. 1–16, Springer, Berlin Heidelberg.

Johan BOS, Stephen CLARK, Mark STEEDMAN, James R. CURRAN, and Julia HOCKENMAIER (2004), Wide-coverage Semantic Representations from a CCG Parser, in *Proceedings of the 20th International Conference on Computational Linguistics, COLING '04*, pp. 1240–1246, Association for Computational Linguistics, Stroudsburg, PA, USA.

Miriam BUTT, Helge DYVIK, Tracy Holloway KING, Hiroshi MASUICHI, and Christian ROHRER (2002), The Parallel Grammar Project, in *Proceedings of COLING-2002 Workshop on Grammar Engineering and Evaluation*, pp. 1–7, Taipei, Taiwan.

Miriam BUTT, Tracy H. KING, Marma-Eugenia NIÑO, and Frédérique SEGOND (1999), *A Grammar Writer's Cookbook*, Center for the Study of Language and Information (CSLI), Stanford, California.

Aoife CAHILL, Michael BURKE, Ruth O'DONOVAN, Stefan RIEZLER, Josef VAN GENABITH, and Andy WAY (2008), Wide-Coverage Deep Statistical Parsing Using Automatic Dependency Structure Annotation, *Computational Linguistics*, 34(1):81–124.

- Aoife CAHILL, Martin FORST, Michael BURKE, Mairead MCCARTHY, Ruth O'DONOVAN, Christian ROHRER, Josef VAN GENABITH, and Andy WAY (2005), Treebank-Based Acquisition of Multilingual Unification Grammar Resources, *Journal of Research on Language and Computation; Special Issue on "Shared Representations in Multilingual Grammar Engineering"*, 3(2):247–279.
- Aoife CAHILL, Mairéad MCCARTHY, Josef VAN GENABITH, and Andy WAY (2002), Parsing with PCFGs and automatic f-structure annotation, in *Proceedings of the 7th International Conference on LFG*, pp. 76–95, Center for the Study of Language and Information (CSLI), Palo Alto, California.
- Bob CARPENTER and Gerald PENN (1999), ALE 3.2 User's Guide, Technical Memo CMU-LTI-99-MEMO, Carnegie Mellon Language Technologies Institute.
- John CARROLL, Nicholas NICOLOV, O. SHAUMYAN, M. SMETS, and D. WEIR (2000), Engineering a wide-coverage lexicalized grammar, in *Proceedings of the Fifth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pp. 55–60, Paris, France.
- Eugene CHARNIAK (1994), *Statistical Language Learning*, MIT Press, Cambridge, Massachusetts.
- N. CHOMSKY (1956), Three models for the description of language, *Information Theory, IEEE Transactions on*, 2(3):113–124.
- Noam CHOMSKY (1957), *Syntactic Structures*, Mouton, The Hague.
- Lionel CLÉMENT and Alexandra KINYON (2003), Generating Parallel Multilingual LFG-TAG Grammars from a MetaGrammar, in *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pp. 184–191, Sapporo, Japan.
- Ann COPESTAKE (2002), *Implementing Typed Feature Structure Grammars*, Center for the Study of Language and Information (CSLI), Stanford, California.
- Benoît CRABBÉ, Denys DUCHIER, Claire GARDENT, Joseph LE ROUX, and Yannick PARMENTIER (2013), XMG : eXtensible MetaGrammar, *Computational Linguistics*, 39(3):591–629.
- Benoît CRABBÉ, Denys DUCHIER, Yannick PARMENTIER, and Simon PETITJEAN (2014), Constraint-driven Grammar Description, in Philippe BLACHE, Henning CHRISTIANSEN, Verónica DAHL, Denys DUCHIER, and Jørgen VILLADSEN, editors, *Constraints and Language*, pp. 93–121, Cambridge Scholar Publishing.
- Christine DORAN, Beth HOCKEY, Philip HOPELY, Joseph ROSENZWEIG, Anoop SARKAR, Srinivas BENGALORE, Fei XIA, Alexis NASR, and Owen RAMBOW (1997), Maintaining the Forest and Burning out the Underbrush in XTAG, in *Proceedings of the ACL Workshop on Computational Environments for Grammar Development and Language Engineering (ENVGRAM)*, pp. 30–37, Madrid, Spain.

Denys DUCHIER, Brunelle MAGNANA EKOUKOU, Yannick PARMENTIER, Simon PETITJEAN, and Emmanuel SCHANG (2012), Describing Morphologically-rich Languages using Metagrammars: a Look at Verbs in Ikota, in *Workshop on "Language technology for normalisation of less-resourced languages"*, 8th SALT MIL Workshop on Minority Languages and the 4th workshop on African Language Technology, pp. 55–60, Istanbul, Turkey, <http://aflat.org/files/saltmil8-aflat2012.pdf>.

Gregor ERBACH (1990), Grammar Engineering: Problems And Prospects, report on the Saarbrücken Grammar Engineering Workshop.

Roger EVANS and Gerald GAZDAR (1996), DATR: A Language for Lexical Knowledge Representation, *Computational Linguistics*, 22(2):167–213.

Dan FLICKINGER, Valia KORDONI, Yi ZHANG, Ant BRANCO, K. SIMOV, Petya OSENOVA, Catarina CARVALHEIRO, Francisco COSTA, and S CASTRO (2012), ParDeepBank : multiple parallel deep treebanking, in *Proceedings of the Eleventh International Workshop on Treebanks and Linguistic Theories (TLT11)*, pp. 97–108, Lisbon, Portugal.

Mary E. FOSTER, Michael WHITE, Andrea SETZER, and Roberta CATIZONE (2005), Multimodal Generation in the COMIC Dialogue System, in *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pp. 45–48, Association for Computational Linguistics, Ann Arbor, Michigan.

Claire GARDENT (2008), Integrating a unification-based semantics in a large scale Lexicalised Tree Adjoining Grammar for French, in *Proceedings of the 22nd International Conference on Computational Linguistics (COLING'08)*, pp. 249–256, Manchester, United Kingdom.

Claire GARDENT and German KRUSZEWSKI (2012), Generation for Grammar Engineering, in *INLG 2012 Proceedings of the Seventh International Natural Language Generation Conference*, pp. 31–39, Association for Computational Linguistics, Utica, Illinois.

Claire GARDENT, Yannick PARMENTIER, Guy PERRIER, and Sylvain SCHMITZ (2014), Lexical Disambiguation in LTAG using Left Context, in Zygmunt VETULANI and Joseph MARIANI, editors, *Human Language Technology. Challenges for Computer Science and Linguistics. 5th Language and Technology Conference, LTC 2011, Poznan, Poland, November 25-27, 2011, Revised Selected Papers*, volume 8387, pp. 67–79, Springer.

Julia HOCKENMAIER and Mark STEEDMAN (2002), Generative Models for Statistical Parsing with Combinatory Categorical Grammar, in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pp. 335–342, Association for Computational Linguistics, Philadelphia, Pennsylvania.

Kenneth HOETMER (2005), *Higher-Order Types for Grammar Engineering*, Master's thesis, University of Toronto, Department of Computer Science, <http://www.cs.toronto.edu/~hoetmer/hoetmerthesis.pdf>.

- Aravind K. JOSHI (1985), Tree Adjoining Grammars: How much Context - sensitivity is Required to Provide Reasonable Structural Descriptions?, in David R. DOWTY, Lauri KARTTUNEN, and Arnold ZWICKY, editors, *Natural Language Parsing*, pp. 206–250, Cambridge University Press, Cambridge.
- Aravind K. JOSHI, Leon S. LEVY, and Masako TAKAHASHI (1975), Tree Adjunct Grammars, *Journal of Computer and System Sciences*, 10(1):136–163.
- Laura KALLMEYER and Rainer OSSWALD (2013), Syntax-Driven Semantic Frame Composition in Lexicalized Tree Adjoining Grammars, *Journal of Language Modelling*, 1(2):267–330.
- Ronald M. KAPLAN and Joan BRESNAN (1982), Lexical-Functional Grammar: A Formal System for Grammatical Representations, in *The Mental Representation of Grammatical Relations*, pp. 173–281, MIT Press.
- Jan Tore LØNNING and Stephan OEPEN (2006), Re-Usable Tools for Precision Machine Translation, in *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pp. 53–56, Association for Computational Linguistics, Sydney, Australia.
- W. Detmar MEURERS, Gerald PENN, and Frank RICHTER (2002), A Web-based Instructional Platform for Constraint-Based Grammar Formalisms and Parsing, in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pp. 19–26, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA.
- Yusuke MIYAO, Takashi NINOMIYA, and Jun'ichi TSUJII (2005), Corpus-Oriented Grammar Development for Acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank, in Keh-Yih SU, Jun'ichi TSUJII, Jong-Hyeok LEE, and OiYee KWONG, editors, *Natural Language Processing – IJCNLP 2004*, volume 3248 of *Lecture Notes in Computer Science*, pp. 684–693, Springer, Berlin, Heidelberg.
- Laura PEREZ-BELTRACHINI, Claire GARDENT, and German KRUSZEWSKI (2012), Generating Grammar Exercises, in *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pp. 147–156, Association for Computational Linguistics, Montréal, Canada.
- Guy PERRIER (2000), Interaction Grammars, in *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pp. 600–606, Saarbruecken, Germany.
- Aarne RANTA (2011), *Grammatical Framework: Programming with Multilingual Grammars*, Center for the Study of Language and Information (CSLI), Stanford, California.
- Ivan A. SAG and Carl J. POLLARD (1987), Head-Driven Phrase Structure Grammar: An Informal Synopsis, CSLI Report 87–79, Stanford University.

- Stuart M. SHIEBER (1984), The Design of a Computer Language for Linguistic Information, in *Proceedings of the Tenth International Conference on Computational Linguistics*, pp. 362–366, Stanford, California.
- Mark STEEDMAN (1987), Combinatory grammars and parasitic gaps, *Natural Language & Linguistic Theory*, 5(3):403–439.
- Yael SYGAL and Shuly WINTNER (2011), Towards Modular Development of Typed Unification Grammars, *Computational Linguistics*, 37(1):29–74.
- Aline VILLAVICENCIO (2002), The acquisition of a unification-based generalised categorial grammar, Technical Report UCAM-CL-TR-533, Number 533, Computer Laboratory, University of Cambridge.
- Éric VILLEMONTÉ DE LA CLERGERIE (2010), Building factorized TAGs with meta-grammars, in *The 10th International Conference on Tree Adjoining Grammars and Related Formalisms - TAG+10*, pp. 111–118, New Haven, Connecticut, <https://hal.inria.fr/inria-00551974>.
- Fei XIA (1999), Extracting Tree Adjoining Grammars from bracketed corpora, *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, pp. 398–403.
- Yao-zhong ZHANG, Takuya MATSUZAKI, and Jun'ichi TSUJII (2009), HPSG Supertagging: A Sequence Labeling View, in *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pp. 210–213, Association for Computational Linguistics, Paris, France.
- Yi ZHANG, Valia KORDONI, Aline VILLAVICENCIO, and Marco IDIART (2006), Automated Multiword Expression Prediction for Grammar Engineering, in *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pp. 36–44, Association for Computational Linguistics, Sydney, Australia.

This work is licensed under the Creative Commons Attribution 3.0 Unported License.

<http://creativecommons.org/licenses/by/3.0/>



The CoreGram project: theoretical linguistics, theory development, and verification

Stefan Müller

German Grammar Group,
Institut für Deutsche und Niederländische Philologie,
Freie Universität Berlin

ABSTRACT

This paper describes the CoreGram project, a multilingual grammar engineering project that develops HPSG grammars for several typologically diverse languages that share a common core. The paper provides a general motivation for doing theoretical linguistics the way it is done in the CoreGram project, and is therefore not exclusively targeted at computational linguists. I argue for a constraint-based approach to language rather than a generative-enumerative one and discuss issues of formalization. Recent advantages in language acquisition research are mentioned and conclusions on how theories should be constructed are drawn. The paper discusses some of the highlights in the implemented grammars, gives a brief overview of central theoretical concepts and their implementation in the TRALE system, and compares the CoreGram project with other multilingual grammar engineering projects.

Keywords:
Universal Grammar, Head-Driven Phrase Structure Grammar, multilingual grammar engineering, TRALE, HPSG, theoretical linguistics

1

OVERVIEW AND MOTIVATION

The goal of the CoreGram project is to contribute to a better understanding of the constraints for specific human languages and of the constraints holding for human language in general or for certain language groups. To reach this goal we develop several large-scale computer-processable grammar fragments of several typologically diverse languages using a common core grammar. We believe that lin-

guistic theories have reached a level of complexity that makes it necessary to implement grammars in order to verify their consistence (Section 1.2).

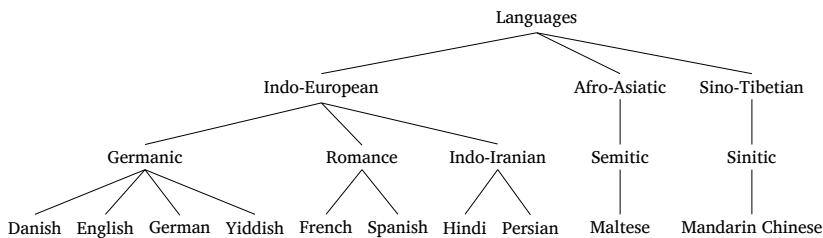
The theoretical framework we use is Head-Driven Phrase Structure Grammar (HPSG, Pollard and Sag 1994; Sag 1997) in the version that is described in detail in Müller 2013b. We are currently working on the following languages:

- German (Müller 2007b, 2009b, 2012; Müller and Ørsnes 2011, 2013a; Müller 2014a, 2015a)
- Danish (Ørsnes 2009; Müller 2009b, 2012; Müller and Ørsnes 2011, 2013a,b, 2015)
- Persian (Müller 2010b; Müller and Ghayoomi 2010; Müller *et al.* In Preparation)
- Maltese (Müller 2009a)
- Mandarin Chinese (Lipenkova 2009; Müller and Lipenkova 2009, 2013, In Preparation)
- Yiddish (Müller and Ørsnes 2011)
- English (Müller 2009b, 2012; Müller and Ørsnes 2013a)
- Hindi
- Spanish
- French

These languages belong to diverse language families (Indo-European, Afro-Asiatic, Sino-Tibetan), and among the Indo-European languages, the languages belong to different groups (Germanic, Romance, Indo-Iranian). Figure 1 provides an overview.

For implementation we use the TRALE system (Meurers *et al.* 2002; Penn 2004), which allows for a rather direct encoding of HPSG analyses (Melnik 2007). The grammars of German, Danish, Persian,

Figure 1:
Language families and groups of the languages covered in the CoreGram project



Maltese, and Mandarin Chinese are of non-trivial size and can be downloaded at <http://hpsg.fu-berlin.de/Projects/CoreGram.html>. They are also part of the next version of the Grammix virtual machine (Müller 2007a). The grammars of Yiddish and English are toy grammars that are used to verify cross-linguistic analyses of special phenomena. The grammar of Hindi is also a small fragment. I developed it together with Shravan Vasishth during a seminar at the University of Potsdam in 2006. The work on Spanish and French is part of ongoing work in the Sonderforschungsbereich 632, which started in 2012. See (Bildhauer 2008) for an implemented grammar of Spanish that will be converted into the format of the grammars mentioned above. As of February 2015, the grammars and test suites have the sizes shown in Table 1.

As is explained in Section 5.2, the grammars share some code. 14,523 lines are shared by at least two grammars. The lexical entries are those lexical items that are specified by the grammar writer, the lexical items are the lexical entries plus those lexical items that are licensed by lexical rules. As the table shows, there is a huge difference between languages with a rich inflection and clitics on the one hand, and languages without inflection, such as Mandarin Chinese, on the other hand. The test suite for the German grammar is the largest; however, many phenomena that work for German are also implemented for other languages but are not part of the respective test suites yet. An example for this is coordination. For Persian we have three test suites: The first consists of the one hundred sentences used by Bahrani

	lines	lexical entries	lexical items	test items
German	18,060	300	11,150	361
Danish	16,373	120	1,325	231
Persian	17,640	1,981	65,847	100 + 216 + 130
Maltese	10,046	95	2,362	134
Mandarin Chinese	10,718	564	855	104
Yiddish	10,383	26	60	34
English	9,955	49	97	23
French	11,831	43	61	20
Hindi	8,907	32	90	46

Table 1:
Sizes of
grammars
and test suits
included in
CoreGram

et al. (2011, p. 405–406) in their GPSG project. 216 sentences comprise our main test suite, which mainly contains examples from the literature, constructed examples, and ungrammatical strings that were discovered during grammar development. The third test suite contains 130 randomly selected sentences from the Peykare corpus (Bijankhan 2004), a balanced corpus provided by the University of Tehran and the Higher Council for Informatics of Iran.

We believe that working out large-scale computer-implemented grammars is the best way to verify the consistency of linguistic theories. Much linguistic work is published in journal articles, but the underlying assumptions of each article may be different, so that it is difficult to imagine a coherent view that incorporates all insights. Even for articles written by the same author there is no guarantee that basic assumptions are shared between articles, since it can take several years for individual papers to be published. Hence, I believe that books are the best format for describing linguistic theories, and ideally, such theories should be backed up by computer implementations. The larger fragments of the CoreGram project will be documented in a series of book publications. The first book in this series was (Müller 2007b), which describes a fragment of German that is implemented in the grammar BerliGram. Three further books are in preparation and will be submitted to the series *Implemented Grammars by Language Science Press*: one on the Persian Grammar developed in the PerGram project (Müller *et al.* In Preparation), one on the Danish Grammar developed in the DanGram project (Müller and Ørsnes 2015), and one on the Mandarin Chinese grammar developed in the ChinGram project (Müller and Lipenkova In Preparation).

The remainder of this paper is structured as follows: the rest of this section describes desiderata for linguistic theories and discusses the importance of formalization, with special focus on mainstream theories and research programs such as GB, Minimalism, and Construction Grammar. Section 2 discusses the way in which theories of our linguistic knowledge should be constructed. It compares Minimalist approaches with the more data-driven approach taken in the CoreGram project. Many Minimalist approaches start with certain assumptions and then try to show that it is possible to account for all languages with these fundamental assumptions. Recent evidence from language acquisition shows that this is not a viable research strategy and that

grammars should be motivated on a language-specific basis. The generative research tradition was criticized by typologists like Haspelmath (2010a), Dryer (1997), and Croft (2001, Section 1.4.2–1.4.3), claiming that the generative methodology was fundamentally flawed and that descriptive categories should be language-specific. I will show why the CoreGram project does not run into methodological problems and suggest a middle way between mainstream generative grammar and radical views like the one held by Croft. Following the discussion of theory construction in Section 2, Section 3 shows some highlights from the various CoreGram grammars. Section 4 discusses basic theoretical assumptions for the treatment of valence, constituent order, morphology, semantics, and information structure, and Section 5 provides details on how things are implemented in the TRALE system. Section 6 compares the CoreGram project with other multilingual projects like ParGram and DELPH-IN. Finally, Section 7 deals with evaluation, grammar profiling and testing, and Section 8 draws some conclusions.

1.1 *Desiderata for linguistic theories*

This section discusses desiderata for linguistic theories and shows that the framework that is assumed in the CoreGram project, namely HPSG, fulfils all of them.

1.1.1 Non-transformational, constraint-based approach

While psycholinguistic experiments at first seemed to confirm the Derivational Theory of Complexity (Miller and McKean 1964; Savin and Perchonock 1965; Clifton and Odom 1966), so that Chomsky assumed it to be correct until 1968 (Chomsky 1976, p. 249–250), it was later shown that the initial experiments were flawed and that transformations are not psycholinguistically real (Fodor *et al.* 1974, p. 324). Since then it has become customary to say that transformations are metaphors (for instance in Chomsky 2001, Footnote 4). This, of course, begs the question why one should formulate one's theories in metaphors (see also Jackendoff 2011, p. 599). This question is even more pressing since a lot of Minimalist theorizing is now done under the label of Biolinguistics and the assumed processes are claimed to be psycholinguistically real. For instance, Chomsky (2001, p. 11, 12, 15; 2007, p. 3, 12; 2008, p. 138, 145, 146, 155) refers to aspects of pro-

cessing and memory requirements (see also Marantz 2005, p. 440, and Richards 2015). However, structure building processes that start with the combination of words and assume later reorderings are highly implausible from a psycholinguistic point of view. As was pointed out by Labelle (2007), human short-term memory is simply too limited to be able to compute complex structure in the way it is envisaged by current Minimalist theories. Models that crucially rely on the order of application of combinatorial operations fail, since we neither use our linguistic knowledge exclusively bottom-up nor exclusively top-down. Phillips (2003) suggested a theoretical variant that allows for incremental parsing, but first, this is tailored towards parsing and ignores generation, and second, it is incompatible with much of the rest of the Minimalist theories.

The way out of all of these problems is a clear separation between competence and performance and a declarative, constraint-based statement of linguistic constraints that do not make any claims about the order of constraint application (Sag and Wasow 2011; Jackendoff 2011, p. 600). The order of application is constrained by performance models, which are an important part of a theory about language and have to be combined with competence models. Proponents of usage-based approaches often reject the competence–performance distinction, but as soon as a grammar contains grammar rules or schemata that can be applied recursively, one has to explain why sentences have a maximal length, why we cannot do center-embeddings with more than four levels and so on. An example of such a schema would be a schema that licences relative clauses. Since relative clauses may contain NPs, and NPs may in turn contain relative clauses, we have a recursive grammar that licences infinitely many sentences. (Bannard *et al.* 2009, proponents of Construction Grammar, use a context-free grammar, a kind of grammar that clearly allows for recursive structures). The limitations with respect to sentence length and embedding are due to factors such as our short term memory and have to be explained by a performance model that takes these factors into account (Gibson 1998).

HPSG is a constraint-based theory which does not make any claims about the order of application of combinatorial processes. Theories in this framework are just statements about relations between linguistic objects or between properties of linguistic objects

and, hence, compatible with psycholinguistic findings and processing models (Sag and Wasow 2011).

Pullum and Scholz (2001) and Pullum (2007) discuss further advantages of model-theoretic, and hence constraint-based, proposals: they allow the construction of partial structures, can deal with graded grammaticality, and no claims about the infinitude of language are necessary.

As Pullum and Scholz note we can assign the structure in Figure 2 to the fragment *and of the* in a sentence like (1):

- (1) That cat is afraid of the dog and of the parrot.

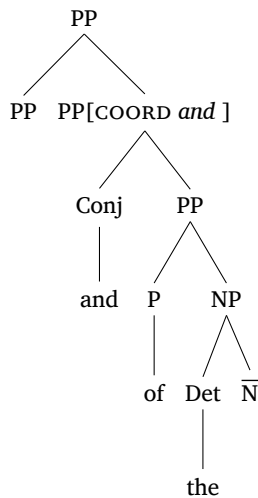


Figure 2:
Structure of
the fragment
“and of the”
following Pullum
and Scholz
(2001, p. 32)

If we hear *the* we know that an \bar{N} will follow, if we hear *of*, we will know that *of* will head a PP. *and* usually is part of symmetric coordinations, so we know that the first part of the coordination will be a PP too. So, the information from lexical items and dominance schemata licenses a complex structure in constraint-based models, while nothing is generated in generative-enumerative models. *and of the* is just not a member of a set of well-formed expressions.

This property of constraint-based approaches also comes in handy when we want to explain the robustness of human sentence processing. In the case of unknown words, information from syntax and semantics can be used to draw inferences about the material that is missing. For instance in an utterance like (2), in which information about

XXX is missing, the hearer can infer that XXX must be a verb and that it must have three arguments. (This argument is also due to Pullum and Scholz 2001, Section 3.3, who make it in a slightly different form.)

(2) Could you XXX me the salt, please?

Hence, the hearer arrives at a prototypical ditransitive verb, a verb of change of possession, that is, something like *give* or *pass*. In current Minimalist models nothing would happen since structure building and movement is triggered by lexical items and their features, but if they are absent or unknown, the derivation does not start.¹

While constraint-based approaches can explain markedness of structures by the number and strength of constraints that are violated by a given example, this is not possible in generative-enumerative approaches. For a discussion of an attempt to incorporate markedness into the picture of generative-enumerative models, see (Chomsky 1975, Chapter 5; Chomsky 1964). For a rejection of these proposals see (Pullum and Scholz 2001, p. 29).

Finally, constraint-based models do not have to make any claims about the infinitude of language. While it is usually claimed that language is infinite by proponents of generative-enumerative proposals,

¹ Chomsky (2007, p. 6) writes: *A Merge-based system of derivation involves parallel operations. Thus if X and Y are merged, each has to be available, possibly constructed by (sometimes) iterated Merge. The process has a loose resemblance to early theories of generalized transformations, abandoned in the early 1960s for good reasons, now resurrected in a far simpler form for better reasons. But a generative system involves no temporal dimension. In this respect, generation of expressions is similar to other recursive processes such as construction of formal proofs. Intuitively, the proof “begins” with axioms and each line is added to earlier lines by rules of inference or additional axioms. But this implies no temporal ordering. It is simply a description of the structural properties of the geometrical object “proof.” The actual construction of a proof may well begin with its last line, involve independently generated lemmas, etc. The choice of axioms might come last. The same is true of generation vs production of an expression, a familiar competence–performance distinction.* This seems to indicate a constraint-based position. However, even if a constraint-based view is assumed in principle, there are lots of unsolved problems with specific Minimalist proposals. For instance, some approaches assume the existence of unvalued features that acquire a value during a derivation. What happens if the information about these features is not available to the hearer? According to Minimalist theorizing, the derivation should crash. But as (2) just shows, it does not crash but results in partial, underspecified information.

no such claims are necessary in the model-theoretic world. Those who claim that language is infinite and biologically real are faced with the paradox that infinitely many members of the infinite set are not biologically real, that is, they can never be realized because of our limited resources (Postal 2009). Model-theoretic approaches do not have to assume infinite sets only to throw away most of them because of performance considerations; rather, they pair the performance model with the competence model directly and therefore end up with theories that are psycholinguistically and biologically plausible.

As I showed in (Müller 2010a, Chapter 3.6.4; 2013c), the combinatory operations of Minimalism as defined in (Chomsky 2008) and (Stabler 2001) correspond to three of the schemata used in HPSG grammars since at least (Pollard and Sag 1994): Merge corresponds to the Head-Specifier Schema and the Head-Complement Schema of HPSG, and Move corresponds to the Head-Filler Schema. So, ideology and rhetorics aside, what we have here is a constraint-based, declarative formalization of Minimalist proposals. Of course, a lot of questions have to be asked about current Minimalist analyses; some of them will be addressed in Section 2.

1.1.1.2 Sign-based, parallel architecture

As was shown by Marslen-Wilson (1975) in the 1970s and confirmed later by many studies, we process linguistic and non-linguistic information as soon as it is available to us, and there is no ordering of strictly encapsulated modules of processing. Tanenhaus *et al.* (1995, 1996) used eye-tracking techniques to establish that we know the referents of NPs even if we just heard a determiner and an adjective. The authors showed that stress on the adjective is interpreted immediately as a sign of contrast, leading to right inferences being drawn and the only possible object in a specific scene being looked at. The studies demonstrated that constraints from phonology, syntax, and information structure are evaluated immediately and that information is processed as soon as it is available. We can use it and draw inferences even though we may not be able to assign a full syntactic structure to a certain phrase yet. Such findings are compatible with architectures that assume that all linguistic levels are accessible simultaneously such as HPSG, LFG, Construction Grammar and Jackendoff's Parallel Architecture (Jackendoff 1999, 2011; Kuhn 2007).

1.1.3 Possibility to include
non-headed and phrasal constructions

I agree with Jackendoff (2008, 2011), Jacobs (2008), Sag (2010), and others that one needs more than some very general binary-branching schemata to deal with language in its full richness in non-stipulative ways. For instance, Jackendoff (2011) pointed out that none of the elements in N-P-N constructions like (3) can be identified as the head. Instead of the usual head-argument structures (\bar{X} or equivalent), Jackendoff (2008) suggests that the N, P, and N are combined into an NP or advP and that the structure as such is unheaded.

- (3) student after student
[_{NP/advP} N-P-N]

The sentences in (4), which were mentioned by Jackendoff and Pinker (2005, p. 220) and whose German equivalent was discussed in detail by Jacobs (2008), are further examples of a construction that is best handled as a headless construction.²

- (4) a. Off with his head!
b. Into the trunk with you!

Hence, I believe that additional schemata or phrasal constructions in the sense of Construction Grammar or Simpler Syntax (Culicover and Jackendoff 2005) are needed. It is an empirical issue to what extent phrasal constructions are needed and where Merge-like combinations together with a rich lexicon are sufficient or rather necessary, and the CoreGram project contributes to this discussion. See, for instance, (Müller and Lipenkova 2009) for a phrasal treatment of serial verbs in Mandarin Chinese, and (Ørsnes 2009) for a phrasal treatment of preposed negation in Danish.

1.1.4 Core-periphery distinction

Given that the name of the project is CoreGram, some words on the core-periphery distinction are in order.

Chomsky (1981, p. 7–8) suggests dividing languages into a core part and a periphery. All regular parts belong to the core. The core

² See (Müller 2011) for an account of Jacobs' data using an empty verbal head and an abstract antipassive morpheme, and (Müller 2015b, Section 12.11.9.1) for discussion.

grammar of a language is assumed to be an instance of Universal Grammar (UG), the genetically determined innate language faculty of human beings. Idioms and other irregular parts of a language belong to the periphery. Critics of Chomsky's Principle and Parameters approach have pointed out that a rather large proportion of our languages consist of, or interact with, irregular constructions, and that the borders between core and periphery cannot be drawn easily and are often motivated theory-internally only (Jackendoff 1997, Chapter 7; Culicover 1999; Ginzburg and Sag 2000, p. 5; Newmeyer 2005, p. 48; Kuhn 2007, p. 619). For instance, Nunberg *et al.* (1994) pointed out that many English idioms interact with syntax. See also (Müller 2010a, p. 350) for interactions of idioms with verb placement, V2, and passive in German.

So, I do not think that it is justified to ignore phenomena that are claimed to belong to the periphery. Rather, I agree with Bender and Flickinger (1999) and Bender (2008, p. 20–21) that studying phenomena that are traditionally assigned to the periphery may discriminate between possible analyses of the alleged core phenomena.

It should be noted however that the methodology described in Section 2.2 results in a separation of core and periphery: Core constraints in the sense of the CoreGram project are those constraints that are shared by at least two languages. All other constraints are specific for a single language and constitute the periphery of the respective languages. This notion of periphery is very different from the Chomskian one: Depending on the version of the theory we look at, the periphery in the Chomskian sense includes phenomena like Exceptional Case Marking (Chomsky 1981, p. 70), which are common in a lot of languages and are, hence, assigned to the core in our setting. Another example is Jackendoff's *student after student* construction, which can be found in several languages (König and Moyses-Faurie 2009) and which we would count to the core grammar, but which would probably not be seen as part of the core grammar in the Chomskian sense. Please refer to Section 2.2 for details of the organization of CoreGram and to (Müller 2014c) for a general discussion of the core–periphery distinction.

1.2 *Formalization, computer implementations,
and theory verification*

The work of Noam Chomsky pioneered the formalization of linguistic theories. In his early writings he states that formalization is necessary for progress in linguistics:

Precisely constructed models for linguistic structure can play an important role, both negative and positive, in the process of discovery itself. By pushing a precise but inadequate formulation to an unacceptable conclusion, we can often expose the exact source of this inadequacy and, consequently, gain a deeper understanding of the linguistic data. More positively, a formalized theory may automatically provide solutions for many problems other than those for which it was explicitly designed. Obscure and intuition-bound notions can neither lead to absurd conclusions nor provide new and correct ones, and hence they fail to be useful in two important respects. I think that some of those linguists who have questioned the value of precise and technical development of linguistic theory have failed to recognize the productive potential in the method of rigorously stating a proposed theory and applying it strictly to linguistic material with no attempt to avoid unacceptable conclusions by ad hoc adjustments or loose formulation. (Chomsky 1957, p. 5)

In a book that appeared some years later, Manfred Bierwisch argued for machine processable implementations of theoretical analyses:

It is therefore very possible that the rules that we have formulated generate sentences which are outside of the set of grammatical sentences in an unpredicted way; that is, they violate grammaticality due to properties that we did not deliberately exclude from our analysis. This is what is meant by the statement that a grammar is a hypothesis about the structure of a language. A systematic examination of the implications of a grammar that is appropriate for natural languages surely is a task that cannot be solved manually anymore. This task could be approached by implementing the grammar as a calculation task on a computer, so that it can be verified

to which degree the result deviates from the language to be described. (translated from Bierwisch 1963, p. 163³)

I wholeheartedly agree with Bierwisch's statement, given that after the time of his writing enormous headway has been made both empirically and theoretically. For instance, Ross (1967) discovered constraints on reordering constituents and non-local dependencies; Perlmutter (1978) discovered unaccusative verbs in the 1970s; and theories were developed for dealing with case (Yip *et al.* 1987; Meurers 1999c; Przepiórkowski 1999), verbal complex formation, and partial fronting (Evers 1975; Grewendorf 1988; Hinrichs and Nakazawa 1994; Kiss 1995; G. Müller 1998; Meurers 1999b; Müller 1999, 2002; De Kuthy 2002). All these phenomena, and a lot more, interact!

To emphasize this point, I give another quotation from Steve Abney, who worked within the GB framework:

A goal of earlier linguistic work, and one that is still a central goal of the linguistic work that goes on in computational linguistics, is to develop grammars that assign a reasonable syntactic structure to every sentence of English, or as nearly every sentence as possible. This is not a goal that is currently much in fashion in theoretical linguistics. Especially in Government-Binding theory (GB), the development of large fragments has long since been abandoned in favor of the pursuit of deep principles of grammar. The scope of the problem of identifying the correct parse cannot be appreciated by examining behavior on small fragments, however deeply analyzed. Large fragments are not just small fragments several times over—there is a qualitative change when one be-

³ Original from (Bierwisch 1963, p. 163): *Es ist also sehr wohl möglich, daß mit den formulierten Regeln Sätze erzeugt werden können, die auch in einer nicht vorausgesehenen Weise aus der Menge der grammatikalisch richtigen Sätze herausfallen, die also durch Eigenschaften gegen die Grammatikalität verstoßen, die wir nicht wissentlich aus der Untersuchung ausgeschlossen haben. Das ist der Sinn der Feststellung, daß eine Grammatik eine Hypothese über die Struktur einer Sprache ist. Eine systematische Überprüfung der Implikationen einer für natürliche Sprachen angemessenen Grammatik ist sicherlich eine mit Hand nicht mehr zu bewältigende Aufgabe. Sie könnte vorgenommen werden, indem die Grammatik als Rechenprogramm in einem Elektronenrechner realisiert wird, so daß überprüft werden kann, in welchem Maße das Resultat von der zu beschreibenden Sprache abweicht.*

gins studying large fragments. As the range of constructions that the grammar accommodates increases, the number of undesired parses for sentences increases dramatically. (Abney 1996, p. 20)

In addition, it is a goal of much current linguistic theorizing to formulate constraints that hold for all languages or at least for certain classes of languages. As a consequence, it is not sufficient to study the interaction between phenomena solely on the basis of one language: Changing the constraints for a certain phenomenon in one language may be compatible with all phenomena that are relevant for the language under discussion, but it may well be the case that unexpected interactions with other phenomena in another language emerge. Verifying the consequences of a simple change of a principle therefore results in a complexity that cannot be handled by human beings. It is therefore necessary to formalize the theories in a way that makes them implementable as computer-processable grammars. After checking the grammar for consistency, a computer grammar can be used to analyze systematically constructed test suites containing thousands of grammatical sentences and ungrammatical word sequences or large corpora containing naturally occurring data. Such parses can be used to verify that the grammar makes the right predictions as far as the empirical facts are concerned (Müller 1999, Chapter 22; Oepen and Flickinger 1998; Bender 2008, Müller 2013a, Section 3.7.2; Müller 2015b, Section 3.7.2). In addition, generators can be used to produce utterances that correspond to a certain meaning. If ill-formed strings are generated, this is an indicator for missing constraints in the grammar.

After more than 55 years of work in transformational grammar, one has to note that there are no large-scale implemented fragments on the basis of transformational analyses. Chomsky made important contributions to the theory of formal languages that are still relevant in computer science (Chomsky 1959), but in 1981 he turned his back on precisely worked-out solutions:

I think that we are, in fact, beginning to approach a grasp of certain basic principles of grammar at what may be the appropriate level of abstraction. At the same time, it is necessary to investigate them and determine their empirical adequacy by developing quite specific mechanisms. We should,

then, try to distinguish as clearly as we can between discussion that bears on leading ideas and discussion that bears on the choice of specific realizations of them. (Chomsky 1981, p. 2–3)

He made it explicit in a letter to *Natural Language and Linguistic Theory*:

Even in mathematics, the concept of formalization in our sense was not developed until a century ago, when it became important for advancing research and understanding. I know of no reason to suppose that linguistics is so much more advanced than 19th century mathematics or contemporary molecular biology that pursuit of Pullum's injunction would be helpful, but if that can be shown, fine. For the present, there is lively interchange and exciting progress without any sign, to my knowledge, of problems related to the level of formality of ongoing work. (Chomsky 1990, p. 146)

The consequence of this change is a very large number of publications in Mainstream Generative Grammar, many of which make incompatible assumptions, so that it is not clear how insights from different publications can be combined. A case in point are the many different definitions of the rather central concept of government (see Aoun and Sportiche 1983 for an overview).

This was repeatedly criticized in the 1980s, for instance by the practitioners of GPSG (Gazdar *et al.* 1985, p. 6; Pullum 1985, 1989; Pullum 1991, p. 48; Kornai and Pullum 1990). The lack in precision, missing details,⁴ and frequent changes in the basic assumptions⁵ resulted in the absence of large-scale computer implementations that incorporate insights from Mainstream Generative Grammar. There are some implementations that borrow from GB/MP models or from ideas from Mainstream Generative Grammar (Petrick 1965; Zwicky *et al.* 1965; Friedman 1969; Friedman *et al.* 1971; Morin 1973; Marcus 1980; Abney and Cole 1986; Kuhns 1986; Correa 1987; Stabler 1987,

⁴ See, for instance, (Kuhns 1986, p. 550), (Crocker and Lewin 1992, p. 508), (Kolb and Thiersch 1991, p. 262), and (Kolb 1997, p. 3) on precision; and (Freidin 1997, p. 580), (Veenstra 1998, p. 25, 47), (Lappin *et al.* 2000, p. 888), and (Stabler 2010, p. 397, 399, 400) on missing details.

⁵ See, for instance, (Kolb 1997, p. 4), (Fanselow 2009), and the quote by Stabler on p. 37.

1992, 2001; Kolb and Thiersch 1991; Fong 1991; Crocker and Lewin 1992; Lohnstein 1993; Fordham and Crocker 1994; Nordgård 1994; Veenstra 1998; Niyogi and Berwick 2005), but these implementations usually do not employ transformations or deviate in other crucial ways from theoretical work. See (Kay 2011, p. 10) for discussion of early transformational systems, and (Müller 2013a, Section 3.7.2; Müller 2015b, Section 3.7.2) for further discussion of GB and Minimalist systems.

There are two implementations that can be regarded as implementations of Minimalist ideas. I will comment on them briefly: Stabler (2001) shows how Kayne's theory of remnant movement can be formalized and implemented. However, his implementation does not use transderivational constraints, does not have numerations, has no Agree (see Fong 2014, p. 132), and so on. Stabler's grammars are small-scale fragments that can be considered proofs of concept, but nothing more. They only deal with syntax; there is no morphology⁶, no treatment of multiple agreement (Stabler 2011, Section 27.4.3), and no semantics; and neither PF nor LF processes are modeled.⁷ Another implementation that uses Minimalist Grammar as a framework is the one of Niyogi and Berwick (2005). This grammar has 347 lexical entries and covers a lot of argument alternations. It is probably the largest implementation of Minimalist ideas. However, it differs from theoretical proposals in not using numerations, in having six or seven (if OPTIONAL MERGE is counted) rules for combination of material rather than just internal and external merge, and in not using labelling but rather a Categorical Grammar-like functor argument approach. In (Berwick *et al.* 2011), a later paper that explicitly discusses computer implementations, this system is not mentioned.

⁶The test sentences have the form in (i).

- (i) a. the king will -s eat
- b. the king have -s eat -en
- c. the king be -s eat -ing
- d. the king -s will -s have been eat -ing the pie

⁷See, for instance, (Sauerland and Elbourne 2002, p. 285) for suggestions on PF and LF movement that includes the deletions of parts of copies. The implementation of such analyses is probably non-trivial.

The grammars and the processing system developed by Sandiway Fong (Fong and Ginsburg 2012; Fong 2014) have a similar status: The grammar fragments are small, encode syntactic aspects like Labeling directly in the phrase structure rules (Fong and Ginsburg 2012, Section 4), and hence fall far behind \bar{X} Theory. The grammars do not have a morphology component, and Spell-Out is not implemented; therefore, this system neither parses nor generates a single sentence from any natural language.⁸

The reason for the absence of large-scale fragments in the framework of GB/MP is probably that the basic assumptions that are made in the Minimalist community are changing very frequently:

In Minimalism, the triggering head is often called a *probe*, the moving element is called a *goal*, and there are various proposals about the relations among the features that trigger syntactic effects. Chomsky (1995, p. 229) begins with the assumption that features represent requirements which are checked and deleted when the requirement is met. The first assumption is modified almost immediately so that only a proper subset of the features, namely the ‘formal’, ‘uninterpretable’ features are deleted by checking operations in a successful derivation (Collins, 1997; Chomsky 1995, §4.5). Another idea is that certain features, in particular the features of certain functional categories, may be initially unval-

⁸The following claim by Berwick *et al.* (2011, p. 1221) is therefore simply wrong: *But since we have sometimes adverted to computational considerations, as with the ability to “check” features of a head/label, this raises a legitimate concern about whether our framework is computationally realizable. So it is worth noting that the copy conception of movement, along with the locally oriented “search and labeling” procedure described above, can be implemented computationally as an efficient parser; see Fong, 2011, for details.* One cannot claim that one has an efficient implementation if the software under consideration does not parse any sentence at all, since it might be possible that the implementation of the missing parts is extremely complex and the resulting program would be inefficient. As was noted above, Fong does not implement Labeling as it was introduced in Chomsky’s papers (for instance, Chomsky 2008, 2013), but simply uses definite clause grammars. In fact, neither of the two Chomsky papers is implementable, since the description of Labeling is not worked out in detail. Crucial cases are missing in the 2008 paper and the 2013 paper is vague in some places and inconsistent in others (Müller 2013c).

ued, becoming valued by entering into appropriate structural configurations with other elements (Chomsky 2008; Hiraiwa, 2005). And some recent work adopts the view that features are never deleted (Chomsky 2007, p. 11). These issues remain unsolved. (Stabler 2010, p. 397)

Developing a grammar fragment takes at least three years. Large grammars accumulate the knowledge of several researchers which has crystallized in international cooperations over the course of several years or even decades. However, such a process is blocked when basic assumptions are changed frequently (see also Fanselow 2009, p. 138).

The same criticism that applies to GB/Minimalism applies to Construction Grammar: The basic notions and key concepts are hardly ever made explicit with the exception of Sign-Based Construction Grammar (Sag 2010, 2012), which is an HPSG variant, Embodied Construction Grammar (Bergen and Chang 2005), which uses feature value matrices and is equivalent to HPSG (see Müller 2010a, Chapter 9.6, for a discussion of both theories), and Fluid Construction Grammar (Steels 2011).⁹

2 THE POVERTY OF THE STIMULUS AND MOTIVATION OF ANALYSES

In this section, I first describe recent advances in research on language acquisition and then show how the data-driven, bottom-up approach to theory development that is followed in the CoreGram project works in detail.

2.1 *Language acquisition and linguistic theory*

As is argued in (Müller 2010a, Chapter 11.4) and (Müller 2015b, Chapter 12.4), HPSG is compatible with UG-based models of language acquisition such as, for instance, the one by Fodor (1998). See (Fodor

⁹Steels (2013, p. 153) sees Fluid Construction Grammar as a toolkit for the implementation of various Construction Grammar ideas. So in this view, it would not be a theory or framework but rather a system such as TRALE or LKB, which are discussed below. Van Trijp (2013, 2014) sees Fluid Construction Grammar as a framework and compares it with HPSG. For a detailed discussion of specific analyses formalized in Fluid Construction Grammar and van Trijp's core assumptions, see (Müller 2015b, Section 9.6.4).

2001, p. 385) for an explicit remark to that end. However, in recent years evidence has accumulated showing that arguments for innate, language-specific knowledge are very weak. For instance, Johnson (2004) showed that Gold's (1967) proof that natural languages are not identifiable in the limit by positive data alone is irrelevant for discussions of human language acquisition. Furthermore, there is evidence that the input that humans have is sufficiently rich to acquire structures which were thought by Chomsky (1971, p. 29–33; 2013, p. 39) and others to be impossible to acquire: Bod (2009) showed how syntactic structures could be derived from an unannotated corpus by Unsupervised Data-Oriented Parsing. He assumed that language is organized in chunks, and it has been described how children can acquire the fact that linguistic expressions are combined from smaller parts into larger units (see Estigarribia 2009 for acquisition of fragments starting at the right periphery of utterances). Given this prerequisite, Bod explained how Chomsky's auxiliary inversion data can be captured even if the input does not contain the data that Chomsky claims to be necessary (see also Eisenberg 1992, Pullum and Scholz 2002, and Scholz and Pullum 2002 for other Poverty of the Stimulus arguments). Input-based models of language acquisition in the spirit of Tomasello (2003) seem highly promising and can, in fact, explain language acquisition data better than previous UG-based models (Freudenthal *et al.* 2006, 2009). I have argued that the results from language acquisition research in the Construction Grammar framework can be carried over to HPSG, even in its lexical variants (Müller 2010a, 2015b; Müller and Wechsler 2014, Section 9).¹⁰ If language acquisition is input-based and language-specific innate knowledge is minimal, as assumed by Chomsky (1995) and Hauser *et al.* (2002), or even non-existing, this has important consequences for the construction of linguistic theories: Proposals that assume more than 400 morpho-syntactic categories that are all innate and that play a role in all languages of the world, even though they are not directly observable in many languages (Cinque

¹⁰In fact, I believe that a lexical treatment of argument structure is the only one that is compatible with the basic tenets of theories like Categorical Grammar (CG), Lexical Functional Grammar (LFG), Construction Grammar, and HPSG that adhere to lexical integrity (Bresnan and Mchombo 1995). For discussion, see (Müller 2006), (Müller 2010a, Chapter 11.11), (Müller 2010b), (Müller 2013c), and (Müller and Wechsler 2014).

and Rizzi 2010, p. 55, 57), have to be rejected right away. Furthermore, one cannot argue for empty functional projections in language X on the basis of overt morphemes in language Y. This has, for instance, been done for Topic projections that are assumed for languages without topic morphemes on the basis of the existence of a topic morpheme in Japanese and Gungbe. Similarly, functional projections for object agreement (AgrO) have been proposed for languages like English and German on the basis of Basque data, even though neither English nor German has object agreement. Since German children do not have any evidence from Basque, they would not be able to learn that there are projections for object agreement, and hence this fact would have to be known in advance. Neither can the existence of postpositions and agreement in Hungarian be seen as evidence for AgrO projections and hidden movement processes in English as assumed in the analysis by Hornstein *et al.* (2005, p. 124). Such complicated analyses cannot be motivated language-internally and hence are not acquirable from input alone. Since there is no theory-external evidence for such projections, theories that can do without such projections and without stipulations about UG should be preferred.

However, this does not mean that the search for universals or for similarities between languages and language classes is fundamentally misguided, although it may be possible that there is very little that is truly universal (Evans and Levinson 2009):¹¹ In principle, infinitely many descriptions of a particular language exist. It is possible to write a grammar that is descriptively adequate, but the way the grammar is written does not extend to other languages. So even without making broad claims about all languages, it is useful to look at several languages, and the more they differ from each other, the better. What we try to do in the CoreGram project is the modest version of mainstream generative grammar: We start with grammars of individual languages and generalize from there. We think that the framework we are using is well-suited for capturing generalizations within a language and across languages, since inheritance hierarchies are ideal tools for this. Note though that inheritance hierarchies are not the only place in the theory where generalizations can be captured. This is discussed in more detail

¹¹ But see (Harbour 2011) and the responses of authors in the same volume as Evans and Levinson's contribution for criticism of this paper.

in Sections 2.2 and 5.1.2 below. Of course, when building grammars, we can rely on several decades of research in theoretical linguistics and build on the insights that were gained by researchers working under UG-oriented assumptions. Without a theory-driven, comparative perspective on language, certain questions would never have been asked, and it is good that we have such valuable resources at hand. We nevertheless see some developments rather critically, as should be clear from the statements I have made above.

2.2 Data-driven, bottom-up theory development

Instead of imposing constraints from one language onto other languages, a bottom-up approach seems to be more appropriate: Grammars for individual languages should be motivated language-internally. Grammars that share certain properties can be grouped in classes. This makes it possible to capture generalizations about groups of languages and language as such. Let us consider some examples: German, Dutch, Danish, English and French. If we start developing grammars for German and Dutch, we find that they share a lot of properties: both are SOV and V2 languages, both have a verbal complex. One main difference is the order of elements in the verbal complex. The situation can be depicted as in Figure 3. There are some properties that are shared between German and Dutch (Set 3). For instance, the argument structure, a list containing descriptions of syntactic and semantic properties of arguments, and the linking of these arguments to the meaning of the sign is contained in Set 3. In addition, the constraints for SOV languages, the verb position in V2 clauses, and the fronting of a constituent in V2 clauses are contained in Set 3. The respective constraints are shared between the two grammars. When we

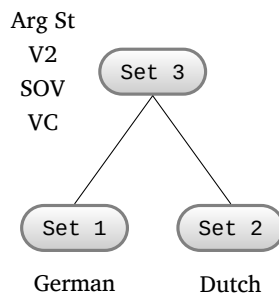
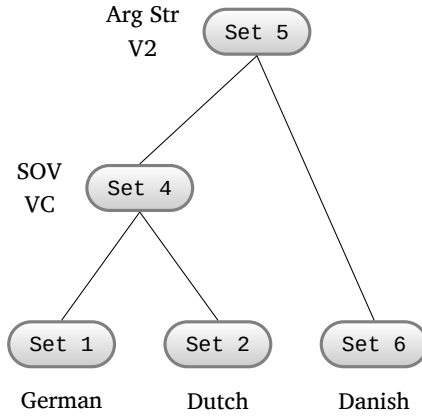


Figure 3:
Shared
properties
of German
and Dutch

Figure 4:
Shared
Properties of
German, Dutch,
and Danish



add another language, say Danish, we get further differences. While German and Dutch are SOV, Danish is an SVO language.

Figure 4 shows the resulting situation: The top-most node represents constraints that hold for all the languages considered so far (for instance, the argument structure constraints, linking, and V2), and the node below it (Set 4) contains constraints that hold for German and Dutch only. For instance, Set 4 contains constraints regarding verbal complexes and SOV order. In principle, there could be constraints that hold for Dutch and Danish but not for German, and for German and Danish but not for Dutch. These constraints would be removed from Set 1 and Set 2 respectively and put into another constraint set higher up in the hierarchy. For clarity, these sets are not illustrated in the figure, and I keep the names Set 1 and Set 2 from Figure 3 for the constraint sets for German and Dutch. The union of Set 4 and Set 5 is Set 3 of Figure 3.

If we add further languages, further constraint sets will be distinguished. Figure 5 on the facing page shows the situation that results when we add English and French. Again, the picture is not complete since there are constraints that are shared by Danish and English but not by French, but the general idea should be clear: By systematically working this way, we should arrive at constraint sets that directly correspond to those that have been established in the typological literature.

It should be clear from what has been said so far that the goal of every scientist who works this way is to find generalizations and

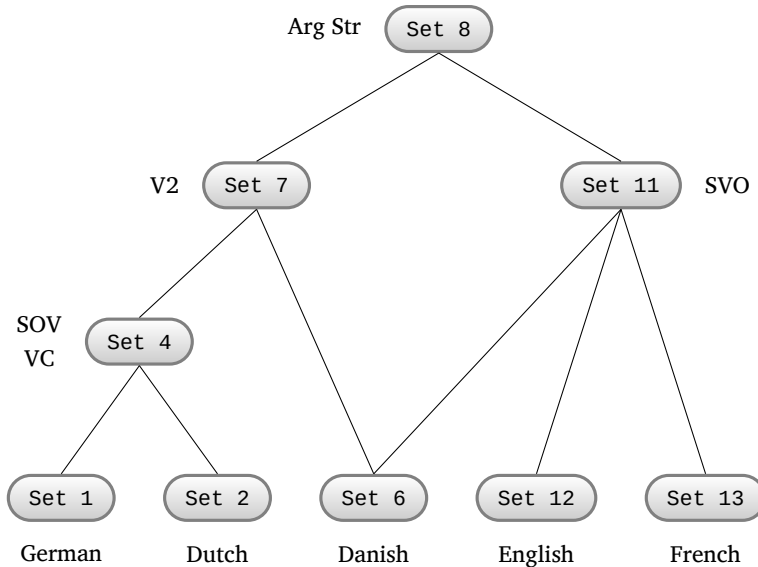


Figure 5:
Languages and
language classes

to describe a new language in a way that reuses theoretical constructs that have been found useful for a language already covered. However, as was explained above, the resulting grammars should be motivated by data of the respective languages and not by facts from other languages. In situations where more than one analysis would be compatible with a given dataset for language X, the evidence from language Y with similar constructs is most welcome and can be used as evidence in favor of one of the two analyses for language X. I call this approach the *bottom-up approach with cheating*: Unless there is contradicting evidence, we can reuse analyses that have been developed for other languages.

Note that this approach is compatible with the rather agnostic view advocated by Haspelmath (2010a), Dryer (1997), Croft (2001, Section 1.4.2–1.4.3), and others, who argue that descriptive categories should be language-specific, that is, the notion of *subject* for Tagalog is different from the one for English, the category *noun* in English is different from the category *noun* in Persian, and so on. Even if one follows such extreme positions, one can still derive generalizations regarding constituent structure, head-argument relations and so on. However, I believe that some categories can fruitfully be used cross-linguistically – if not universally, then at least for language

classes. As Newmeyer (2010, p.692) notes with regard to the notion of *subject*: Calling two items *subject* in one language does not entail that they have identical properties. The same is true for two linguistic items from different languages: calling a Persian linguistic item *subject* does not entail that it has exactly the same properties as an English linguistic object that is called *subject*. The same is, of course, true for all other categories and relations, for instance, parts of speech: Persian nouns do not share all properties with English nouns.¹² Haspelmath (2010b, p.697) writes: *Generative linguists try to use as many crosslinguistic categories in the description of individual languages as possible, and this often leads to insurmountable problems.* If the assumption of a category results in problems, they have to be solved. If this is not possible with the given set of categories or features, new ones have to be assumed. This is not a drawback of the methodology, quite the opposite: If we have found something that does not integrate nicely into what we already have, this is a sign that we have discovered something new and exciting. If we stick to language-particular categories and features, it is much harder to notice that a special phenomenon is involved since all categories and features are specific to one language anyway. Note also that not all speakers of a language community have exactly the same categories. If one were to take the idea of language-particular category symbols to an extreme, one would end up with person-specific category symbols like *Kim-English-noun*.

¹²Note that using labels like *Persian Noun* and *English Noun* is somewhat strange since it implies that both Persian nouns and English nouns are nouns in some way (see, for instance, Haspelmath 2010a, Section 2, for such a suggestion regarding case, e. g. Russian Dative, Korean Dative, ...). Instead of using the category *Persian Noun* one could assign objects of the respective class to the class *noun* and add a feature LANGUAGE with the value *persian*. This simple trick makes it possible to assign both objects of the type *Persian Noun* and objects of the type *English Noun* to the class *noun* and still maintain the fact that there are differences. Of course, no theoretical linguist would introduce the LANGUAGE feature to differentiate between Persian and English nouns, but nouns in the respective languages have other features that make them differ. So, the part-of-speech classification as noun is a generalization over nouns in various languages and the categories *Persian Noun* and *English Noun* are feature bundles that contain further, language-specific information.

After a talk I gave at the MIT in 2013, members of the linguistics department objected to the approach taken in the CoreGram project and claimed that it would not make any predictions as far as possible or impossible languages are concerned. Regarding predictions, two things must be said: Firstly, predictions are being made on a language-particular basis. As an example, consider the following sentences from (Netter 1991):

- (5) a. [Versucht, zu lesen], hat er das Buch nicht.
tried to read has he the book not
'He did not try to read the book.'
- b. [Versucht, einen Freund vorzustellen], hat er ihr noch
tried a friend to.introduce has he her yet
nie.
never
'He never tried to introduce her to a friend.'

When I first read these sentences, I had no idea about their structure. I typed them into my computer, and within milliseconds, got a syntactic analysis. When I studied the results, I realized that these sentences are combinations of partial verb phrase fronting and the so-called third construction (Müller 1999, p. 439). I had previously implemented analyses of both phenomena, but had never thought about the interaction of the two. The grammar predicted that examples like (5) are grammatical. Similarly, the constraints of the grammar can interact to rule out certain structures. So, predictions about ungrammaticality or impossible structures are in fact made as well.

Secondly, the top-most constraint set holds for all languages seen so far. It can be regarded as a hypothesis about properties that are shared by all languages. This set contains constraints for the connection between syntax and information structure, and such constraints allow for V2 languages but rule out languages with the verb in penultimate position. (See Kayne 1994, p. 50, for the claim that such languages do not exist. Kayne develops a complicated syntactic system that predicts this.) Of course, if a language is found that puts the verb in penultimate position for the encoding of sentence types or some other communicative effect, a more general top-most set has to be defined, but this is parallel for Minimalist theories: If languages are found that are incompatible with basic assumptions, the basic assumptions

have to be revised. As with the language particular constraints, the constraints from the top-most set make certain predictions about the phenomena that can and cannot be found in languages.

Cinque (1999, p. 106) suggested a cascade of functional projections to account for reoccurring orderings in the languages of the world. He assumes elaborate tree structures to play a role in the analysis of all sentences in all languages, even if there is no evidence for respective morphosyntactic distinctions in a particular language (see also Cinque and Rizzi 2010, p. 55). In the latter case, Cinque assumes that the respective tree nodes are empty. Cinque's results could be incorporated in the model advocated here. We would define part-of-speech categories and morpho-syntactic features in the top-most set, and state linearization constraints that enforce the order that Cinque encoded directly in his tree structure. In languages in which such categories are not manifested by lexical material, the constraints would never apply. Neither empty elements nor elaborate tree structures would be needed. So, Cinque's data could be covered in a better way in an HPSG with a rich UG. However, we refrain from assuming a rich UG and introducing 400 categories (or features) into the theories of all languages. Again, I would like to point out that this would be implausible from a genetic point of view, and I wait for other, probably functional, explanations of the Cinque data.

Frequently discussed examples, such as those languages that form questions by reversing the order of the words in a string (Musso *et al.* 2003), need not be ruled out in the grammar since they are ruled out by language external constraints: We simply do not have enough working memory to do such complex computations.

After having justified the basic approach taken in the CoreGram project, I now turn to the coverage of the grammars and discuss some highlights.

3

COVERAGE AND HIGHLIGHTS

The computer-processable grammar fragments of German, Persian, and Danish are relatively big.¹³ The German grammar (BerliGram)

¹³A list of covered phenomena accompanied by appropriate grammatical and ungrammatical test strings is part of the distributions of the grammars and

was the first one to be implemented. It is an extension of the grammars that were developed for the individual chapters of the HPSG textbook (Müller 2007b). The textbook covers noun phrases with adjuncts (APs, PPs, relative clauses), determinerless NPs, constituent order (scrambling, verb position and clause types, nonlocal dependencies for fronting), agreement (subject–verb and NP internal), predicate complex formation (verbal complexes and adjective–verb complexes, Oberfeldumstellung, partial fronting), control and raising, case assignment and passive (personal, impersonal, remote passive, attributive participles, *lassen* passive, *bekommen* passive, state passive, modal infinitives), particle verbs (productive and lexicalized), morphology (inflectional and derivational, for instance *-bar* ‘able’ derivation), and (symmetric) coordination. The Situation Semantics that is used in the textbook was replaced by a Minimal Recursion Semantics (MRS, Copestake *et al.* 2005). MRS allows for underspecification of scope, so that a sentence like (6) gets one representation from which the several scopings can be derived. See (Dowty 1979, Section 5.6) for the discussion of the readings of *again* in English, and (Egg 1999) for the explanation of the different readings of (6).

- (6) a. dass Max wieder alle Fenster öffnete
that Max again all windows opened
‘that Max opened all windows again’
b. *again'*(\forall (CAUSE(*open'*))); repetitive
c. *again'*(CAUSE(\forall (*open'*))); repetitive
d. CAUSE(*again'*(\forall (*open'*))); restitutive

Von Stechow (1996, p. 93) develops an analysis in the framework of Minimalism that assumes an empty VOICE head that contributes the CAUSE relation and some further functional heads for AgrO and AgrS. The empty heads were used to derive several readings in a movement-based analysis. However, as Jäger and Blutner (2003) pointed out, von Stechow’s analysis cannot derive all readings. We therefore follow Egg (1999) and treat (6a) as an instance of sublexical scoping: *öffnen* is lexically decomposed into CAUSE(*open'*), and the *again* can scope below the CAUSE operator although there is no decomposition in syntax. The

can be downloaded at the respective web pages: <http://hpsg.fu-berlin.de/Projects/CoreGram.html>.

scope relations are represented in dominance graphs. Egg's analysis has been translated into MRS (see Müller 2010a, Section 19.9.2, and Müller 2015c, Section 3). Since there is no decomposition in syntax, our analysis avoids empty elements: It is just the words of (6a) that are combined in an analysis, and only these words contribute to the interpretation.

In addition to the modification of the semantics component, some further special phenomena have been implemented. For instance, an analysis of multiple frontings (Müller 2003a), something that is unique among existing HPSG implementations. For a discussion of approaches to constituent order that are incompatible with the multiple frontings data, see (Müller 2005, 2015a). Furthermore, analyses of depictives (Müller 2008), left dislocation, copula constructions (Müller 2012), and positional expletives (Müller and Ørsnes 2011) were added. Some phenomena that have been covered in my earlier grammars of German have not yet been transferred to BerliGram.

The Danish grammar is documented in a book of more than 500 pages, which is not complete yet. The grammar covers the NP (definite marking by suffix, bare plurals, adjuncts), verb position (SVO, but verb inversion in V2 sentences), negation preposing, questions, predicational constructions, specificational constructions, agreement, coordination, case assignment, passive, perfect, adverbial phrases, embedded interrogative clauses, object shift and negation shift, partial fronting, raising and control, passive (personal and impersonal constructions) and complex passive, adjectival passives, preposition stranding, and modal verbs. The following examples show in a compact way the interaction of several phenomena: passive with promotion of either the direct object or the indirect object (7a,c), passive and pronoun shift (7b,d), and partial fronting and object shift (7b,d):

- (7) a. Bjarne bliver ikke anbefalet den.
 Bjarne.NOM is not recommended it.ACC
 'It is not recommended to Bjarne.' (lit: 'Bjarne is not
 recommended it.')
- b. ? Anbefalet bliver Bjarne den ikke.
 recommended is Bjarne.NOM it.ACC not
 'It is not recommended to Bjarne.'

- c. Bogen bliver ikke anbefalet ham.
book.DEF.NOM is not recommended him.ACC
'The book is not recommended to him.'
- d. ? Anbefalet bliver bogen ham ikke.
recommended is book.DEF.NOM him.ACC not
'The book is not recommended to him.'

The examples in (7b,d) are interesting since Danish differs from German and Dutch in not allowing incomplete category fronting in general. Such partial frontings can only be found if the missing components are shifted pronouns, that is, pronouns to the left of the negation (Holmberg 1999). Due to the complexity of the construction, examples are marked, but Müller and Ørsnes (2013b) and Müller and Ørsnes (2015) provide attested data.

The Mandarin Chinese grammar was implemented with the help of Jia Zhongheng. We used the description in (Li and Thompson 1981) as the basis for our implementation. Currently, we cover the NP (classifiers, determiners, attributive phrases with adjectives and relative clauses), location words and localizer phrases, basic clause structure, passive (*bei* construction), the *ba* construction, adverbials (PPs, adverbs), negation, auxiliaries, aspect marking, reduplication, presentational constructions, and serial verb constructions. Among the things that are special are NPs that contain classifiers, as in (8), and change of part of speech by reduplication, as in (9).

- (8) 那 辆 红 的 车 锈了。
na4 liang4 hong2 de che1 xiu4.le
that CL red DE car rust.ASP
'That red car rusts.'

The adjective 高兴 (gao1xing4, 'happy') in (9a) is converted into an adverb by forming the pattern AABB from the original adjective AB, that is, both gao1 and xing4 are doubled.

- (9) a. 他 很 高兴。
ta1 hen gao1xing4
he very happy
'he is very happy'

- b. 他 高高兴兴 游泳。
tal gao1gao1xing4xing4 you3yong3
he AABB = happily swims
'He swims happily.'

The Persian grammar is a larger fragment, which still needs to be fully documented (Müller *et al.* In Preparation). A description of some parts of the grammar can be found in (Müller and Ghayoomi 2010). The grammar covers various types of light verb constructions, which are crucial for the analysis of Persian, since Persian has only very few verbs. The light verb constructions interact with other constructions like negation, all tenses (periphrastic and synthetic), and cliticization. All of these interactions are covered. Furthermore, the grammar contains analyses of passive, adjectival passives, the NP structure (*ezafe* construction, possessives, adjectives, ...), direct object marking, agreement, pro-drop, non-local dependencies including those with resumptive pronouns, inflectional and derivational morphology, coordination, relative clauses including free relative clauses, and questions.

The grammar can be used with Persian script or with a romanized version that is usually used in linguistic texts. The examples in (10) show light verb constructions, which are an important feature of the language. (10a) shows that the future auxiliary can interrupt the preverb–verb sequence of light verbs. (10b) shows an example with the negation prefix in the middle of the light verb construction and pro-drop.

- (10) a. من این کار را انجام خواهم داد.
man in kār rā anjām xāh-am dād.¹⁴
I this job DOM performance will-1SG gave
'I will do this work.'
- b. مرد را دوست نداشت.
mard rā dust na-dāšt.
man DOM friend NEG-had
'He/she did not love the man.'

The Maltese grammar is an implementation of the description by Fabri (1993). Fabri works in the framework of Lexical Decomposition Grammar, which is also a lexical framework, and his analyses are

¹⁴ Example taken from (Karimi-Doostan 1997, p. 73).

translatable into HPSG without great effort. The grammar covers basic sentence structure, pro-drop, clitics (with correct spelling and modeling of the morphophonological changes, see Section 5.1.5), adjectival predication (without copula), agreement, NP structure (including adjective order, which depends on the class of adjective), definiteness marking, and case assignment. The examples in (11) show definiteness marking. (11b) shows assimilation and (11c) shows clitics:¹⁵

- (11) a. Il-komunist xejjer lil-l-papa.
DEF-communist waves.3M.SG Case-DEF-pope.M.SG
'The communist waves at the pope.'
- b. It-terrorist bagħat l-ittr-a lil-l-president.
DEF-terrorist sent DEF-letter-F Case-DEF-president
'The terrorist sent the president the letter.'
- c. It-terrorist bagħat = hie = l.
DEF-terrorist sent.3M.SG = 3F.SG = 3M.SG
'The terrorist sent it to him.'

(11c) is ambiguous, as there is a reading with clitic left dislocation. Both readings are accommodated by the grammar.

The grammars of Yiddish, English, and Hindi are small-scale. They cover the basic structures of these languages and were implemented in connection with work comparing several languages. For instance, Yiddish covers V2 in main and embedded clauses, positional expletives (Müller and Ørsnes 2011), and embedded interrogative clauses. English covers NP structures, the main clause structure, case assignment, agreement, auxiliary verb constructions, negation, coordination, and inflectional and derivational morphology (*-able* derivation).

Among the basic clause structures, the grammar of Hindi covers case assignment, agreement, verbal complex formation, nonlocal dependencies, adjuncts, information structure markings, and inflectional morphology.

¹⁵The examples are taken from (Fabri 1993, p. 130).

In the CoreGram project, we assume that valence is represented in a uniform way across languages.¹⁶ Arguments of a head are represented in the ARG-ST list (Pollard and Sag 1994, Chapter 9). They are mapped to the valence features SPR and COMPS in a language-dependent fashion. For instance, English and Danish map the subject to the SPR list and the other arguments to COMPS. Danish inserts an expletive in cases in which there is no element that can be mapped to SPR, while English does not do this (Müller and Ørsnes 2013a). German differs from both languages in mapping all arguments of finite verbs to the COMPS list (Pollard 1996).

The arguments in the ARG-ST list are ordered according to the obliqueness hierarchy of Keenan and Comrie (1977), which plays a role in the analysis of a variety of phenomena (for instance case assignment and depictive predicates). The elements of the ARG-ST list are linked to the semantic roles that a certain head has to fill. Since the traditional role labels like agent and patient are problematic, the CoreGram grammars adopt Dowty's proto-role approach (1991). ARG1, ARG2, and so on are used as role labels.

Originally, HPSG came with very few immediate dominance schemata: Head-Complement Schema, Head-Specifier Schema, Head-Adjunct Schema, the Head-Filler Schema for binding off unbounded dependencies, and the Head-Extra Schema for binding off clause-bound nonlocal dependencies. Since (Sag 1997), many HPSG analyses have a more constructional flavor, that is, specific subconstructions of these general schemata are introduced (Sag 2010). In the CoreGram project we stay within the old tradition of HPSG and continue to use the rather abstract dominance schemata. However, it is possible to state further constraints on the respective structures. So, rather than having several very specific instances of the Head-Filler Schema, we have

¹⁶ Koenig and Michelson (2012) argue for an analysis of Oneida (a Northern Iroquoian language) that does not include a representation of syntactic valence. If this analysis is correct, syntactic argument structure would not be universal, but would be characteristic for a large number of languages.

very few (for instance, for verb-second clauses and relative clauses) and formulate additional implicational constraints that constrain actual instances of head-filler phrases further if the antecedent of the implicational constraint is true. An example of such a constraint is the following one, which was suggested by Bildhauer and Cook (2010, p. 75):

$$(12) \left[\begin{array}{l} \text{NON-HEAD-DTRS } \langle [\text{HEAD} | \text{DSL } local] \rangle \\ \text{head-filler-phrase} \end{array} \right] \Rightarrow \left[\text{IS } pres \vee a\text{-top-com} \vee \dots \right]$$

The constraint says that, for all head-filler phrases that have a non-head daughter whose DSL value is of type *local*, the value of the information structure feature IS has to be *pres* \vee *a-top-com* \vee ¹⁷

Since the schemata are rather general, they can be used for all languages under consideration so far. Of course, the languages differ in terms of constituent order, but this can be dealt with by using linearization rules that are sensitive to features whose values are language specific. For instance, all heads have a feature INITIAL. The value is ‘+’, if the head has to be serialized before its complements, and ‘-’ if it follows its complements. German and Persian verbs are INITIAL -, while English, Danish, Mandarin Chinese and Maltese verbs are INITIAL +.

We assume binary branching structures, and hence we get the structures in (13) for English and the corresponding German example:

- (13) a. He [[gave the woman] a book].
 b. er [der Frau [ein Buch gab]]
 he the woman a book gave

The LP rules enforce that *gave* is linearized before *the woman* and *gave the woman* is linearized before *a book*.

The scrambling of arguments is accounted for by ID schemata that allow the combination of a head with any of its arguments independently of the position an element has in the valence list of its head. Similar analyses have been suggested in the framework of HPSG by

¹⁷ *pres* is an abbreviation for *presentational* and *a-top-com* abbreviates *assessed-topic-comment*. For further details see (Bildhauer and Cook 2010).

Gunji (1986) for Japanese and Pollard (1996) for German. Many authors assume a valence set rather than a list. However, the order of the elements has to be represented somewhere in the grammar, since it is relevant for various phenomena. The proposal adopted in the Core-Gram project assumes an ordered list but allows the saturation in an arbitrary order. For non-HPSG analyses that are similar to the set-based approaches, see (Fanselow 2001) and (Steedman and Baldrige 2006).

Non-scrambling languages like English combine heads with their complements in a strict order: The least oblique element is combined with the head first and then the more oblique complements follow. Non-scrambling languages with head-final order take the last element from the valence list first. Again, see (Steedman and Baldrige 2006) for a similar proposal in the framework of Categorical Grammar.

4.3 *Morphology and lexical rules*

We follow a lexical rule-based approach to morphology. Lexical rules are basically unary branching trees that license new lexical items (Briscoe and Copestake 1999; Meurers 2001).¹⁸ A lexical rule can add to or subtract from the phonology (or, in implementations, the orthography) of an input item. For instance, it is possible to analyze the complex morphological patterns that we observe in Semitic languages by mapping a root consisting of consonants to a full-fledged stem or word that has the appropriate vowels inserted. We follow Bresnan and Mchombo (1995) in assuming the Lexical Integrity Principle. This means that all morphological combinations have to be done by lexical rules, that is, fully inflected forms are part of the lexicon, most of them being licensed by productive lexical rules.

Lexical rules do not have to change the phonology or orthography of the item they apply to. For instance, lexical rules can be used to license further lexical items with extended or reduced valence requirements. As was argued in (Müller 2002, 2006) resultative constructions should be treated lexically. So, there is a lexical rule that maps the stem *fisch-* of the intransitive version of the verb *fischen* ('to fish') onto

¹⁸Goldberg (2013) calls such lexical rules *lexical templates* and sets them apart from lexical rules that relate stored lexical items.

a stem *fisch-* that selects for a secondary predicate (adjective or PP) and the subject of this predicate as well.

- (14) Er fischt den Teich leer.
 he fishes the pond empty

4.4 Semantics

All grammars come with a semantics component. We use Minimal Recursion Semantics (Copestake *et al.* 2005), since it allows for the underspecification of scope (see Section 3). For instance, the sentence in (15) has two readings: one in which the existential quantifier outscopes the universal quantifier and one in which the scopings are reversed.

- (15) a. Every dog chased some cat.
 b. $\forall x(\text{dog}(x) \rightarrow \exists y(\text{cat}(y) \wedge \text{chase}(x, y)))$
 c. $\exists y(\text{cat}(y) \wedge \forall x(\text{dog}(x) \rightarrow \text{chase}(x, y)))$

These readings can be represented compactly in an underspecified way as in (16):

- (16) $\langle h_0, \{$
 $h_1:\text{every}(x, h_2, h_3), h_4:\text{dog}(x), h_5:\text{chase}(e, x, y),$
 $h_6:\text{some}(y, h_7, h_8), h_9:\text{cat}(y) \}, \{ h_2 =_q h_4, h_7 =_q h_9 \} \rangle$

Every word that contributes semantically has a referential index (x, y in (16)) or event variable (e in (16)) and a list of relations that contains elementary predications. Elementary predications come with a so-called handle ($h_1, h_4, h_5, h_6,$ and h_9 in (16)) that can be used to embed the respective elementary predicate under another one. Quantifiers are represented as three-place predicates. They have one slot for the variable they bind and two further slots for their scope and their restriction (for instance h_7 and h_8 in (16)). In addition, it is possible to specify scope constraints that say which elementary predication has to be outscoped by a certain quantifier (for instance, (16) says that the h_2 argument of *every* has to outscope h_4 , which is the handle of *dog*).

The MRS in (16) can best be depicted as in Figure 6. h_0 stands for the top element. This is a handle that dominates all other handles in a dominance graph. The restriction of *every* dominates *dog* and the restriction of *some* dominates *cat*. The bodies of both quantifiers dominate *chase*. The interesting thing is that exact dominance relations are

Figure 6:
Dominance
graph for *Every
dog chases some
cat*.

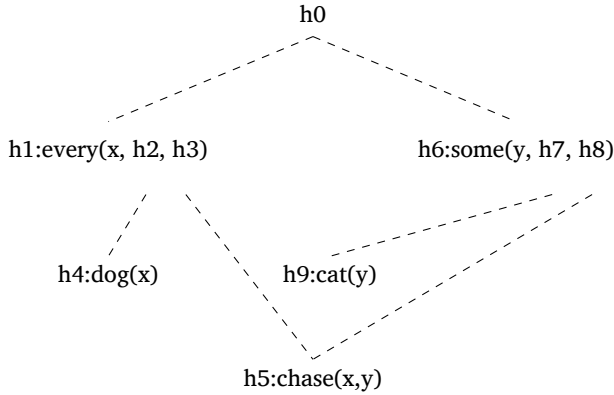
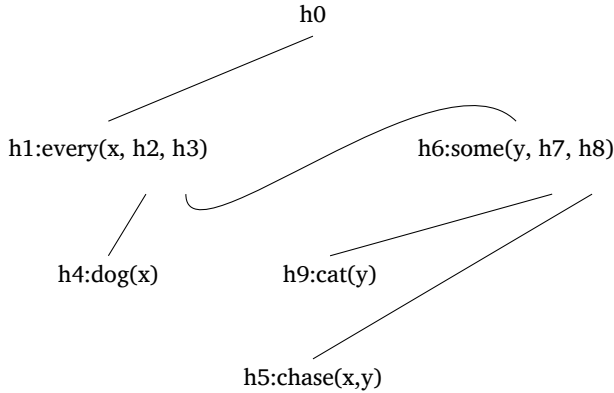


Figure 7:
 $\text{every}(x, \text{dog}(x),$
 $\text{some}(y, \text{cat}(y),$
 $\text{chase}(x, y))) \equiv$
(15b)



not fixed, which is indicated by the dashed lines in Figure 6. There are two ways to plug an elementary predication into the open slots of the quantifiers:

- (17) a. Solution one: $h_0 = h_1, h_2 = h_4, h_3 = h_5, h_7 = h_9,$ and $h_8 = h_5.$
(*every dog* has wide scope)
- b. Solution two: $h_0 = h_6, h_7 = h_9, h_8 = h_1, h_2 = h_4,$ and $h_3 = h_5.$
(*some cat* has wide scope)

The solutions are depicted as Figure 7 and Figure 8.

When several linguistic objects are combined, all the elementary predications and the scope constraints are collected at mother nodes.

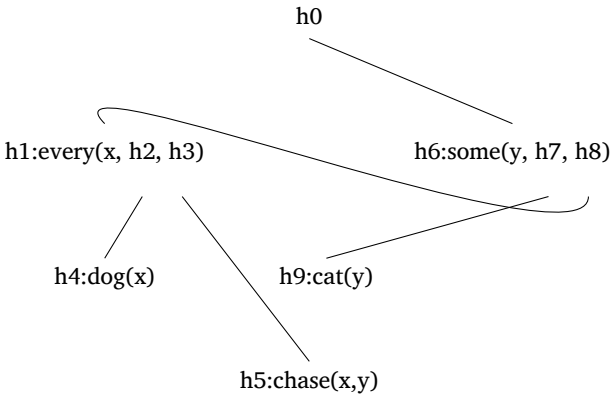


Figure 8:
 $\text{some}(y, \text{cat}(y),$
 $\text{every}(x, \text{dog}(x),$
 $\text{chase}(x, y))) \equiv$
 (15c)

This corresponds to traditional compositional semantics. However, it is also possible to add additional relations and scope constraints in the computation of the meaning of a certain combination. This makes it possible to capture the insight from Construction Grammar that sometimes the meaning of a complex combination contains more than is present in the individual components of the complex object.

See (Copestake *et al.* 2005) for a full introduction to MRS, and (Müller 2015c) for a brief one.

4.5 Information structure

The German grammar contains constraints on information structure. The corresponding theory was developed by Felix Bildhauer and Philippa Cook and implemented by Felix Bildhauer in Project A6 of the collaborative research centre SFB 632 on information structure (Bildhauer and Cook 2010; Müller *et al.* 2012). See also Example (12) above. Project A6 explored the various contexts for so-called multiple frontings. It implemented an analysis that refers to syntactic configurations and assigns the elementary predications from an MRS representation to TOPIC and FOCUS lists. See (Engdahl and Vallduví 1996) for the general approach.

Elodie Winckel is currently augmenting the French grammar with an information-structure component. This work is also carried out in the context of SFB 632, and one goal of this work is to test to what extent it is possible to explain island constraints with respect to information structure (Ambridge and Goldberg 2008).

Bildhauer's (2008) Spanish grammar also implements a theory of information structure, and as was mentioned above, this grammar is currently ported to the CoreGram format.

5 IMPLEMENTATION DETAILS

5.1 TRALE

The grammars are implemented in TRALE (Meurers *et al.* 2002; Penn 2004). TRALE implements typed feature descriptions. Every grammar consists of a signature (a type hierarchy with feature introduction and appropriateness constraints) and a theory that states constraints on objects of these types. TRALE is implemented in Prolog and comes with an implementation of relational constraints that maps the TRALE relations to Prolog relations. TRALE has two parsers: a standard bottom-up chart parser and a linearization parser (Suhre 1999). The CoreGram project uses the standard bottom-up parser. Both parsers use a phrase structure backbone. TRALE is available as bootable CD-ROM (Müller 2007a). The CD-ROM contains a full installation of all components that were available in 2007. This includes a chart display for developing and debugging grammars, Utool for visualizing dominance graphs from semantic representations and scope resolving (Koller and Thater 2005), and [incr tsdb()] for systematic testing (see Section 7). We hope to finish a new virtual machine soon that includes a new and much faster version of TRALE, the most recent versions of the CoreGram grammars, and Kahina, a powerful debugger for constraint resolution systems like TRALE (Dellert *et al.* 2010, 2013). A beta version is already available.

Compared to other systems such as LKB (Copestake 2002), the expressive power of the description language is high (see also Melnik 2007). This allows for the rather direct implementation of analyses that are proposed by theoretical linguists. The following descriptive devices are used in the theory and are provided by TRALE; the references point to papers which argue for such constructs:

- empty elements (Kiss 1995; Meurers 1999a; Levine and Hukari 2006; Bender 2000; Sag *et al.* 2003, p. 464; Borsley 2004, Section 3.3; Müller 2007b, 2014b,d; Haugereid *et al.* 2013)

- relational constraints (Pollard and Sag 1994; Bouma *et al.* 2001; Meurers *et al.* 2003),
- complex antecedents in implicational constraints (Bonami *et al.* 1998; Meurers 2000, p. 207; Bonami and Godard 2001, p. 148; Koenig and Davis 2004, p. 145, 149; Müller 2007b, p. 145, Section 10.3; 2014d; Bildhauer and Cook 2010, p. 75; Van Eynde and Augustinus 2014, p. 166; Alotaibi and Borsley 2013, p. 18),
- cyclic structures (Engdahl and Vallduví 1994, p. 56; Meurers, 2000, p. 2007; 2001, p. 176, Samvelian 2007, p. 638),
- macros, and
- a morphology component that has the expressive power needed to account for nontrivial morphological phenomena.

5.1.1 Empty elements

All CoreGram grammars use empty elements to account for extraction phenomena. Auxiliary inversion in English has not yet been implemented, but German and Danish use head-movement analyses to account for the verb in initial position in questions and V2 clauses. TRALE has mechanisms to precompile grammars and to eliminate most of the empty elements (for a discussion, see Müller 2014b).

5.1.2 Complex antecedents

To see how useful implicational constraints with complex antecedents are both from a theoretical and an implementational perspective, consider the constraint in (12) again. Proposals that do not make use of such implementational constraints would have to introduce two subtypes of *head-filler-phrase*: one for head-filler phrases with the DSL value *local* – let us call this type *head-filler-phrase-dsl* – and one for head-filler phrases with the DSL value different from *local*. The information structure constraints from (12) will be constraints on structures of type *head-filler-phrase-dsl*:

$$(18) \text{ head-filler-phrase-dsl} \Rightarrow \left[\text{NON-HEAD-DTRS} \left\langle \left[\text{HEAD|DSL } local \right] \right\rangle \right] \\ \left[\text{IS } pres \vee a\text{-top-com} \vee \dots \right]$$

Proponents of such a theory would basically make explicit which daughters could appear in the filler position.

In our setting with implicational constraints, we do not need these two additional types. We formulate the constraint in (12), and this constraint applies only to those head-filler phrases that have a non-head daughter with a DSL value of type *local*. Therefore, our theory is simpler and has to be preferred over other approaches that duplicate information about the combinatorics of linguistic objects in type names.

5.1.3 Relational constraints

The relational constraint that is used most often in HPSG is *append* (\oplus), which concatenates two lists. While many of the uses of *append* can be recoded using difference lists, this is not always the case. See (Meurers *et al.* 2003) for some discussion. In the implementation of scrambling that was sketched in Section 4.2 above, a valence list is split into three parts. The first is a list of arbitrary length, the second is a list containing the element that has to be combined with the head, and the third is a list of arbitrary length again. This can be implemented directly using *append*: $A \oplus \langle XP \rangle \oplus B$.

Another application of relational constraints is the determination of the last element of a list. For technical reasons the argument structure and valence lists are represented with the most oblique element at the beginning (as in Pollard and Sag 1987). If one wants to access the least oblique element in the ARG-ST list, one has to find the last element of this list. In the theory of Heinz and Matiasek (1994), which follows Haider (1986), transitive and unergative verbs have a designated argument that is identical to the least oblique argument of the verb. (19) shows how this can be expressed in TRALE:

- (19) (synsem:loc:cat:(head:da:[last(ArgSt)],
arg_st:ArgSt))

ArgSt is the value of the ARG-ST feature, and last(ArgSt) returns the last element of this list, which is represented as the element of the list which is the value of the feature DA. Note that this works for lists of arbitrary length. This is important for verbs like the German *lassen* ('to let') that raise the arguments of the verbal element that they embed. TRALE uses a delay mechanism to postpone the execution of relational constraints until enough information is available. In the case of *lassen*,

the constraints are delayed until it is combined with the embedded verb and the actual length of its argument structure list is known.

5.1.4 Macros

Just like types, macros can be organized in hierarchies. The type hierarchies are stated in a signature, but the macro hierarchies are constructed by calling other macros in a macro definition. Macros differ from types in allowing parameters. This makes it possible to represent the lexicon in a rather compact way. For instance, (20a) shows the lexical item for *work*, and (20b) the definition of the macro that is called.

- (20) a. `work ---> @intrans_verb(a_ work, agentive).`
b. `intrans_verb(Relation, Sort) :=
 (@strict_intrans_nerg_verb,
 rels:[(pred:Relation,
 arg1:sort:Sort)]).`

5.1.5 Lexical rules and morphology

TRALE uses a special syntax for lexical rules: an identifier is followed by the keyword ‘lex_rule’, by the input description, the arrow ‘**>’ and the output description. Since we want to be as close as possible to HPSG analyses, we assume that every lexical rule has a specific type (*definiteness_lr* in (24) below). The respective typed feature structure models a linguistic object with a daughters list. The daughter is the input of the lexical rule. In addition to this, a lexical rule has to have a *morphs* statement that says something about how the orthography of the input is related to the orthography of the output. (24) shows the lexical rule that is used to account for definiteness marking in Maltese. Definiteness is marked with an /l/ at nouns and adjectives in Maltese. (21) gives an example:¹⁹

- (21) `l-ktieb`
`DEF-book`

If the noun starts with one of the coronals /d/, /t/, /s/, /z/, /ʃ/, /ts/, /tʃ/, /n/, or /r/, the /l/ is assimilated. (22) gives an example:

¹⁹(21), (22) and (23b) are underlying forms. If the definite form of *book* is used in isolation, an /i/ has to be added.

- (22) r-raġel
DEF-man

The only exception is the coronal /dʒ/, which is exempt from assimilation.

Inner epenthesis can be observed if the word starts with /s/ or /ʃ/ followed by a consonant:

- (23) a. skola
school
b. l-iskola
DEF-school

If inner epenthesis applies, it prevents assimilation.

The lexical rule splits the input characters into an initial part S and a part X and tests whether S is [s] or [x] by calling the predicate `s_sh`. If this call succeeds, *li* is appended to the output. If this call does not succeed, other clauses are tried. If the input starts with a coronal, the coronal is doubled. Otherwise the input is prefixed with an *l*.²⁰

- ```
(24) definiteness_lr lex_rule
 Dtr
 **>
 (definiteness_lr,
 dtrs:[Dtr])
 morphs
 (S,X) becomes (li,S,X) when s_sh(S), % l-iskola
 (C,X) becomes (C,C,X) when coronal(C), % r-raġel
 X becomes (l,X). % l-ktieb

 s_sh([s]).
 s_sh([x]).

 coronal([d]).
 coronal([t]).
```

---

<sup>20</sup> A reviewer asked whether CoreGram parses sounds or orthographic representations. The latter is the case, but since the Maltese spelling is close to the phonological representation in the relevant area, phonological concepts like coronal can be used in the rules that relate orthographic forms. `s_sh/1` and `coronal/1` are Prolog predicates.

```
coronal([s]).
coronal([z]).
coronal([n]).
coronal([r]).
```

## 5.2 *Setup of CoreGram*

The grammars are organized in one directory for every language. The respective directories contain a subdirectory named *Core-Grammar*. This directory contains files that are shared between the grammars. For instance, the file *core-macros.pl* contains macros that are or can be used by all languages. For every language, there is a load file to fetch the relevant files from the core grammar directory that. So, for instance *english.pl*, *french.pl*, and *danish.pl* all load *nom-acc.pl* since these languages are nominative–accusative languages. These files also contain code for loading macros and constraints for languages that do not form a verbal complex, while *german.pl* does load the files for cluster-forming languages. These files directly correspond to the constraint sets that were discussed in Section 2.

The option to specify type constraints makes it possible to state constraints that hold for a certain construction cross-linguistically in a file that is loaded by all grammars and restrict structures of this type further in language-particular files.

Lexical rules are also described by feature descriptions and organized in type hierarchies (Meurers 2001). Just like other constraints, the constraints on lexical rules can be shared.

## 6 COMPARISON TO OTHER MULTILINGUAL PROJECTS

Two other large groups are currently working in the area of multilingual grammar engineering. We will deal with both of them in the following subsections and will explain in what way the CoreGram project differs from them. The DELPH-IN consortium<sup>21</sup> will be described in Section 6.1, and ParGram in Section 6.2.

---

<sup>21</sup> DELPH-IN is an abbreviation for Deep Linguistic Processing with HPSG.

6.1 *DELPH-IN and the LinGO Grammar Matrix*

The DELPH-IN group uses the LKB system (Copestake 2002) for grammar development. The LinGO Grammar Matrix provides a collection of types for lexical objects and phrasal schemata that can be used by grammar writers (Bender *et al.* 2002; Bender and Flickinger 2005). The Matrix builds on experiences from the development of grammars for English, German, Japanese, and Spanish. The Grammar Matrix provides a starter set. This can be modified and extended by individual grammar writers without any interaction with any other grammars that were derived from the Matrix. Of course, there is a feedback loop: Grammar writers can inform the developers of the Grammar Matrix about their requirements and changes which they believe to be appropriate. In the CoreGram project all grammars use the same core files. If the grammar core is changed because of evidence in, say, Persian, all other grammars have to be compatible with the change or have to be adapted. While this increases the complexity of the development process considerably, the result is a consistent set of grammars with partly shared constraints for several typologically diverse languages.

Work that is done in the DELPH-IN consortium has a strong focus on applications. Efficient processing has a high priority. This, among other causes, led to a reduction of the expressive power of the description formalism and is also reflected in analyses. See, for instance, (Crysmann 2003) and the criticism in (Müller 2005, Section 3.6). In our project, we see processing issues as secondary and want to treat computationally expensive, but linguistically interesting phenomena as well as those that can be handled efficiently. Although the development of linguistically motivated analyses has the highest priority, processability is not ignored completely. Since profiling tools (chart display and the test suite tool [incr tsdb()], see Oepen and Carroll 2000) are integrated into TRALE, an exact examination of the grammars is possible, and unnecessary computations of the parser can be detected and eliminated.

6.2 *ParGram*

A similar community to the DELPH-IN consortium is working in the framework of LFG and is organized in the ParGram project (Butt *et al.* 1999, 2002). The goal of the project is the implementation of parallel

LFG grammars for a set of languages. Currently, there exist grammars for Arabic, Danish, English, Georgian, German, Hungarian, Japanese, Malagasy, Norwegian, Polish, Tigrinya, Turkish, Hungarian, Urdu, Vietnamese, Welsh, and Wolof (see Müller 2015b, Chapter 6, for an overview and references). These grammars are parallel in that they produce f-structures that have the same feature geometry and uniform analyses of the phenomena. Parallel grammar development is challenging for developers, since each phenomenon has to be examined carefully, and it has to be decided whether a cross-linguistic analysis is feasible at all, whether the phenomenon is idiosyncratic and language-specific, or whether the feature geometry has to be changed. The grammars are developed in the X(erox) L(inguistic) E(nvironment) system (Kaplan *et al.* 2002; Butt *et al.* 1999).

The ambitions of our project (and also the Grammar Matrix) are higher than those of the ParGram project, since HPSG grammars model the whole range of grammatical properties: Morphology, syntax, semantics, and information structure are described with the same feature geometry. Dominance schemata for head-argument phrases, head-adjunct phrases, filler-head phrases, specifier-head phrases, and so on are specified for all languages or for certain language classes. In comparison, computational LFG grammars differ enormously in their c-structures, while morphology is usually taken care of by external programs (Finite State Morphology) and is not part of theoretical considerations. Since, for instance, complex predicate formation in languages such as German and Persian interacts with derivational morphology (Müller 2003b, 2010b), we consider it crucial that morphology is dealt with within the grammatical framework, and that the computational implementation reflects the tight connection between the two parts of grammar.

Much to the frustration of many linguists, the contribution of certain theoretical approaches to progress in linguistics is rather unclear. Many proposals do not extend the amount of data that was already covered by analyses developed during the 1980s in the framework of GB and other, non-transformational, frameworks. In comparison, the methodology described in Section 2 leads to grammars with in-

creasing coverage and analyses that are improved by cross-linguistic considerations.

The TRALE system has been combined with [incr tsdb()], a piece of software for systematic grammar profiling (Oepen and Flickinger 1998). The grammars are accompanied with a set of example phrases that can be analyzed by the grammar. In addition, the test suite files contain ungrammatical word sequences from the literature and ungrammatical sequences that were discovered during the grammar development process. See (Oepen *et al.* 1997; Müller 2004) on the construction of test suites. Since TRALE has a chart display that makes it possible to inspect the parse chart, it is possible to inspect all linguistic objects that are licensed by the grammar, even if they do not play a role in analyzing the particular sentence under consideration. The result of this careful inspection is a collection of ungrammatical word sequences that no theoretical linguist would have been able to come up with, since it is very difficult to find all the possible side effects of an analysis that is not sufficiently constrained. These negative examples are distributed with the grammars and book publications and can help theoretical and computational linguists improve their theories and implementations.

After changing a grammar, the sentences of the respective test suite are parsed and the result can be compared to previous results. This ensures that the coverage of grammars is extended. If constraints in files that are shared among several grammars are changed, the respective grammars are tested as well. The effects that changes to grammar X cause in grammar Y are often unexpected, and hence it is very important to do systematic testing.

I have discussed desiderata for linguistic theories and argued that linguistic theories have reached a level of complexity that cannot be handled by humans without help by computers. I have presented a certain type of UG-based approaches to natural language that assumes that evidence for an entity in a certain language provides evidence for the presence in other languages as well, even though the entity might be covert in the latter languages. I have argued that such lines of argumentation are not appropriate given what we



know about language acquisition today. I have suggested an alternative bottom-up method for constructing theories by developing grammars that are surface-oriented and motivated on a language-specific basis, without stipulating entities that could not be acquired from input of the language under consideration alone. Generalizations regarding language or language classes are derived by extending the number of languages that are considered and by organizing the constraints of the languages under consideration into sets of constraints that are shared by two or more languages. I have defended this method against rather agnostic views maintained by some typologists. Finally, I have provided a brief description of basic assumptions and the basic setup of the CoreGram project and argued that working in such a general setting makes sure that progress is made.

9

#### ACKNOWLEDGEMENTS

I thank Hans-Heinrich Lieb, Antonio Machicao y Priemer and three reviewers of the *Journal of Language Modeling* for comments on earlier versions of this paper; Werner Abraham, Peter Eisenberg, Evelina Fedorenko, Ted Gibson, Matthias Hüning, Tibor Kiss, Jean-Pierre Koenig, Elisabeth Leiss, Bob Levine, Frank Richter, Gerald Penn, Geoffrey Pullum, Uli Reich, Ivan Sag, Anatol Stefanowitsch and Dieter Wunderlich for discussion; and Philippa Cook and Adam Przepiórkowski for comments and proof-reading.

In 2013, parts of this paper were presented at the MIT Gibson Lab, Brain and Cognitive Sciences. I also presented this work in Alexander Koller's Theoretical Computational Linguistics group at the University of Potsdam in 2014, at the ESSLLI 2013 Workshop on *High-level Methodologies for Grammar Engineering* in Düsseldorf, which was organized by Denys Duchier and Yannick Parmentier, and at the workshop *Grammatical categories in macro- and microcomparative linguistics*, which was organized by Martin Haspelmath, Andreas Dufter, and Aria Adli at the annual meeting of the DGfS in Marburg in 2014. I thank the respective organizers for the invitations/inclusions into workshop programs, and all the participants for discussion.

The work reported in this paper was supported by grants from the Deutsche Forschungsgemeinschaft (InfStruk MU 2822/1-1, SFB

632 project A6, DanGram MU 2822/2-1, PerGram MU 2822/3-1, and ChinGram MU 2822/5-1).

Last but not least, I want to thank the copy editors of the Journal of Language Modelling. They did a very good job on the manuscript!

## REFERENCES

- Steven P. ABNEY (1996), Statistical methods and linguistics, in Judith L. KLAVANS and Philip RESNIK, editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, Language, Speech, and Communication, pp. 1–26, MIT Press.
- Steven P. ABNEY and Jennifer COLE (1986), A government-binding parser, *Proceedings of North Eastern Linguistic Society*, 16:1–17.
- Mansour ALOTAIBI and Robert D. BORSLEY (2013), Gaps and resumptive pronouns in Modern Standard Arabic, in *Proceedings of the 20th International Conference on Head-Driven Phrase Structure Grammar*, pp. 6–26, <http://cslipublications.stanford.edu/HPSG/2013/alotaibi-borsley.pdf>.
- Ben AMBRIDGE and Adele E. GOLDBERG (2008), The island status of clausal complements: Evidence in favor of an information structure explanation, *Cognitive Linguistics*, 19:349–381, [http://www.princeton.edu/~adele/Publications\\_files/08Ambridge%26Goldberg-islands.pdf](http://www.princeton.edu/~adele/Publications_files/08Ambridge%26Goldberg-islands.pdf).
- Joseph AOUN and Dominique SPORTICHE (1983), On the formal theory of government, *The Linguistic Review*, 2(3):211–236.
- Mohammad BAHRANI, Hossein SAMETI, and Mehdi Hafezi MANSHADI (2011), A computational grammar for Persian based on GPSG, *Language Resources and Evaluation*, 45(4):387–408.
- Colin BANNARD, Elena LIEVEN, and Michael TOMASELLO (2009), Modeling children’s early grammatical knowledge, *Proceedings of the National Academy of Sciences*, 106(41):17284–17289.
- Emily BENDER and Daniel P. FLICKINGER (1999), Peripheral constructions and core phenomena: Agreement in tag questions, in Gert WEBELHUTH, Jean-Pierre KOENIG, and Andreas KATHOL, editors, *Lexical and Constructional Aspects of Linguistic Explanation*, number 1 in Studies in Constraint-Based Lexicalism, pp. 199–214, CSLI Publications.
- Emily M. BENDER (2000), *Syntactic Variation and Linguistic Competence: The Case of AAVE Copula Absence*, Ph.D. thesis, Stanford University, <http://faculty.washington.edu/ebender/dissertation/>.
- Emily M. BENDER (2008), Grammar engineering for linguistic hypothesis testing, in *Proceedings of the Texas Linguistics Society X Conference: Computational Linguistics for Less-Studied Languages*, pp. 16–36.

*The CoreGram project*

- Emily M. BENDER and Daniel P. FLICKINGER (2005), Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core, in *Proceedings of the 2nd International Joint Conference on Natural Language Processing IJCNLP-05 (Posters/Demos)*, <http://turing.cs.washington.edu/papers/modules05.pdf>.
- Emily M. BENDER, Daniel P. FLICKINGER, and Stephan OEPEN (2002), The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars, in *Proceedings of the Workshop on Grammar Engineering and Evaluation at COLING 2002*, pp. 8–14.
- Benjamin K. BERGEN and Nancy CHANG (2005), Embodied Construction Grammar in simulation-based language understanding, in Jan-Ola ÖSTMAN and Mirjam FRIED, editors, *Construction Grammars: Cognitive Grounding and Theoretical Extensions*, pp. 147–190, John Benjamins Publishing Co.
- Robert C. BERWICK, Paul PIETROSKI, Beracah YANKAMA, and Noam CHOMSKY (2011), Poverty of the stimulus revisited, *Cognitive Science*, 35(7):1207–1242.
- Manfred BIERWISCH (1963), *Grammatik des deutschen Verbs [The Grammar of the German Verb]*, number 2 in *studia grammatica*, Akademie Verlag.
- Mahmood BIJANKHAN (2004), The role of corpora in writing a grammar [article in Persian], *Iranian Journal of Linguistics*, 19(2):48–67.
- Felix BILDHAUER (2008), *Representing Information Structure in an HPSG Grammar of Spanish*, Ph.D. thesis, Universität Bremen.
- Felix BILDHAUER and Philippa COOK (2010), German multiple fronting and expected topic-hood, in *Proceedings of the 17th International Conference on Head-Driven Phrase Structure Grammar*, pp. 68–79.
- Rens BOD (2009), From exemplar to grammar: Integrating analogy and probability in language learning, *Cognitive Science*, 33(4):752–793, <http://staff.science.uva.nl/~rens/analogy.pdf>.
- Olivier BONAMI and Danièle GODARD (2001), Inversion du sujet, constituance et ordre des mots, in Jean-Marie MARANDIN, editor, *Cahier Jean-Claude Milner*, pp. 117–174, Editions Verdier.
- Olivier BONAMI, Daniele GODARD, and Jean-Marie MARANDIN (1998), French subject inversion in extraction contexts, *Proceedings of FHCG*, 98:101–112.
- Robert D. BORSLEY (2004), An approach to English comparative correlatives, in *Proceedings of the 11th International Conference on Head-Driven Phrase Structure Grammar*, pp. 70–92.
- Gosse BOUMA, Robert MALOUF, and Ivan A. SAG (2001), Satisfying constraints on extraction and adjunction, *Natural Language and Linguistic Theory*, 19(1):1–65, <http://ftp-linguistics.stanford.edu/sag/bms-nllt.pdf>.

- Joan BRESNAN and Sam A. MCHOMBO (1995), The lexical integrity principle: Evidence from Bantu, *Natural Language and Linguistic Theory*, 13:181–254.
- Ted J. BRISCOE and Ann COPESTAKE (1999), Lexical rules in constraint-based grammar, *Computational Linguistics*, 25(4):487–526, <http://acl.ldc.upenn.edu/J/J99/J99-4002.pdf>.
- Miriam BUTT, Helge DYVIK, Tracy Holloway KING, Hiroshi MASUICHI, and Christian ROHRER (2002), The parallel grammar project, in *Proceedings of the Workshop on Grammar Engineering and Evaluation at COLING 2002*, pp. 1–7.
- Miriam BUTT, Tracy Holloway KING, María-Eugenia NIÑO, and Frédérique SEGOND (1999), *A Grammar Writer's Cookbook*, number 95 in CSLI Lecture Notes, CSLI Publications.
- Noam CHOMSKY (1957), *Syntactic Structures*, number 4 in *Janua Linguarum, Series Minor*, Mouton.
- Noam CHOMSKY (1959), On certain formal properties of grammars, *Information and Control*, 2(2):137–167.
- Noam CHOMSKY (1964), Degrees of grammaticalness, in Jerry A. FODOR and Jerrold J. KATZ, editors, *The Structure of Language*, pp. 384–389, Prentice-Hall.
- Noam CHOMSKY (1968), Language and the mind, *Psychology Today*, 1(9):48–68, Reprint as: Chomsky 1976.
- Noam CHOMSKY (1971), *Problems of Knowledge and Freedom*, Fontana.
- Noam CHOMSKY (1975), *The Logical Structure of Linguistic Theory*, Plenum Press.
- Noam CHOMSKY (1976), Language and the mind, in Diane D. BORSTEIN, editor, *Readings in the Theory of Grammar: From the 17th to the 20th Century*, pp. 241–251, Winthrop, Reprint from: Chomsky 1968.
- Noam CHOMSKY (1981), *Lectures on Government and Binding*, Foris Publications.
- Noam CHOMSKY (1990), On formalization and formal linguistics, *Natural Language and Linguistic Theory*, 8(1):143–147.
- Noam CHOMSKY (1995), *The Minimalist Program*, number 28 in *Current Studies in Linguistics*, MIT Press.
- Noam CHOMSKY (2001), Derivation by phase, in Michael KENSTOWICZ, editor, *Ken Hale. A Life in Language*, pp. 1–52, MIT Press.
- Noam CHOMSKY (2007), Approaching UG from below, in Uli SAUERLAND and Hans-Martin GÄRTNER, editors, *Interfaces + Recursion = Language? Chomsky's Minimalism and the View from Syntax-Semantics*, number 89 in *Studies in Generative Grammar*, pp. 1–29, Mouton de Gruyter.
- Noam CHOMSKY (2008), On phases, in Robert FREIDIN, Carlos P. OTERO, and Maria Luisa ZUBIZARRETA, editors, *Foundational Issues in Linguistic Theory. Essays in Honor of Jean-Roger Vergnaud*, pp. 133–166, MIT Press.

*The CoreGram project*

- Noam CHOMSKY (2013), Problems of projection, *Lingua*, 130:33–49.
- Guglielmo CINQUE (1999), *Adverbs and Functional Heads. A Cross-Linguistic Perspective*, Oxford University Press.
- Guglielmo CINQUE and Luigi RIZZI (2010), The cartography of syntactic structures, in Bernd HEINE and Heiko NARROG, editors, *The Oxford Handbook of Linguistic Analysis*, pp. 51–65, Oxford University Press.
- Charles Jr. CLIFTON and Penelope ODOM (1966), Similarity relations among certain English sentence constructions, *Psychological Monographs: General and Applied*, 80(5):1–35.
- Ann COPESTAKE (2002), *Implementing Typed Feature Structure Grammars*, number 110 in CSLI Lecture Notes, CSLI Publications.
- Ann COPESTAKE, Daniel P. FLICKINGER, Carl J. POLLARD, and Ivan A. SAG (2005), Minimal Recursion Semantics: An introduction, *Research on Language and Computation*, 4(3):281–332,  
<http://lingo.stanford.edu/sag/papers/copestake.pdf>.
- Nelson CORREA (1987), An attribute-grammar implementation of Government-Binding Theory, in *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pp. 45–51,  
<http://acl.ldc.upenn.edu/P/P87/P87-1007.pdf>.
- Matthew Walter CROCKER and Ian LEWIN (1992), Parsing as deduction: Rules versus principles, in *Proceedings of the 10th European Conference on Artificial Intelligence*, pp. 508–512.
- William CROFT (2001), *Radical Construction Grammar: Syntactic Theory in Typological Perspective*, Oxford University Press.
- Berthold CRYSMANN (2003), On the efficient implementation of German verb placement in HPSG, in *Proceedings of RANLP 2003*, pp. 112–116.
- Peter W. CULICOVER (1999), *Syntactic Nuts: Hard Cases, Syntactic Theory, and Language Acquisition*, volume 1 of *Foundations of Syntax*, Oxford University Press.
- Peter W. CULICOVER and Ray S. JACKENDOFF (2005), *Simpler Syntax*, Oxford University Press.
- Kordula DE KUTHY (2002), *Discontinuous NPs in German*, number 14 in *Studies in Constraint-Based Lexicalism*, CSLI Publications.
- Johannes DELLERT, Kilian EVANG, and Frank RICHTER (2010), Kahina, a debugging framework for logic programs and TRALE, presentation at the 17th International Conference on Head-Driven Phrase Structure Grammar.
- Johannes DELLERT, Kilian EVANG, and Frank RICHTER (2013), Kahina: A hybrid trace-based and chart-based debugging system for grammar engineering, in *Proceedings of the Workshop on High-level Methodologies for Grammar Engineering (HMGE 2013)*, pp. 75–86.

- David R. DOWTY (1979), *Word Meaning and Montague Grammar*, number 7 in Synthese Language Library, D. Reidel Publishing Company.
- David R. DOWTY (1991), Thematic proto-roles and argument selection, *Language*, 67(3):547–619.
- Matthew S. DRYER (1997), Are grammatical relations universal?, in Joan BYBEE, John HAIMAN, and Sandra THOMPSON, editors, *Essays on Language Function and Language Type: Dedicated to T. Givon*, pp. 115–143, John Benjamins Publishing Co.
- Markus EGG (1999), Derivation and resolution of ambiguities in *wieder*-sentences, in *Proceedings of the 12th Amsterdam Colloquium*, pp. 109–114.
- Peter EISENBERG (1992), Platos Problem und die Lernbarkeit der Syntax [Plato's problem and the learnability of syntax], in Peter SUCHSLAND, editor, *Biologische und soziale Grundlagen der Sprache*, number 280 in *Linguistische Arbeiten*, pp. 371–378, Max Niemeyer Verlag.
- Elisabet ENGDAHL and Enric VALLDUVÍ (1994), Information packaging and grammar architecture: A constraint-based approach, in Elisabet ENGDAHL, editor, *Integrating Information Structure into Constraint-Based and Categorical Approaches*, pp. 39–79, ILLC, DYANA-2 Report R.1.3.B.
- Elisabet ENGDAHL and Enric VALLDUVÍ (1996), Information packaging in HPSG, in Claire GROVER and Enric VALLDUVÍ, editors, *Edinburgh Working Papers in Cognitive Science, Vol. 12: Studies in HPSG*, chapter 1, pp. 1–32, Centre for Cognitive Science, University of Edinburgh, <ftp://ftp.cogsci.ed.ac.uk/pub/CCS-WPs/wp-12.ps.gz>.
- Bruno ESTIGARRIBIA (2009), Facilitation by variation: Right-to-left learning of English yes/no questions, *Cognitive Science*, 34(1):68–93.
- Nicholas EVANS and Stephen C. LEVINSON (2009), The myth of language universals: Language diversity and its importance for cognitive science, *The Behavioral and Brain Sciences*, 32(5):429–448.
- Arnold EVERS (1975), *The Transformational Cycle in Dutch and German*, Ph.D. thesis, University of Utrecht.
- Ray FABRI (1993), *Kongruenz und die Grammatik des Maltesischen [Agreement and the Grammar of Maltese]*, number 292 in *Linguistische Arbeiten*, Max Niemeyer Verlag.
- Gisbert FANSELOW (2001), Features,  $\theta$ -roles, and free constituent order, *Linguistic Inquiry*, 32(3):405–437.
- Gisbert FANSELOW (2009), Die (generative) Syntax in den Zeiten der Empiriediskussion [(Generative) syntax in the times of the empirical evidence discussion], *Zeitschrift für Sprachwissenschaft*, 28(1):133–139.
- Janet Dean FODOR (1998), Unambiguous triggers, *Linguistic Inquiry*, 29(1):1–36.

*The CoreGram project*

Janet Dean FODOR (2001), Parameters and the periphery: Reflections on syntactic nuts, *Journal of Linguistics*, 37:367–392.

Jerry A. FODOR, Thomas G. BEVER, and Merrill F. GARRETT (1974), *The Psychology of Language: An Introduction to Psycholinguistics and Generative Grammar*, McGraw-Hill Book Co.

Sandiway FONG (1991), *Computational Properties of Principle-Based Grammatical Theories*, Ph.D. thesis, MIT Artificial Intelligence Lab,  
<http://www.neci.nec.com/homepages/sandiway/pappi/index.html>.

Sandiway FONG (2014), Unification and efficient computation in the Minimalist Program, in L. FRANCIS and L. LAURENT, editors, *Language and Recursion*, pp. 129–138, Springer Verlag.

Sandiway FONG and Jason GINSBURG (2012), Computation with doubling constituents: Pronouns and antecedents in phase theory, in Anna Maria Di SCIULLO, editor, *Towards a Biolinguistic Understanding of Grammar: Essays on Interfaces*, number 194 in *Linguistik Aktuell/Linguistics Today*, pp. 303–338, John Benjamins Publishing Co.

Andrew FORDHAM and Matthew Walter CROCKER (1994), Parsing with principles and probabilities, in *Proceedings of the Workshop “The Balancing Act: Combining Symbolic and Statistical Approaches to Language”*, pp. 37–42.

Robert FREIDIN (1997), Review article: The Minimalist Program, *Language*, 73(3):571–582.

Daniel FREUDENTHAL, Julian M. PINE, and Fernand GOBET (2006), Modeling the development of children’s use of optional infinitives in Dutch and English using MOSAIC, *Cognitive Science*, 30(2):277–310.

Daniel FREUDENTHAL, Julian M. PINE, and Fernand GOBET (2009), Simulating the referential properties of Dutch, German, and English root infinitives in MOSAIC, *Language Learning and Development*, 5(1):1–29.

Joyce FRIEDMAN (1969), Applications of a computer system for Transformational Grammar, in *Proceedings of the International Conference on Computational Linguistics 1969*.

Joyce FRIEDMAN, Thomas H. BREDT, Robert W. DORAN, Bary W. POLLACK, and Theodore S. MARTNER (1971), *A Computer Model of Transformational Grammar*, number 9 in *Mathematical Linguistics and Automatic Language Processing*, Elsevier.

Gerald GAZDAR, Ewan KLEIN, Geoffrey K. PULLUM, and Ivan A. SAG (1985), *Generalized Phrase Structure Grammar*, Harvard University Press.

Edward GIBSON (1998), Linguistic complexity: Locality of syntactic dependencies, *Cognition*, 68(1):1–76.

Jonathan GINZBURG and Ivan A. SAG (2000), *Interrogative Investigations: the Form, Meaning, and Use of English Interrogatives*, number 123 in CSLI Lecture Notes, CSLI Publications.

Mark E. GOLD (1967), Language identification in the limit, *Information and Control*, 10(5):447–474.

Adele E. GOLDBERG (2013), Argument structure constructions vs. lexical rules or derivational verb templates, *Mind and Language*, 28(4):435–465.

Günther GREWENDORF (1988), *Aspekte der deutschen Syntax. Eine Rektions-Bindungs-Analyse [Aspects of German Syntax. A Government and Binding Analysis]*, number 33 in *Studien zur deutschen Grammatik*, originally Gunter Narr Verlag now Stauffenburg Verlag.

Takao GUNJI (1986), Subcategorization and word order, in William J. POSER, editor, *Papers from the Second International Workshop on Japanese Syntax*, pp. 1–21, CSLI Publications.

Hubert HAIDER (1986), Fehlende Argumente: vom Passiv zu kohärenten Infinitiven [Missing arguments: from passive to coherent infinitives], *Linguistische Berichte*, 101:3–33.

Daniel HARBOUR (2011), Mythomania? methods and morals from ‘the myth of language universals’, *Lingua*, 121(12):1820–1830.

Martin HASPELMATH (2010a), Comparative concepts and descriptive categories in crosslinguistic studies, *Language*, 86(3):663–687.

Martin HASPELMATH (2010b), The interplay between comparative concepts and descriptive categories (reply to Newmeyer), *Language*, 86(3):696–699.

Petter HAUGEREID, Nurit MELNIK, and Shuly WINTNER (2013), Nonverbal predicates in Modern Hebrew, in *Proceedings of the 20th International Conference on Head-Driven Phrase Structure Grammar*, pp. 69–89, <http://csli-publications.stanford.edu/HPSG/2013/hmw.pdf>.

Marc D. HAUSER, Noam CHOMSKY, and W. Tecumseh FITCH (2002), The faculty of language: What is it, who has it, and how did it evolve?, *Science*, 298:1569–1579, doi:10.1126/science.298.5598.1569, <http://www.chomsky.info/articles/20021122.pdf>.

Wolfgang HEINZ and Johannes MATIASSEK (1994), Argument structure and case assignment in German, in John NERBONNE, Klaus NETTER, and Carl J. POLLARD, editors, *German in Head-Driven Phrase Structure Grammar*, number 46 in CSLI Lecture Notes, pp. 199–236, CSLI Publications.

Erhard W. HINRICHS and Tsuneko NAKAZAWA (1994), Linearizing AUXs in German verbal complexes, in John NERBONNE, Klaus NETTER, and Carl J. POLLARD, editors, *German in Head-Driven Phrase Structure Grammar*, number 46 in CSLI Lecture Notes, pp. 11–38, CSLI Publications.



*The CoreGram project*

- Anders HOLMBERG (1999), Remarks on Holmberg's generalization, *Studia Linguistica*, 53(1):1–39.
- Norbert HORNSTEIN, Jairo NUNES, and Kleantes K. GROHMANN (2005), *Understanding Minimalism*, Cambridge Textbooks in Linguistics, Cambridge University Press.
- Ray S. JACKENDOFF (1997), *The Architecture of the Language Faculty*, number 28 in Linguistic Inquiry Monographs, MIT Press.
- Ray S. JACKENDOFF (1999), Parallel constraint-based generative theories of language, *Trends in Cognitive Science*, 3(10):393–400.
- Ray S. JACKENDOFF (2008), Construction after construction and its theoretical challenges, *Language*, 84(1):8–28.
- Ray S. JACKENDOFF (2011), What is the human language faculty? Two views, *Language*, 87(3):586–624.
- Ray S. JACKENDOFF and Steven PINKER (2005), The nature of the language faculty and its implications for evolution of language (reply to Fitch, Hauser, and Chomsky), *Cognition*, 97(2):211–225.
- Joachim JACOBS (2008), Wozu Konstruktionen? [Why constructions?], *Linguistische Berichte*, 213:3–44.
- Gerhard JÄGER and Reinhard BLUTNER (2003), Competition and interpretation: The German adverb *wieder* (“again”), in Ewald LANG, Claudia MAIENBORN, and Cathrine FABRICIUS-HANSEN, editors, *Modifying Adjuncts*, number 4 in Interface Explorations, pp. 393–416, Mouton de Gruyter.
- Kent JOHNSON (2004), Gold's theorem and cognitive science, *Philosophy of Science*, 71(4):571–592.
- Ronald M. KAPLAN, Tracy Holloway KING, and John T. MAXWELL III (2002), Adapting existing grammars: The XLE approach, in *Proceedings of the Workshop on Grammar Engineering and Evaluation at COLING 2002*, pp. 29–35, <http://www2.parc.com/isl/groups/nltt/pargram/kaplanetal-coling02.pdf>.
- Gholamhossein KARIMI-DOOSTAN (1997), *Light Verb Constructions in Persian*, Ph.D. thesis, Department of Language and Linguistics, University of Essex.
- Martin KAY (2011), Zipf's law and *L'Arbitraire du Signe*, *Linguistic Issues in Language Technology*, 6(8): Special Issue on Interaction of Linguistics and Computational Linguistics, <http://elanguage.net/journals/index.php/lilt/issue/view/330>.
- Richard S. KAYNE (1994), *The Antisymmetry of Syntax*, number 25 in Linguistic Inquiry Monographs, MIT Press.
- Edward L. KEENAN and Bernard COMRIE (1977), Noun phrase accessibility and universal grammar, *Linguistic Inquiry*, 8(1):63–99.

Tibor KISS (1995), *Infinite Komplementation. Neue Studien zum deutschen Verbum infinitum [Non-finite Complementation. New Studies on the German Non-Finite Verb]*, number 333 in *Linguistische Arbeiten*, Max Niemeyer Verlag.

Jean-Pierre KOENIG and Anthony R. DAVIS (2004), Raising doubts about Russian impersonals, in *Proceedings of the 11th International Conference on Head-Driven Phrase Structure Grammar*.

Jean-Pierre KOENIG and Karin MICHELSON (2012), The (non)universality of syntactic selection and functional application, in *Empirical Issues in Syntax and Semantics*, volume 9, pp. 185–205.

Hans-Peter KOLB (1997), GB blues: Two essays on procedures and structures in generative syntax, *Arbeitspapiere des SFB 340 No. 110*, Eberhard-Karls-Universität, Tübingen.

Hans-Peter KOLB and Craig L. THIERSCH (1991), Levels and empty categories in a Principles and Parameters based approach to parsing, in Hubert HAIDER and Klaus NETTER, editors, *Representation and Derivation in the Theory of Grammar*, number 22 in *Studies in Natural Language and Linguistic Theory*, Kluwer Academic Publishers.

Alexander KOLLER and Stefan THATER (2005), Efficient solving and exploration of scope ambiguities, in *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pp. 9–12, <http://acl.ldc.upenn.edu/P/P05/P05-3003.pdf>.

Ekkehard KÖNIG and Claire MOYSE-FAURIE (2009), Spatial reciprocity: Between grammar and lexis, in Johannes HELMBRECHT, Yoko NISHINA, Yong-Min SHIN, Stavros SKOPETEAS, and Elisabeth VERHOEVEN, editors, *Form and Function in Language Research: Papers in Honour of Christian Lehmann*, number 210 in *Trends in Linguistics. Studies and Monographs*, pp. 57–68, de Gruyter.

András KORNAI and Geoffrey K. PULLUM (1990), The X-bar Theory of phrase structure, *Language*, 66(1):24–50.

Jonas KUHN (2007), Interfaces in constraint-based theories of grammar, in Gillian RAMCHAND and Charles REISS, editors, *The Oxford Handbook of Linguistic Interfaces*, pp. 613–650, Oxford University Press.

Robert J. KUHN (1986), A PROLOG implementation of Government-Binding Theory, in *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, pp. 546–550.

Marie LABELLE (2007), Biolinguistics, the Minimalist Program, and psycholinguistic reality, *Snippets*, 14, <http://www.ledonline.it/snippets/>.

Shalom LAPPIN, Robert D. LEVINE, and David E. JOHNSON (2000), The revolution confused: A response to our critics, *Natural Language and Linguistic Theory*, 18(4):873–890.

*The CoreGram project*

Robert D. LEVINE and Thomas E. HUKARI (2006), *The Unity of Unbounded Dependency Constructions*, number 166 in CSLI Lecture Notes, CSLI Publications.

Charles N. LI and Sandra A. THOMPSON (1981), *Mandarin Chinese. A Functional Reference Grammar*, University of California Press.

Janna LIPENKOVA (2009), *Serienverbkonstruktionen im Chinesischen und ihre Analyse im Rahmen von HPSG [Serial Verb Constructions in Chinese and their Analysis in the Framework of HPSG]*, Master's thesis, Institut für Sinologie, Freie Universität Berlin, <http://hpsg.fu-berlin.de/~lipenkov/magister.html>.

Horst LOHNSTEIN (1993), *Projektion und Linking. Ein prinzipienbasierter Parser fürs Deutsche [Projection and Linking. A Principle-Based Parser for German]*, number 287 in *Linguistische Arbeiten*, Max Niemeyer Verlag.

Alec MARANTZ (2005), Generative linguistics within the cognitive neuroscience of language, *The Linguistic Review*, 22(2–4):429–445.

Mitchell P. MARCUS (1980), *A Theory of Syntactic Recognition for Natural Language*, MIT Press.

William MARSLEN-WILSON (1975), Sentence perception as an interactive parallel process, *Science*, 189(4198):226–228.

Nurit MELNIK (2007), From “hand-written” to computationally implemented HPSG theories, *Research on Language and Computation*, 5(2):199–236.

Walt D. MEURERS, Kordula DE KUTHY, and Vanessa METCALF (2003), Modularity of grammatical constraints in HPSG-based grammar implementations, in *Proceedings of the ESSLLI 2003 Workshop “Ideas and Strategies for Multilingual Grammar Development”*, pp. 83–90, <http://www.sfs.uni-tuebingen.de/~dm/papers/meurers-dekuthy-metcalf-03.html>.

Walt Detmar MEURERS (1999a), German partial-VP fronting revisited, in Gert WEBELHUTH, Jean-Pierre KOENIG, and Andreas KATHOL, editors, *Lexical and Constructional Aspects of Linguistic Explanation*, number 1 in *Studies in Constraint-Based Lexicalism*, pp. 129–144, CSLI Publications, <http://www.sfs.uni-tuebingen.de/~dm/papers/hpsg-volume98/pvp-revisited.html>.

Walt Detmar MEURERS (1999b), *Lexical Generalizations in the Syntax of German Non-Finite Constructions*, Ph.D. thesis, Eberhard-Karls-Universität, Tübingen.

Walt Detmar MEURERS (1999c), Raising spirits (and assigning them case), *Groninger Arbeiten zur Germanistischen Linguistik (GAGL)*, 43:173–226, <http://www.sfs.uni-tuebingen.de/~dm/papers/gagl99.html>.

Walt Detmar MEURERS (2000), Lexical generalizations in the syntax of German non-finite constructions, *Arbeitspapiere des SFB 340 No. 145*, Eberhard-Karls-Universität, Tübingen, <http://www.sfs.uni-tuebingen.de/~dm/papers/diss.html>.

Walt Detmar MEURERS (2001), On expressing lexical generalizations in HPSG, *Nordic Journal of Linguistics*, 24(2):161–217, <http://www.sfs.uni-tuebingen.de/~dm/papers/lexical-generalizations.html>.

Walt Detmar MEURERS, Gerald PENN, and Frank RICHTER (2002), A web-based instructional platform for constraint-based grammar formalisms and parsing, in *Effective Tools and Methodologies for Teaching NLP and CL*, pp. 18–25, <http://www.sfs.uni-tuebingen.de/~dm/papers/acl02.html>, proceedings of the Workshop held at 40th Annual Meeting of the Association for Computational Linguistics. Philadelphia, PA.

George A. MILLER and Kathryn Ojemann MCKEAN (1964), A chronometric study of some relations between sentences, *Quarterly Journal of Experimental Psychology*, 16(4):297–308.

Yves Ch. MORIN (1973), A computer tested Transformational Grammar of French, *Linguistics*, 116(11):49–114.

Gereon MÜLLER (1998), *Incomplete Category Fronting. A Derivational Approach to Remnant Movement in German*, number 42 in *Studies in Natural Language and Linguistic Theory*, Kluwer Academic Publishers.

Gereon MÜLLER (2011), Regeln oder Konstruktionen? Von verblosen Direktiven zur sequentiellen Nominalreduplikation [Rules or constructions? From verbless directives to sequential nominal reduplication], in Stefan ENGELBERG, Anke HOLLER, and Kristel PROOST, editors, *Sprachliches Wissen zwischen Lexikon und Grammatik*, Institut für Deutsche Sprache, Jahrbuch 2010, pp. 211–249, de Gruyter, <http://www.uni-leipzig.de/~muellerg/mu242.pdf>.

Stefan MÜLLER (1999), *Deutsche Syntax deklarativ. Head-Driven Phrase Structure Grammar für das Deutsche [German Syntax Declarative. Head-Driven Phrase Structure Grammar for German]*, number 394 in *Linguistische Arbeiten*, Max Niemeyer Verlag, <http://hpsg.fu-berlin.de/~stefan/Pub/hpsg.html>.

Stefan MÜLLER (2002), *Complex Predicates: Verbal Complexes, Resultative Constructions, and Particle Verbs in German*, number 13 in *Studies in Constraint-Based Lexicalism*, CSLI Publications, <http://hpsg.fu-berlin.de/~stefan/Pub/complex.html>.

Stefan MÜLLER (2003a), Mehrfache Vorfeldbesetzung [Multiple frontings], *Deutsche Sprache*, 31(1):29–62, <http://hpsg.fu-berlin.de/~stefan/Pub/mehr-vf-ds.html>.

Stefan MÜLLER (2003b), Solving the bracketing paradox: an analysis of the morphology of German particle verbs, *Journal of Linguistics*, 39(2):275–325, <http://hpsg.fu-berlin.de/~stefan/Pub/paradox.html>.

Stefan MÜLLER (2004), Example sentences and making them useful for theoretical and computational linguistics, Presentation at the DGfS Jahrestagung: AG Empirische Fundierung der Modellbildung in der Syntax,

*The CoreGram project*

http:

[//hpsg.fu-berlin.de/~stefan/PS/b-ger-ts-dgfs-2004-slides.pdf](http://hpsg.fu-berlin.de/~stefan/PS/b-ger-ts-dgfs-2004-slides.pdf).

Stefan MÜLLER (2005), Zur Analyse der deutschen Satzstruktur [Towards the analysis of the German sentence structure], *Linguistische Berichte*, 201:3–39, <http://hpsg.fu-berlin.de/~stefan/Pub/satz-1b.html>.

Stefan MÜLLER (2006), Phrasal or lexical constructions?, *Language*, 82(4):850–883, <http://hpsg.fu-berlin.de/~stefan/Pub/phrasal.html>.

Stefan MÜLLER (2007a), The Grammix CD Rom: A software collection for developing typed feature structure grammars, in Tracy Holloway KING and Emily M. BENDER, editors, *Grammar Engineering across Frameworks 2007*, Studies in Computational Linguistics ONLINE, CSLI Publications.

Stefan MÜLLER (2007b), *Head-Driven Phrase Structure Grammar: Eine Einführung*, number 17 in Stauffenburg Einführungen, Stauffenburg Verlag, 1st edition, <http://hpsg.fu-berlin.de/~stefan/Pub/hpsg-lehrbuch.html>.

Stefan MÜLLER (2008), Depictive secondary predicates in German and English, in Christoph SCHROEDER, Gerd HENTSCHEL, and Winfried BOEDER, editors, *Secondary Predicates in Eastern European Languages and Beyond*, number 16 in Studia Slavica Oldenburgensia, pp. 255–273, BIS-Verlag, <http://hpsg.fu-berlin.de/~stefan/Pub/depiktiv-2006.html>.

Stefan MÜLLER (2009a), A Head-Driven Phrase Structure Grammar for Maltese, in Bernard COMRIE, Ray FABRI, Beth HUME, Manwel MIFSUD, Thomas STOLZ, and Martine VANHOVE, editors, *Introducing Maltese Linguistics: Papers from the 1st International Conference on Maltese Linguistics*, number 113 in Studies in Language Companion Series, pp. 83–112, John Benjamins Publishing Co., <http://hpsg.fu-berlin.de/~stefan/Pub/maltese-sketch.html>.

Stefan MÜLLER (2009b), On predication, in *Proceedings of the 16th International Conference on Head-Driven Phrase Structure Grammar*, pp. 213–233, <http://hpsg.fu-berlin.de/~stefan/Pub/predication.html>.

Stefan MÜLLER (2010a), *Grammatiktheorie [Grammatical Theory]*, number 20 in Stauffenburg Einführungen, Stauffenburg Verlag, <http://hpsg.fu-berlin.de/~stefan/Pub/grammatiktheorie.html>.

Stefan MÜLLER (2010b), Persian complex predicates and the limits of inheritance-based analyses, *Journal of Linguistics*, 46(3):601–655, <http://hpsg.fu-berlin.de/~stefan/Pub/persian-cp.html>.

Stefan MÜLLER (2012), On the copula, specificational constructions and type shifting, <http://hpsg.fu-berlin.de/~stefan/Pub/copula.html>, draft, Freie Universität Berlin.

Stefan MÜLLER (2013a), *Grammatiktheorie [Grammatical Theory]*, number 20 in Stauffenburg Einführungen, Stauffenburg Verlag, 2nd edition, <http://hpsg.fu-berlin.de/~stefan/Pub/grammatiktheorie.html>.

Stefan MÜLLER (2013b), *Head-Driven Phrase Structure Grammar: Eine Einführung [Head-Driven Phrase Structure Grammar: an Introduction]*, number 17 in *Stauffenburg Einführungen*, Stauffenburg Verlag, 3rd edition, <http://hpsg.fu-berlin.de/~stefan/Pub/hpsg-lehrbuch.html>.

Stefan MÜLLER (2013c), Unifying everything: Some remarks on Simpler Syntax, Construction Grammar, Minimalism and HPSG, *Language*, 89(4):920–950, <http://hpsg.fu-berlin.de/~stefan/Pub/unifying-everything.html>.

Stefan MÜLLER (2014a), Artenvielfalt und Head-Driven Phrase Structure Grammar [Biological diversity and Head-Driven Phrase Structure Grammar], in *Syntaxtheorien: Analysen im Vergleich*, number 28 in *Stauffenburg Einführungen*, pp. 187–233, Stauffenburg Verlag, <http://hpsg.fu-berlin.de/~stefan/Pub/artenvielfalt.html>.

Stefan MÜLLER (2014b), Elliptical constructions, multiple frontings, and surface-based syntax, in *Proceedings of Formal Grammar 2004*, pp. 91–109, <http://hpsg.fu-berlin.de/~stefan/Pub/surface.html>.

Stefan MÜLLER (2014c), Kernigkeit: Anmerkungen zur Kern-Peripherie-Unterscheidung [Coriness: Some remarks on the core-periphery distinction], in Andreas NOLDA, Athina SIOUPI, and Antonio Machicao Y PRIEMER, editors, *Zwischen Kern und Peripherie*, number 76 in *studia grammatica*, pp. 25–39, de Gruyter, <http://hpsg.fu-berlin.de/~stefan/Pub/kernigkeit.html>.

Stefan MÜLLER (2014d), Satztypen: Lexikalisch oder/und phrasal [Sentence types. Lexically and/or phrasally], in Rita FINKBEINER and Jörg MEIBAUER, editors, *Satztypen und Konstruktionen im Deutschen*, *Linguistik – Impulse und Tendenzen*, de Gruyter, To appear.

Stefan MÜLLER (2015a), *German Sentence Structure: An Analysis with Special Consideration of So-Called Multiple Fronting*, Empirically Oriented Theoretical Morphology and Syntax, Language Science Press, <http://hpsg.fu-berlin.de/~stefan/Pub/gS.html>, in Preparation.

Stefan MÜLLER (2015b), *Grammatical Theory: From Transformational Grammar to Constraint-Based Approaches*, number 1 in *Lecture Notes in Language Sciences*, Language Science Press, In Preparation.

Stefan MÜLLER (2015c), HPSG – a synopsis, in Artemis ALEXIADOU and Tibor KISS, editors, *Syntax – Theory and Analysis. An International Handbook*, number 42.2 in *Handbooks of Linguistics and Communication Science*, chapter 27, Walter de Gruyter, 2nd edition, <http://hpsg.fu-berlin.de/~stefan/Pub/hpsg-hsk.html>, In Print.

Stefan MÜLLER, Felix BILDHAUER, and Philippa COOK (2012), Beschränkungen für die scheinbar mehrfache Vorfelddbesetzung im Deutschen [Constraints on apparent multiple frontings in German], in Colette CORTÈS, editor,

*The CoreGram project*

*Satzeröffnung. Formen, Funktionen, Strategien*, number 31 in Eurogermanistik, pp. 113–128, Stauffenburg Verlag.

Stefan MÜLLER and Masood GHAYOOMI (2010), PerGram: A TRALE implementation of an HPSG fragment of Persian, in *Proceedings of the 2010 IEEE International Multiconference on Computer Science and Information Technology – Computational Linguistics Applications (CLA'10)*, volume 5, pp. 461–467, <http://hpsg.fu-berlin.de/~stefan/Pub/pergram.html>.

Stefan MÜLLER and Janna LIPENKOVA (2009), Serial verb constructions in Chinese: An HPSG account, in *Proceedings of the 16th International Conference on Head-Driven Phrase Structure Grammar*, pp. 234–254, <http://hpsg.fu-berlin.de/~stefan/Pub/chinese-svc.html>.

Stefan MÜLLER and Janna LIPENKOVA (2013), ChinGram: A TRALE implementation of an HPSG fragment of Mandarin Chinese, in *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27)*, pp. 240–249, Department of English, National Chengchi University.

Stefan MÜLLER and Janna LIPENKOVA (In Preparation), *Mandarin Chinese in Head-Driven Phrase Structure Grammar*, Empirically Oriented Theoretical Morphology and Syntax, Language Science Press.

Stefan MÜLLER and Bjarne ØRSNES (2011), Positional expletives in Danish, German, and Yiddish, in *Proceedings of the 18th International Conference on Head-Driven Phrase Structure Grammar*, pp. 167–187, <http://hpsg.fu-berlin.de/~stefan/Pub/expletives.html>.

Stefan MÜLLER and Bjarne ØRSNES (2013a), Passive in Danish, English, and German, in *Proceedings of the 20th International Conference on Head-Driven Phrase Structure Grammar*, pp. 140–160.

Stefan MÜLLER and Bjarne ØRSNES (2013b), Towards an HPSG analysis of object shift in Danish, in Glyn MORRILL and Mark-Jan NEDERHOF, editors, *Formal Grammar: 17th and 18th International Conferences, FG 2012/2013*, number 8036 in Lecture Notes in Computer Science, pp. 69–89, Springer Verlag, <http://hpsg.fu-berlin.de/~stefan/Pub/object-shift.html>.

Stefan MÜLLER and Bjarne ØRSNES (2015), *Danish in Head-Driven Phrase Structure Grammar*, Empirically Oriented Theoretical Morphology and Syntax, Language Science Press, <http://hpsg.fu-berlin.de/~stefan/Pub/danish.html>, In Preparation.

Stefan MÜLLER, Pollet SAMVELIAN, and Olivier BONAMI (In Preparation), *Persian in Head-Driven Phrase Structure Grammar*, Empirically Oriented Theoretical Morphology and Syntax, Language Science Press, <http://hpsg.fu-berlin.de/~stefan/Pub/persian.html>.

Stefan MÜLLER and Stephen Mark WECHSLER (2014), Lexical approaches to argument structure, *Theoretical Linguistics*, 40(1–2):1–76, <http://hpsg.fu-berlin.de/~stefan/Pub/arg-st.html>.

Mariacristina MUSSO, Andrea MORO, Volkmar GLAUCHE, Michel RIJNTJES, Jürgen REICHENBACH, Christian BÜCHEL, and Cornelius WEILLER (2003), Broca's area and the language instinct, *Nature Neuroscience*, 6(7):774–781.

Klaus NETTER (1991), Clause union phenomena and complex predicates in German, in Klaus NETTER and Mike REAPE, editors, *Clause Structure and Word Order Variation in Germanic*, DYANA Report R1.1.B, University of Edinburgh.

Frederick J. NEWMAYER (2005), *Possible and Probable Languages: A Generative Perspective on Linguistic Typology*, Oxford University Press.

Frederick J. NEWMAYER (2010), On comparative concepts and descriptive categories: A reply to Haspelmath, *Language*, 86(3):688–695.

Sourabh NIYOGI and Robert C. BERWICK (2005), A Minimalist implementation of Hale-Keyser incorporation theory, in Anna Maria Di SCIULLO, editor, *UG and External Systems: Language, Brain and Computation*, number 75 in *Linguistik Aktuell/Linguistics Today*, pp. 269–288, John Benjamins Publishing Co.

Torbjørn NORDGÅRD (1994), E-Parser: An implementation of a deterministic GB-related parsing system, *Computers and the Humanities*, 28(4–5):259–272.

Geoffrey NUNBERG, Ivan A. SAG, and Thomas WASOW (1994), Idioms, *Language*, 70(3):491–538.

Stephan OEPEN and John A. CARROLL (2000), Parser engineering and performance profiling, *Natural Language Engineering*, 6(1):81–97, <http://www.delph-in.net/itsdb/publications/parsing.ps.gz>.

Stephan OEPEN and Daniel P. FLICKINGER (1998), Towards systematic grammar profiling. Test suite technology ten years after, *Journal of Computer Speech and Language*, 12(4):411–436, <http://www.delph-in.net/itsdb/publications/profiling.ps.gz>, (Special Issue on Evaluation).

Stephan OEPEN, Klaus NETTER, and Judith KLEIN (1997), TSNLP – Test Suites for Natural Language Processing, in John NERBONNE, editor, *Linguistic Databases*, pp. 13–36, CSLI Publications.

Bjarne ØRSNES (2009), Preposed negation in Danish, in *Proceedings of the 16th International Conference on Head-Driven Phrase Structure Grammar*, pp. 255–275.

Gerald PENN (2004), Balancing clarity and efficiency in typed feature logic through delaying, in *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pp. 239–246.

David M. PERLMUTTER (1978), Impersonal passives and the Unaccusative Hypothesis, in *Proceedings of the 4th Annual Meeting of the Berkeley Linguistics Society*, pp. 157–189.

Stanley Roy PETRICK (1965), *A Recognition Procedure for Transformational Grammars*, Ph.D. thesis, Department of Modern Languages, MIT, <http://hdl.handle.net/1721.1/13013>.



*The CoreGram project*

- Colin PHILLIPS (2003), Linear order and constituency, *Linguistic Inquiry*, 34(1):37–90.
- Carl J. POLLARD (1996), On head non-movement, in Harry BUNT and Arthur VAN HORCK, editors, *Discontinuous Constituency*, number 6 in Natural Language Processing, pp. 279–305, Mouton de Gruyter, published Master's thesis from 1990.
- Carl J. POLLARD and Ivan A. SAG (1987), *Information-Based Syntax and Semantics*, number 13 in CSLI Lecture Notes, CSLI Publications.
- Carl J. POLLARD and Ivan A. SAG (1994), *Head-Driven Phrase Structure Grammar*, Studies in Contemporary Linguistics, The University of Chicago Press.
- Paul M. POSTAL (2009), The incoherence of Chomsky's 'Biolinguistic' ontology, *Biolinguistics*, 3(1):104–123.
- Adam PRZEPIÓRKOWSKI (1999), On case assignment and “adjuncts as complements”, in Gert WEBELHUTH, Jean-Pierre KOENIG, and Andreas KATHOL, editors, *Lexical and Constructional Aspects of Linguistic Explanation*, number 1 in Studies in Constraint-Based Lexicalism, pp. 231–245, CSLI Publications.
- Geoffrey K. PULLUM (1985), Assuming some version of X-bar Theory, in *Papers from the 21st Annual Meeting of the Chicago Linguistic Society*, pp. 323–353.
- Geoffrey K. PULLUM (1989), Formal linguistics meets the boojum, *Natural Language and Linguistic Theory*, 7(1):137–143, <http://dx.doi.org/10.1007/BF00141350>.
- Geoffrey K. PULLUM (1991), *The Great Eskimo Vocabulary Hoax and Other Irreverent Essays on the Study of Language*, The University of Chicago Press.
- Geoffrey K. PULLUM (2007), The evolution of model-theoretic frameworks in linguistics, in *Model-Theoretic Syntax at 10 – Proceedings of the ESSLLI 2007 MTS@10 Workshop, August 13–17*, pp. 1–10, <http://cs.earlham.edu/esslli07mts/>.
- Geoffrey K. PULLUM and Barbara C. SCHOLZ (2001), On the distinction between generative-enumerative and model-theoretic syntactic frameworks, in Philippe DE GROOTE, Glyn MORRILL, and Christian RETOR, editors, *Logical Aspects of Computational Linguistics: 4th International Conference*, number 2099 in Lecture Notes in Computer Science, pp. 17–43, Springer Verlag.
- Geoffrey K. PULLUM and Barbara C. SCHOLZ (2002), Empirical assessment of stimulus poverty arguments, *The Linguistic Review*, 19(1–2):9–50.
- Marc RICHARDS (2015), Minimalism, in Artemis ALEXIADOU and Tibor KISS, editors, *Syntax – Ein internationales Handbuch zeitgenössischer Forschung*, volume 42 of *Handbooks of Linguistics and Communication Science*, Mouton de Gruyter, 2nd edition.

- John Robert ROSS (1967), *Constraints on Variables in Syntax*, Ph.D. thesis, MIT, <http://www.eric.ed.gov/>, reproduced by the Indiana University Linguistics Club.
- Ivan A. SAG (1997), English relative clause constructions, *Journal of Linguistics*, 33(2):431–484, <http://lingo.stanford.edu/sag/papers/re1-pap.pdf>.
- Ivan A. SAG (2010), English filler-gap constructions, *Language*, 86(3):486–545, <http://lingo.stanford.edu/sag/papers/xcons.pdf>.
- Ivan A. SAG (2012), Sign-based construction grammar: An informal synopsis, in Hans C. BOAS and Ivan A. SAG, editors, *Sign-based Construction Grammar*, number 193 in CSLI Lecture Notes, pp. 69–202, CSLI Publications, <http://lingo.stanford.edu/sag/papers/theo-syno.pdf>.
- Ivan A. SAG and Thomas WASOW (2011), Performance-compatible competence grammar, in Robert BORSLEY and Kersti BÖRJARS, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar: A Guide to Current Models*, pp. 359–377, Blackwell Publishing Ltd.
- Ivan A. SAG, Thomas WASOW, and Emily M. BENDER (2003), *Syntactic Theory: A Formal Introduction*, number 152 in CSLI Lecture Notes, CSLI Publications, 2nd edition.
- Pollet SAMVELIAN (2007), A (phrasal) affix analysis of the Persian Ezafe, *Journal of Linguistics*, 43:605–645.
- Uli SAUERLAND and Paul ELBOURNE (2002), Total reconstruction, PF movement, and derivational order, *Linguistic Inquiry*, 33(2):283–319.
- Harris B. SAVIN and Ellen PERCHONOCK (1965), Grammatical structure and the immediate recall of English sentences, *Journal of Verbal Learning and Verbal Behavior*, 4(5):348–353.
- Barbara C. SCHOLZ and Geoffrey K. PULLUM (2002), Searching for arguments to support linguistic nativism, *The Linguistic Review*, 19(1–2):185–223.
- Edward P. STABLER (1987), Restricting logic grammars with Government-Binding Theory, *Computational Linguistics*, 13(1–2):1–10.
- Edward P. STABLER (1992), *The Logical Approach to Syntax: Foundations, Specifications, and Implementations of Theories of Government and Binding*, ACL-MIT Press Series in Natural Language Processing, MIT Press.
- Edward P. STABLER (2001), Minimalist grammars and recognition, in Christian ROHRER, Antje ROSSDEUTSCHER, and Hans KAMP, editors, *Linguistic Form and its Computation*, number 1 in Studies in Computational Linguistics, pp. 327–352, CSLI Publications.
- Edward P. STABLER (2010), After Government and Binding Theory, in Johan F. A. K. VAN BENTHEM and G. B. Alice TER MEULEN, editors, *Handbook of Logic and Language*, pp. 395–414, MIT Press, 2nd edition, <http://www.linguistics.ucla.edu/people/stabler/afterGB.pdf>.

*The CoreGram project*

Edward P. STABLER (2011), Computational perspectives on Minimalism, in Cedric BOECKX, editor, *The Oxford Handbook of Linguistic Minimalism*, chapter 27, pp. 616–641, Oxford University Press, <http://www.linguistics.ucla.edu/people/stabler/Stabler10-Min.pdf>.

Mark J. STEEDMAN and Jason BALDRIDGE (2006), Combinatory Categorical Grammar, in Keith BROWN, editor, *Encyclopedia of Language and Linguistics*, pp. 610–621, Elsevier, 2nd edition.

Luc STEELS, editor (2011), *Design Patterns in Fluid Construction Grammar*, number 11 in *Constructional Approaches to Language*, John Benjamins Publishing Co.

Luc STEELS (2013), Fluid Construction Grammar, in Thomas HOFFMANN and Graeme TROUSDALE, editors, *The Oxford Handbook of Construction Grammar*, Oxford University Press.

Oliver SUHRE (1999), *Computational Aspects of a Grammar Formalism for Languages with Freer Word Order*, Master's thesis, Department of Computer Science, Eberhard-Karls-Universität Tübingen, [http://www.sfs.uni-tuebingen.de/hpsg/archive/bibliography/papers/suhre\\_lsl-thesis.pdf](http://www.sfs.uni-tuebingen.de/hpsg/archive/bibliography/papers/suhre_lsl-thesis.pdf).

Michael K. TANENHAUS, Michael J. SPIVEY-KNOWLTON, Kathleen M. EBERHARD, and Julie C. SEDIVY (1995), Integration of visual and linguistic information in spoken language comprehension, *Science*, 268(5217):1632–1634, [http://www.bcs.rochester.edu/people/mtan/publications/1995Tanenhaus\\_Sci.pdf](http://www.bcs.rochester.edu/people/mtan/publications/1995Tanenhaus_Sci.pdf).

Michael K. TANENHAUS, Michael J. SPIVEY-KNOWLTON, Kathleen M. EBERHARD, and Julie C. SEDIVY (1996), Using eye movements to study spoken language comprehension: Evidence for visually mediated incremental interpretation, in Toshio INUI and James L. MCCLELLAND, editors, *Information Integration in Perception and Communication*, number XVI in *Attention and Performance*, pp. 457–478, MIT Press.

Michael TOMASELLO (2003), *Constructing a Language: A Usage-Based Theory of Language Acquisition*, Harvard University Press.

Frank VAN EYNDE and Liesbeth AUGUSTINUS (2014), Complement raising, extraction and adposition stranding in Dutch, in *Proceedings of the 21st International Conference on Head-Driven Phrase Structure Grammar, University at Buffalo*, pp. 156–175, <http://csli-publications.stanford.edu/HPSG/2014/vaneynde-augustinus.pdf>.

Remi VAN TRIJP (2013), A comparison between Fluid Construction Grammar and Sign-Based Construction Grammar, *Constructions and Frames*, 5(1):88–116.

Remi VAN TRIJP (2014), Long-distance dependencies without filler – gaps: A cognitive-functional alternative in Fluid Construction Grammar, *Language and Cognition*, pp. 1–29.

*Stefan Müller*

Mettina VEENSTRA (1998), *Formalizing the Minimalist Program*, Ph.D. thesis, Rijksuniversiteit Groningen.

Arnim VON STECHOW (1996), The different readings of *wieder* “again”: A structural account, *Journal of Semantics*, 13(2):87–138.

Moira YIP, Joan MALING, and Ray S. JACKENDOFF (1987), Case in tiers, *Language*, 63(2):217–250.

Arnold M. ZWICKY, Joyce FRIEDMAN, Barbara C. HALL, and Donald E. WALKER (1965), The MITRE syntactic analysis procedure for Transformational Grammars, in *Proceedings of the FALL Joint Computer Conference*, pp. 317–326, <http://doi.ieeecomputersociety.org/10.1109/AFIPS.1965.108>.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*

<http://creativecommons.org/licenses/by/3.0/>



# A logical approach to grammar description

*Lionel Clément, Jérôme Kirman, and Sylvain Salvati*  
Université de Bordeaux LaBRI  
France

## ABSTRACT

In the tradition of Model Theoretic Syntax, we propose a logical approach to the description of grammars. We combine in one formalism several tools that are used throughout computer science for their power of abstraction: logic and lambda calculus. We propose then a high-level formalism for describing mildly context sensitive grammars and their semantic interpretation. As we rely on the correspondence between logic and finite state automata, our method combines conciseness with effectivity. We illustrate our approach with a simple linguistic model of several interleaved linguistic phenomena involving extraction. The level of abstraction provided by logic and lambda calculus allows us not only to use this linguistic model for several languages, namely English, German, and Dutch, but also for semantic interpretation.

*Keywords:*  
*grammar description,*  
*logic,*  
*Finite State Automata,*  
*logical transduction,*  
*lambda calculus,*  
*Abstract*  
*Categorial*  
*Grammars*

1

## INTRODUCTION

We propose a high-level approach to represent second-order abstract categorial grammars (2-ACGs) of de Groote (2001). This approach is close in spirit to the two step approach proposed by Kolb *et al.* (2003), and also to Model Theoretic Syntax (Rogers 1996). It is also closely related to the recent work of Boral and Schmitz (2013) which advocates and subsequently studies the complexity of context-free grammars with constraints on derivations expressed in propositional dynamic logic (Kracht 1995).

The choice of 2-ACGs as a target class of grammars is motivated by several reasons. First, linear 2-ACGs capture exactly mildly context sensitive languages as shown by de Groote and Pogodalla (2004) and Salvati (2007). In particular they enjoy polynomial parsing algorithms (Salvati 2005 and Salvati 2009), the parsing problem is actually in functional LOGCFL (Kanazawa 2011). Secondly, they allow one to express both syntax and semantics with a very small number of primitives. Thirdly, when dealing with semantics, non-linear 2-ACGs (that is 2-ACGs with copying) have a decidable parsing problem as shown by Salvati (2010) (see Kobele and Salvati 2013 for a more general proof) allowing one to generate text from semantic representation. Finally, following an idea that can be traced back to Curry (1961), they offer a neat separation between syntax, that is how constituents form together a coherent sentence, and word order. Indeed, Abstract categorial grammars (ACGs) split naturally the definition of a language in two parts:

1. the *abstract language* that is meant to represent *deep structures*,
2. the *object language* that is meant to represent *surface structures*.

The mediation between abstract and object languages is made with a *lexicon*. Lexicons, in the context of 2-ACGs, are higher-order homomorphisms mapping each tree of the abstract language to the element of the object language it denotes. Their abstract language is made of ranked trees, which are widely used to model the syntactic structures of languages. They indeed naturally represent the hierarchical structure of natural language syntagmas. Another feature of the ACG approach is that the object language need not consist of strings, but can also be a language of  $\lambda$ -terms representing truth-conditional meanings of sentences. More importantly, different grammars may share the same abstract language, which then serves as a description of relations between the elements of their object languages. In particular, this yields a simple and elegant way of modelling the relation between syntax and semantics, following the work of Montague (1974). Or even, when languages are sufficiently similar, this gives a natural way of constructing synchronous grammars.

Thus, our approach is closely following the ACG's two-level description so as to model the syntax of a natural language. A first assumption we take is that syntactic structures need not represent

directly the word order. This assumption leads us to study syntactic structures as *abstract structures* that satisfy certain properties. These abstract structures are defined by means of a regular tree grammar so as to model the recursive nature of syntax, and are further constrained with logic. We use unordered trees with labelled edges to represent abstract structures. This technical choice emphasizes the fact that syntax and word order are assumed not to be directly connected. The labels of the tree are used to represent the grammatical functions of each node with respect to its parent. Moreover, this structure allows us to define a logical language in which we can describe high-level linguistic properties. This logical language is at the centre of the definition of syntactic validity and also of the mechanism of linearization which associates sentences or meaning representations with abstract structures. As in the ACG setting, we use  $\lambda$ -calculus as a means to achieve complex transformations. When compared to ACGs, the originality of our approach lies in the fact that linearization is guided by logic in a strong way.

Our goal is to design concise and linguistically informed 2-ACGs. For this, as we mentioned, we heavily rely on logic. The reason why we can do so in a computationally effective manner is that sufficiently weak logics can be represented with finite state automata. Seminal results from formal language theory by Doner (1965) and Rabin (1969) have had a wealth of applications in computer science and are still at the root of active research. They also have given rise to the idea of modelling syntax with logic, championed under the name of Model Theoretic Syntax in a series of papers: Rogers (1996), Cornell and Rogers (1998), Rogers (1998), Rogers (2003b), Pullum and Scholz (2001), Pullum and Scholz (2005), Pullum (2007)... One of the successes of Model Theoretic Syntax is the model (Rogers 1998) of the most consensual part of the theory of Government and Binding of Chomsky (1981). It thus showed that this theory could only model context-free languages and was inadequate to model natural languages which contain phenomena beyond context-freeness (see Shieber 1985).

Indeed, the way Model Theoretic Syntax is usually formulated ties word orders to syntactic structures: syntactic structures take the form of trees satisfying the axioms of a linguistic theory and the sentences they represent are simply the sequences of leaves of those trees read

from left to right. This approach has as consequence that only context-free languages can be represented that way. Rogers (2003a) bypasses this limitation by using multidimensional trees. Another approach is the two step approach of Kolb *et al.* (2003) which is based on macro tree transducers and logic. Our approach is similar but, as Morawietz (2003), who proposes to model multiple context-free grammars by means of logical transduction, it relies on logic in a stronger manner and it uses  $\lambda$ -calculus instead of the macro mechanism of macro tree transducers. Indeed, we adapt the notion of logical transductions proposed by Courcelle (1994) (see also Courcelle and Engelfriet 2012) so as to avoid the use of a finite state transduction. This brings an interesting descriptive flavour to the linearization mechanism, which simplifies linguistic descriptions. Thus from the perspective of Model Theoretic Syntax, we propose an approach that allows one to go beyond context-freeness, by relying as much as possible on logic and representing complex linearization operation with  $\lambda$ -calculus. We hope that the separation between abstract language and linearization allows one to obtain some interesting linguistic generalization that might be missed by approaches such as Rogers (2003a), which tie the description of the syntactic constraints and linearization.

The formalism we propose provides high-level descriptions of languages. Indeed, it uses logic as a leverage to model linguistic concepts in the most direct manner. Moreover, as we carefully chose to use trees as syntactic structures and a logic that is weak enough to be representable by finite state automata, the use of this level of abstraction does not come at the cost of the computational decidability of the formalism. Another advantage of this approach that is related to it in a high-level way is conciseness. Finally, merging the ACG approach, Model Theoretic Syntax, and logical transductions allows one to describe in a flexible and relatively simple manner complex realizations that depend subtly on the context. Somehow, we could say that, in our formalization, linearization of abstract structures rely on both a *logical look-around* provided by logical transductions and on *complex data flow* provided by  $\lambda$ -calculus.

#### Related work

The paper is related to the work that tries to give concise descriptions of grammars. It is thus close in spirit to the line of work undertaken



under the name of *Metagrammars*. This work was initiated by Candito (1999) and subsequently developed by Thomasset and De La Clergerie (2005) and Crabbé *et al.* (2013). The main difference between our approach and the Metagrammar approach is that we try to have a formal definition of the languages our linguistic descriptions define, while Metagrammars are defined in a more algorithmic way and tied to rule descriptions. Instead we specify how syntactic structures should look like. Our representation of syntactic structures has a lot in common with f-structures in Lexical Functional Grammars (LFG; Bresnan 2001, Dalrymple 2001), except that we use logic, rather than unification, to describe them. This makes our approach very close in spirit to dependency grammars such as Bröker (1998), Debusmann *et al.* (2004), and Foth *et al.* (2005), property grammars (Blache 2001) and their model theoretic formalization (Duchier *et al.* 2009, 2012, 2014). Most of the fundamental ideas we use in our formalization are similar to those works, in particular Bröker (1998) also proposes to separate syntactic description from linearization. The main difference between LFG, dependency grammars, and our approach is that we try to build a formalization whose expressive power is limited to the classes of languages that are mildly context sensitive and which are believed to be a good fit to the class of natural languages (see Joshi 1985 and Weir 1988).

#### Contribution

We propose a logical language for describing tree structures that is similar to propositional dynamic logic of Kracht (1995). We show how to use this logic to describe abstract structures and their linearization while only defining 2-ACGs. We also show that our formalism can represent in a simple manner various linguistic phenomena in several languages together with the semantics of phrases.

#### Organization of the paper

The paper is divided into two parts: first, Section 2 presents the formalism, while Section 3 presents a grammatical model that is based on that formalism. Section 2 is an incremental presentation of the formalism. We start by explaining how we model abstract structures in Section 2.1. This section explains how our formalization is articulated with lexicons. It gives a definition of the logical language we use

throughout the paper. We then turn to defining the grammatical formalism that combines regular tree grammars and logical constraints that we use to model the valid abstract structures. This section closes with the formal definition of the set of valid abstract structures and an explanation of why this set is a regular set of trees. Then we define the mechanism that linearizes abstract structures and give its formal semantics. The formal semantics of the linearization mechanism is rather complex; moreover, due to space limitations, we need to assume that the reader is familiar with simply-typed  $\lambda$ -calculus (see Hindley and Seldin 2008 and Barendregt 1984 for details).

Section 3 illustrates how the formalism can be used to model languages. It presents a formalization of a fragment of language involving several overlapping extraction phenomena. We start by defining the set of abstract structures, then linearization rules are given that produce from those abstract structures phonological realizations for English, German, and Dutch, and Montagovian semantic representations. In order to clarify the behaviour of the formalism, the section finishes with a detailed example of an intricate sentence involving many of the phenomena we treat.

The article concludes by summarizing the contributions of the paper and discussing the approach and future work.

## 2

## FORMALISM

We will now give an exhaustive definition of the formalism and discuss its underlying linguistic motivations. For the sake of clarity, we exemplify the definitions by means of a toy grammar.

We are first going to explain how we wish to model the trees that represent deep structures of languages.

### 2.1

### *Abstract structure*

Instead of being treated as ranked labelled trees, the abstract structures will be depicted as labelled trees with labelled edges. From a formal point of view this causes no real difficulty as the two presentations of trees can be seen as isomorphic. Nevertheless, from the point of view of grammar design, it is helpful to handle the argument structure of a given syntactic construction by means of names that reflect syntactic functions rather than the relative position of arguments. This

simple choice also makes it more transparent that in ACGs the left-to-right ordering of arguments in the abstract structure does not reflect the word order of their realization in the surface structure. As we will see, for technical convenience, the trees will have two kinds of leaves: lexical entries and the empty leaf  $\perp$ .

Lexical entries

The set of lexical entries, or *vocabulary*, is a set of words along with their properties, as in Table 1. These properties are a set of constants which will represent either a part-of-speech (POS) that governs how lexical entries may be used locally, or some additional syntactic information (like subcategorization, selection restrictions, etc.) that is used to restrict the contexts in which lexical entries may be used. Examples of such properties could be: *proper noun*, *noun*, *determiner*, *verb* (POS) or *intransitive*, *transitive*. Nevertheless, as long as the lexical entries are unambiguously determined by the words they specify, we shall use those very words in place of the lexical entries as a short-hand in the trees we use as examples. Formally, we fix a finite set of words  $W$  and a finite set of properties  $P$ . A vocabulary is then a set of pairs  $(w, Q)$  where  $w \in W$  and  $Q \subseteq P$ .

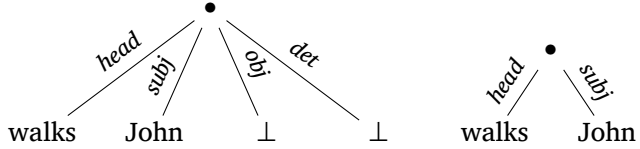
|              |                           |
|--------------|---------------------------|
| <i>John</i>  | <i>proper noun</i>        |
| <i>Mary</i>  | <i>proper noun</i>        |
| <i>man</i>   | <i>noun</i>               |
| <i>a</i>     | <i>determiner</i>         |
| <i>walks</i> | <i>verb, intransitive</i> |
| <i>loves</i> | <i>verb, transitive</i>   |

Table 1:  
Vocabulary example

In all of our examples, apart from the leaves, the nodes of the trees will not be labelled; it is nevertheless important to notice that, if linguistic descriptions require it, the methodology we propose extends with no difficulty to trees with labelled internal nodes. The relation between a node and its child shall be labelled; the labels we use in this example are: *head*, *subj*, *obj*, *det*. We assume that for every internal node  $v$  of a tree and every label  $lbl$ ,  $v$  has a child  $u$  and the edge between  $v$  and  $u$  has the label  $lbl$ . Nevertheless, when  $u$  is a leaf labelled  $\perp$ , we shall not draw it in the picture representations of trees. These technical assumptions are made so as to have a clean treatment of optional constructions of nodes in regular tree grammars and in

logical constraints. These optional constructions are interesting when one seeks concision. Figure 1 shows both a complete tree and the way we draw it.

Figure 1:  
Tree logical structure and  
tree drawing examples



To make it clear that the trees we use are just a variant of the notation of the ranked trees, we explain how to represent the trees we use as ranked trees. For this, it suffices to fix an arbitrary total order on the set of labels and to define term constructors that consist in subsets  $S$  of labels whose arity is the cardinal of  $S$ . Then the  $k^{th}$  argument of the constructor  $S$  represents the child with the  $k^{th}$  label in  $S$  according to the fixed order of labels. For example, fixing a total order where the label *head* precedes the label *subj*, the term representation of the tree in Figure 1 is  $\{head, subj\} \text{ walks John}$ .

Formally, given a finite set of edge labels  $\Sigma$ , we define a *tree domain*  $\text{dom}(t)$  as being a non-empty finite subset of  $\Sigma^*$ , that is prefix-closed and so that if for  $a$  in  $\Sigma$ ,  $ua$  is in  $\text{dom}(t)$ , then for each  $b$  in  $\Sigma$ ,  $ub$  is in  $\text{dom}(t)$ . Given a tree domain  $\text{dom}(t)$ , we write  $\overline{\text{dom}(t)}$  for the set of longest strings in  $\text{dom}(t)$ . The elements of  $\overline{\text{dom}(t)}$  are the positions that correspond to leaves in the tree domain. Given a finite set of labels  $\Lambda$ , a *tree*  $t$  is a pair  $(\text{dom}(t), \text{lbl} : \overline{\text{dom}(t)} \rightarrow \Lambda \cup \{\perp\})$ .<sup>1</sup> The set  $\Lambda$  of labels shall be the vocabulary, while  $\Sigma$  shall be the set of syntactic functions.

#### Logical definition of abstract languages

We have now settled the class of objects that will serve as elements of our abstract language. We then lay out how the set of valid abstract structures is defined, that is how we specify which abstract structures are the syntactically correct ones.

This process will be carried on by logic, in the sense that the set of valid abstract structures will be the set of all trees that satisfy some logical constraints. Provided that the logic expressing those constraints is

<sup>1</sup> Of course, we assume that  $\perp$  is not an element of  $\Lambda$ .

kept simple enough, the resulting abstract language will be both suitably structured and concisely described, while being recognizable by a finite state automaton.

In order to satisfy this last condition, we shall restrict our attention to the class of logical languages that only define regular tree languages. There are several reasons for this. First of all, it is easy to represent the run of a tree automaton as the abstract language of a 2-ACG, and, therefore, logical constraints that only define regular languages can be compiled as abstract languages of 2-ACGs. Second, those logics have decidable satisfiability problems and thus it is in principle possible to automatically check the coherence of a set of constraints or check whether valid abstract structures satisfy a given property. Moreover, neither of those properties are preserved in general when considering more powerful logics. Finally, it seems that linguistic constraints do not need extra logical power. The most expressive and concise logic that is known in this class is Monadic Second-Order Logic (MSOL), but various kinds of first-order or modal logics may suit very well the needs of linguistics.

#### The logical language

We define a first-order logical language that we believe is a good candidate for describing the linguistically relevant properties of abstract structures. The set of well-formed formulae in this logic is defined in the usual way for first-order logic, with the conventional connectives ( $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \exists, \forall$ ) and first-order variables ( $x, y, z, \dots$ ) that will be interpreted as positions in the tree. Then, atomic formulae will be based on the following predicates and relations.

First, we assume that we have been given a vocabulary such as the one in Table 1 that uses the finite set of properties  $P$ . Each element  $p$  of  $P$  (listed on the right in the tables representing vocabularies) will correspond to a unary predicate  $p(x)$  in our logic. By definition, such a predicate  $p$  will be true if and only if  $x$  is the position of a leaf in the tree that is labelled by a lexical entry containing  $p$  in its list of properties. From a linguistic point of view, those predicates allow us to talk about the lexical properties of words and ensure that the sentence structure is in accordance with those properties (which can be used to deal with agreement, verb valency, control, etc.).

Then, we add another predicate noted  $none(x)$  which is true if and only if  $x$  is a leaf labelled with  $\perp$ . This will be particularly useful in the case of optional arguments. This predicate will enable us to condition the presence or the absence of an argument with respect to the context. We shall also write  $some(x)$  as a short-hand for  $\neg none(x)$ .

Since we have decided to leave the internal nodes of the tree unlabelled, no additional single-argument predicate is required. Had we chosen to add linguistic information to internal nodes, we could have introduced a set of corresponding predicates to take this information into account when defining the set of valid abstract structures.

Finally, we add a countable set of binary relations that express properties about paths between nodes. This set is defined as the set of all regular expressions over the alphabet of argument labels. If we assume that the set of argument labels is  $\Sigma$ , then regular expressions are defined inductively with the following grammar:

$$reg ::= \varepsilon \mid \Sigma \mid (reg + reg) \mid reg \, reg \mid (reg)^*$$

The language denoted by a regular expression is defined as usual ( $\varepsilon$  denoting the empty word). We shall also take the liberty of dropping useless parentheses. Let  $e$  be such a regular expression, we write  $L(e)$  for the language defined by  $e$ . Then  $e(x, y)$  is a well-formed formula that is true if and only if  $x$  is an ancestor of  $y$  and the (possibly empty) sequence of edge labels  $l_i$  on the path between  $x$  and  $y$  induces a word  $w = l_1 \dots l_n$  such that  $w \in L(e)$ . This set of relations could also be obtained indirectly, by using the more usual finite set of successor relations and either adding a transitive closure operator to first-order logic or using the full power of Monadic Second-Order Logic. In either case, this set of relations is intended to enable the description of long-distance phenomena in sentences (as, for example, wh-movement). In order to shorten some formulae, we also add the following relation notation:  $e_1 \uparrow e_2(x, y)$  which is true if and only if the lowest common ancestor  $z$  of  $x$  and  $y$  is such that  $e_1(z, x)$  and  $e_2(z, y)$ . We also use the shorthand  $any$  to denote any element of  $\Sigma$ . Notice that the relation  $e_1 \uparrow e_2(x, y)$  can indeed be expressed as:

$$\exists z. e_1(z, x) \wedge e_2(z, y) \wedge \forall z'. (any^*(z', x) \wedge any^*(z', y)) \Rightarrow any^*(z', z)$$

Formally, given a tree  $t = (\text{dom}(t), \text{lbl})$ , a formula  $\varphi$ , and a valuation  $\nu$  that maps the free variables of  $\varphi$  to elements of  $\text{dom}(t)$  we define the validity relation  $t, \nu \models \varphi$  by induction on  $\varphi$ :<sup>2</sup>

- $t, \nu \models \text{true}$  is always correct,
- $t, \nu \models p(x)$  iff  $\nu(x)$  is in  $\overline{\text{dom}(t)}$  and  $\text{lbl}(\nu(x)) = (w, Q)$  with  $p \in Q$ ,
- $t, \nu \models \text{none}(x)$  iff  $\nu(x)$  is in  $\overline{\text{dom}(t)}$  and  $\text{lbl}(\nu(x)) = \perp$ ,
- $t, \nu \models e(x, y)$  iff  $\nu(x) = w_1$ ,  $\nu(y) = w_1 w_2$ , and  $w_2 \in L(e)$  for some  $w_1$  and  $w_2$  in  $\text{dom}(t)$ ,
- $t, \nu \models \varphi \vee \psi$  iff  $t, \nu \models \varphi$  or  $t, \nu \models \psi$ ,
- $t, \nu \models \neg\varphi$  iff it is not the case that  $t, \nu \models \varphi$ ,
- $t, \nu \models \exists x.\varphi$  iff there is  $u$  in  $\text{dom}(t)$  so that  $t, \nu[x \leftarrow u] \models \varphi$ , where  $\nu[x \leftarrow u]$  is the valuation that maps every variable  $y$  different from  $x$  to  $\nu(y)$  and maps  $x$  to  $u$ .

#### Regular over-approximation of abstract structures

Though we believe that the class of logical formulae described above constitutes a powerful tool to describe the abstract structures of human languages, we also think that the recursive shape of these structures can be expressed by simpler and more concise means. Hence, we suggest to use regular tree grammars to provide an over-approximation of the intended abstract language, and then refine this sketch by adding logical constraints on the grammar's productions to filter out the undesired structures. Thus, we gain the ability to model the predicate-argument structure in a more readable way. In general, the regular grammar aims at modelling the recursive structure of natural languages while the constraints are meant to express relations between constituents and to ensure that these relations satisfy the grammatical constraints of the language.

This over-approximation is defined by means of a regular tree grammar. Figure 2 gives such a grammar as an example. Note that some non-terminals may occur between parentheses in the right-hand sides of some rules. The intended meaning is that they are optional: given a non-terminal  $X$ , we may think of  $(X)$  as a non-terminal that can

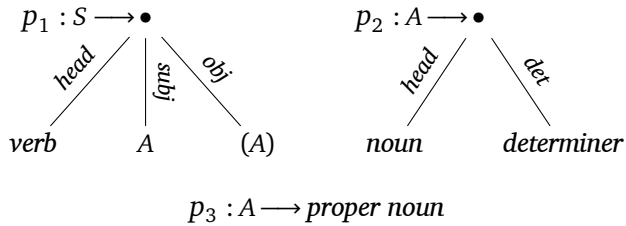
---

<sup>2</sup>We only treat the connectives  $\exists$ ,  $\vee$ , and  $\neg$  which are sufficient to express all the other logical connectives.

be rewritten to either  $X$  or  $\perp$ . Other non-terminals are simply properties of lexical entries (one could also use sets of properties), these non-terminals may be rewritten to any lexical entry which contains this property in its list of properties.

This over-approximation simply puts in place the definitions of linguistic syntagmas so as to model the hierarchical structure of language constructs. From the perspective of grammatical design, such an over-approximation should be based on high-level linguistic considerations and only take care of simple local constraints, accounting for the universals of language, or for the common features of a given family of languages. In particular, it should only use the broadest and simplest lexical properties, such as parts-of-speech.

Figure 2:  
Over-approximating  
regular tree grammar  
example



#### Constraining the regular productions

We now describe how the logical language will be used to refine the regular tree grammar productions that over-approximate the language of abstract structures.

The general idea is that one or several logical formulae can be attached to each production rule of the regular grammar. For this, some nodes on the right-hand sides of rules are tagged with pairwise distinct variables (see Figure 3), and the rules are paired with a set of formulae whose free variables range over the variables that tag their right-hand sides. Now when a rule is used in the course of a derivation, the nodes it creates are constrained by the logical formula paired with the rule. Thus, once a derivation is completed, the resulting tree is considered *valid* only when it satisfies all the constraints that have been introduced in the course of the derivation.

Let us consider Figure 3 as an example: the first production  $p_1$  of our toy grammar is now tagged with two variables respectively named



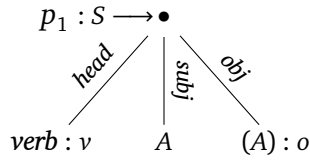


Figure 3:  
Labelled production rule  
with logical guards

$$\begin{aligned} some(o) &\Rightarrow transitive(v) \\ none(o) &\Rightarrow intransitive(v) \end{aligned}$$

$v$  and  $o$  and which respectively designate the *head* and *obj* arguments of the root node. These labels are indicated after colons at the position that they correspond to. Below the rewrite rule is a list of logical constraints that deal with verb valency, and the logical formula  $\varphi(v, o)$  that the final abstract tree must satisfy is implicitly taken to be the conjunction of those two constraints.

We now give a formal definition of what it means for a tree to be *valid*. A derivation is seen as generating a triple  $(t, \nu, \varphi)$  where  $t$  is a tree,  $\varphi$  a logical formula, and  $\nu$  a valuation of the free variables of  $\varphi$  in  $\text{dom}(t)$ . The rules of the grammar act on these triples as follows: if  $(t, \nu, \varphi)$  is such that at the position  $u$ ,  $t$  has a leaf labelled with the non-terminal  $A$  and if there is a rule that rewrites  $A$  into  $s$  with the constraints  $\varphi_1(x_1, \dots, x_n), \dots, \varphi_p(x_1, \dots, x_n)$ , then  $(t, \nu, \varphi)$  rewrites into

$$(t', \nu', \varphi \wedge \varphi_1(x'_1, \dots, x'_n) \wedge \dots \wedge \varphi_p(x'_1, \dots, x'_n))$$

where:

- $t'$  is obtained from  $t$  by replacing the occurrence of  $A$  at the position  $u$  by  $s$ ,
- $x'_1, \dots, x'_n$  are fresh variables,
- $\nu'$  is the valuation that maps every variable  $x$  distinct from the  $x'_i$ 's to  $\nu(x)$  and that maps each variable  $x'_i$  to  $\nu u_i$  when  $u_i$  is the position of the node that is tagged with  $x_i$  in  $s$ .

Now, a tree  $t$  that does not contain any occurrence of a non-terminal is valid when, with  $\emptyset$  being the empty valuation and  $S$  being the starting symbol of the regular grammar,  $(S, \emptyset, true)$  rewrites (in any number of steps) to  $(t, \nu, \varphi)$ , so that  $t, \nu \models \varphi$ . We shall call *language* or *set of valid*

*trees* the set of trees that are generated by the regular grammar and satisfy the logical constraints.

#### Compilation of logical constraints

We are going to show here that the set of valid trees, i.e. the trees generated by the regular grammar that satisfy the logical constraints, is also a regular set. Actually, since we restricted ourselves to a logical language weaker than MSOL, the constraints can be seen as a sort of regular “look-around” for the regular grammar which explains why the valid trees form a regular language. We outline here a construction that defines effectively the language of valid trees as a regular language. This construction is going to be at a rather high-level and is mainly meant to convince the reader that the set of valid trees is indeed regular.

In order to simplify the construction, we first transform the set of constraints associated with rules that bear on several nodes in the right-hand side of a rule into a unique constraint that bears on the root node of the right-hand side. For this, if  $\varphi_1(x_1, \dots, x_n), \dots, \varphi_p(x_1, \dots, x_n)$  is the set of constraints that are associated with a production  $r$ , then there is a unique path labelled with the word  $e_i$  that leads from the root of the tree in the right-hand side of  $r$  to the node labelled  $x_i$ , and then, for the nodes labelled  $x_1, \dots, x_n$ , to satisfy the constraints  $\varphi_1(x_1, \dots, x_n), \dots, \varphi_p(x_1, \dots, x_n)$  is equivalent to the root satisfying the unique constraint:

$$\psi_r(x) = \exists x_1, \dots, x_n. e_1(x, x_1) \wedge \dots \wedge e_n(x, x_n) \wedge \bigwedge_{i=1}^p \varphi_i(x_1, \dots, x_n)$$

Thus, for the construction we are going to present, we assume that each rule has a unique constraint that bears on the root of its right-hand side. Given such a grammar  $G$ , we first remark that the set  $\mathcal{F}$  of constraints used in rules is finite. We then construct a grammar  $G'$  so that each node of  $G'$  is labelled with the set of constraints included in  $\mathcal{F}$  that it needs to satisfy. Hence, in the trees generated by  $G'$ , sets of formulae are labels of internal nodes. We extend our logical language with predicates that *reify* those labels. Thus, given a set of formulae  $S$  included in  $\mathcal{F}$  we define a unary predicate  $[S](x)$  that holds true on nodes  $x$  that are labelled with  $S$ . The predicates used to define the constraint language keep

their former meaning. We can now define a formula *valid* as follows:

$$valid ::= \bigwedge_{S \subseteq \mathcal{F}} \left( \forall x. [S](x) \Rightarrow \bigwedge_{\varphi(x) \in S} \varphi(x) \right)$$

As *valid* is a constraint that is definable in the logical language we have introduced which in turn can be represented in Monadic Second-Order Logic, the set of trees that satisfy this constraint is regular. Thus, the set  $V$  of trees generated by  $G'$  that satisfy *valid*, being the intersection of two regular sets, is also regular. Now, the set of valid trees of  $G$  is precisely the set of trees in  $V$  where the labels of internal nodes have been erased. As regular languages are closed under relabelling, this explains why the set of valid trees is regular.

Let us now briefly sketch how  $G'$  is constructed. Its non-terminals are pairs  $(A, S)$  so that  $A$  is a non-terminal of  $G$  and  $S$  is included in  $\mathcal{F}$ . Each rule  $r$  of  $G$  of the form  $A \rightarrow t$  with constraint  $\varphi(x)$  on its root is mapped to a rule  $(A, S) \rightarrow t'$  of  $G'$  so that if  $t$  is reduced to a non-terminal  $B$ , then  $t'$  is  $(B, S \cup \{\varphi(x)\})$ ; if  $t$  is not reduced to a non-terminal, then  $t'$  is the tree  $t$  where the occurrences of non-terminals  $B$  of  $G$  are replaced by the non-terminals  $(B, \emptyset)$  of  $G'$  and the root of  $t'$  is labelled with the set of formulae  $S \cup \{\varphi(x)\}$ . This transformation is illustrated in Figure 4.

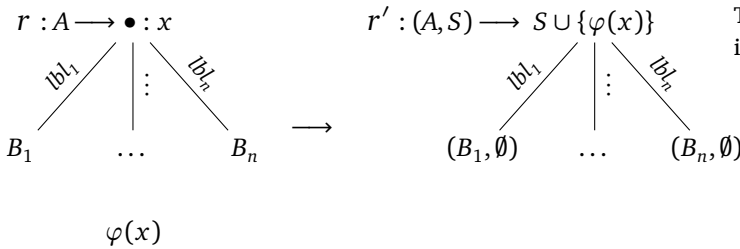


Figure 4:  
Transformation of a rule  
in  $G$  into rules of  $G'$

## 2.2

### The linearization process

We are now going to explain how we intend to linearize the accepted sentences, by describing mappings from the set of valid abstract structures to various languages of surface realizations, which may either

represent the actual sequence of words, or the semantic interpretation of the sentence, or any other structure of interest. Since we have elected to work within the framework of second order ACGs, linearizations can be seen as high-level specifications of lexicons (in the sense of abstract categorial grammars), that is to say morphisms from the trees that belong to the abstract language to simply typed  $\lambda$ -terms of a specific object language. The signature upon which we build the simply typed  $\lambda$ -terms of the object language may vary, but we give here some straightforward examples of target languages for our toy grammar. We assume that the reader is familiar with simply typed  $\lambda$ -calculus (see Hindley and Seldin (2008) and Barendregt (1984) for more details), and contrary to what is usual in ACG, we also use product types, that is the ability to use typed pairs and the related projections in the calculus. It is well-known that this does not increase the expressive power of ACGs, but these constructs are often convenient and intuitive.

#### Surface structures

When mapping abstract structures to surface structures of a language (English, German, and Dutch in this paper), we assume that we can freely handle sequences of words within simply typed  $\lambda$ -calculus (a canonical encoding of those sequences is given by de Groote (2001)).

When dealing with mapping abstract structures to meaning representations, we build an appropriate signature for Montague-style semantics with atomic types that denote propositions ( $p$ ) and entities ( $e$ ) and a set of constants that include the usual logical connectives ( $\neg^{p \rightarrow p}$ ,  $\wedge^{p \rightarrow p \rightarrow p}$ ,  $\exists^{(e \rightarrow p) \rightarrow p}$ , etc.). We add additional constants for verb predicates ( $walks^{e \rightarrow p}$ ,  $loves^{e \rightarrow e \rightarrow p}$ ) and actual entities ( $John^e$ ,  $Mary^e$ ). Notice that to avoid confusion between the logical formula we use for syntax from the logical formula representing truth-conditions in Montague semantics, we use a different font for the connectives of the two logical languages. There are many other choices of signatures and constants for semantic representation, depending on which theory of semantic representation one adheres to. Nevertheless, any set of formulae that can be adequately represented by terms of a 2-ACG's object language may be used for semantic representation in this formalism.

The linearization process

The mapping between abstract structures and surface structures is defined by associating *linearization rules* with the production rules of the regular grammar. This mapping is mediated by the analyses of abstract structures by the regular grammar. Realizations are indeed associated to parse trees of abstract structures in the regular grammar. Nevertheless, as we wish to guide the way realizations are computed with logical constraints over abstract structures, we need to relate nodes of parse trees to nodes in abstract structures. This relation is as follows: each node in the parse tree corresponds to the use of a rule. Such a rule rewrites a non-terminal to a tree that occurs in the abstract structure. As a convention, *we associate the root of that tree with the node in the parse tree*. Notice that due to possible  $\epsilon$ -rules in the regular grammar, i.e. rules of the form  $A \rightarrow B$ , where  $B$  is another non-terminal, there may be several nodes in the parse trees that are related to the same node in the abstract structure; there may also be nodes in the abstract structure that are not related to any node in the parse tree by our convention. This is simply because they are inner nodes of some right-hand side of a rule. Observe also that when a node in the abstract structure is related to several nodes in the parse tree, all those nodes form a chain in the parse tree (all of them correspond to an  $\epsilon$ -rule) and they are thus totally ordered. Since, once we have fixed a parse tree, it is convenient to associate realizations with nodes in the abstract structure, we take the convention that the realization of a node  $x'$  in the abstract structure is the realization of the node  $x$  at the highest position in the parse structure that is related to  $x'$ .

The realization of nodes in the parse tree may depend on several parameters: (i) the realization of the other non-terminals that occur in the right-hand side of the production, (ii) the context in which the rule is used (for example the realization of German or Dutch subordinate clauses differ from that of the main clause), (iii) the realization of nodes that appear elsewhere in the abstract structure, typically, this shall be the case in the presence of wh-movement.

In order to take all those constraints into account, given a rule  $A \rightarrow t$  of the regular grammar, we tag the non-terminal  $A$  with a variable  $x_0$  and assume that the non-terminals that occur in  $t$  are labelled with

the variables  $x_1, \dots, x_n$ . Then the linearization rules are expressed as a list of the form:

$$\mathit{real}(x_0) ::= \varphi(x_0, x_1, \dots, x_n, y_1, \dots, y_m) \rightarrow \\ M[\mathit{real}(x_1), \dots, \mathit{real}(x_n), \mathit{real}(y_1), \dots, \mathit{real}(y_m)]$$

where  $M$  is a simply typed  $\lambda$ -term that is meant to combine the realizations of the nodes denoted by  $x_0, \dots, x_n, y_1, \dots, y_m$ . The variables  $y_1, \dots, y_m$  are not tagging any node in the right-hand side of the rule. The variables  $y_1, \dots, y_m$  represent nodes of a complete abstract structure (i.e. nodes from the context in which the rule is used), which makes the formula  $\varphi(x_0, x_1, \dots, x_n, y_1, \dots, y_m)$  true in the abstract structure (here  $x_0$  is interpreted as the node in the abstract tree that is related to the use of the rule, i.e., by our convention, the root of the subtree generated by the rule). In linearization rules, we shall call *internal variables* those variables (the  $x_i$ 's) that are tagging the production rules, while we shall call the other (the  $y_i$ 's) *external variables*. The intended meaning of such a rule is that given nodes  $y_1, \dots, y_m$  in the abstract structure so that  $\varphi(x_0, x_1, \dots, x_n, y_1, \dots, y_m)$  holds true, if the realizations of  $x_1, \dots, x_n, y_1, \dots, y_m$  respectively are  $\mathit{real}(x_1), \dots, \mathit{real}(x_n), \mathit{real}(y_1), \dots, \mathit{real}(y_m)$  then the realization  $\mathit{real}(x_0)$  of  $x_0$  is the (simply typed)  $\lambda$ -term

$$M[\mathit{real}(x_1), \dots, \mathit{real}(x_n), \mathit{real}(y_1), \dots, \mathit{real}(y_m)] .$$

The realization of lexical entries needs to be explicitly given. For the particular case of phonological realizations, we assume that each lexical entry is realized as the very word given by the entry. Then *a realization of a parse tree* is a realization of its root. By extension, *a realization of an abstract structure* is a realization of one of its parse trees.

Importantly, two different linearization rules need not use  $\lambda$ -terms that have the same type. Indeed, depending on the context, a rule may give rise to realizations that have distinct types. An example of this is provided by the realizations of Dutch clauses depending on whether they are relative clauses or main clauses: in the case of main clauses, the realization is simply a string, while in the case of relative clauses, the realization is a pair of strings so as to compute the cross serial placement of arguments and verbs (see Section 3.1).

The use of external variables is motivated by the linguistic notion of movement in syntax. Indeed, we shall see in Section 3 how to *move*

a relative pronoun from its canonical place in the abstract structure to its landing site in front of the linearization of a relative clause.

*A priori*, linearization rules associate non-deterministically a set of realizations with a given parse tree of an abstract structure. Indeed, there are two sources of non-determinism: (i) there may be several linearization rules that may apply in a given node of the parse tree, (ii) there may be several tuples  $y_1, \dots, y_m$  that make the formula  $\varphi(x_0, x_1, \dots, x_n, y_1, \dots, y_m)$  true. The use of non-determinism may be of interest for linguistic models where some surface variation has no incidence on the syntactic relations between the constituents like for example the order of circumstantial clauses in French.

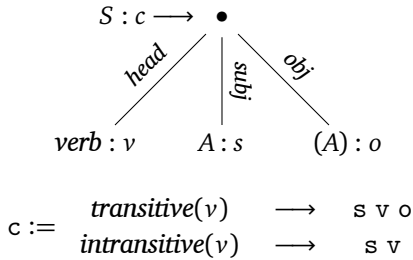


Figure 5:  
Example of a guided  
linearization

Figure 5 gives an example of a linearization rule. This rule, as most of the rules we shall meet later, does not use external variables. To make the writing of rules shorter, we shall write realizations in the teletype font, that is, v, s, o instead of *real(v)*, *real(s)*, *real(o)*.

It is worthwhile to notice that the presence of external variables can be problematic for realizations. Indeed, using this mechanism, it is not hard to realize two nodes  $x$  and  $y$  so that the realization of  $x$  depends on that of  $y$  and vice versa. In such a case we assume that the realization is ill-formed and do not consider it. In the linguistic examples we have considered so far, this situation has never arisen as the external variables  $y_1, \dots, y_m$  on which the realization of a node  $x$  depends are always strictly dominated by that node  $x$ . Nevertheless, from a theoretical point of view, we show in the discussion at the end of the section that the situations giving rise to circular definitions can be filtered out with usual finite state automata techniques.

We now give a formal definition of what it means for an abstract structure  $t$  to be *realized* by a term  $M$ . For the sake of simplicity and without loss of generality (as we have seen in Section 2.1, page 100),

we assume that the grammars we use have constraints that bear only on the root of the right-hand sides of rules. Thus, given a constrained grammar with linearization rules, such a grammar generates 5-tuples  $(E, V, t, \nu, \varphi)$  where:

- $t$  is a tree,
- $\varphi$  is a logical formula,
- $\nu$  is a valuation of some of the free variables of  $\varphi$  in  $\text{dom}(t)$ ,
- $V$  is a function from the positions of  $t$  which are labelled with non-terminals to variables that are free in  $\varphi$ ,
- $E$  is a deterministic grammar (i.e. each of its non-terminals can be rewritten with at most one rule) whose non-terminals are the free variables in  $\varphi$ ; the rules of the grammar rewrite non-terminals to  $\lambda$ -terms (that may contain occurrences of non-terminals). Moreover, the variables occurring in the right-hand sides of rules but not in the left-hand sides are either variables that are mapped to a position of a non-terminal in  $t$  by  $V$ , or which are not in the domain of  $\nu$  (i.e. external variables).

As we have defined valid abstract structures,  $t$  is the abstract structure being produced,  $\varphi$  is a logical formula that the completely derived tree needs to verify. The valuation  $\nu$  is a bit different from the definition of valid abstract structures in that it does not map every variable that is free in  $\varphi$  to a node in  $t$ . This is due to the *external variables* that need to be found once the derivation is completed.

The other elements of the tuple, namely  $E$  and  $V$ , are there to construct a parse tree and maintain the relation between the nodes of the parse tree and the nodes of  $t$ , respecting the convention we spelled out earlier. The unique derivation of  $E$  actually represents the parse tree being constructed, while its rewriting rules contain the necessary information to construct the realization. The function  $V$  maps the non-terminals occurring in  $t$  to variables that shall later be used in the construction of  $E$  once they are rewritten. The relation between the nodes in the parse tree and the nodes in the abstract tree is maintained by  $\nu$  via the use of variables: a variable  $x$  that is a non-terminal in  $E$  represents the use of a rule (i.e. a node in the parse tree) which is related to the node  $\nu(x)$  of  $t$ . The role of  $V$  is to permit the extension of the relation in the course of the derivation.



Let us now see how this works. A rule of the grammar such as the one given in Figure 6 can act on such a tuple. Let us consider a tree  $t$  that has an occurrence of the non-terminal  $A$  at position  $u$ . Then a rule of the form  $A \rightarrow s$  can rewrite a tuple  $(E, V, t, \nu, \varphi)$  into

$$(E', V', t', \nu', \varphi \wedge \psi(x'_0) \wedge \psi_k(x'_0, \dots, x'_n, y'_1, \dots, y'_m))$$

where:

- $t'$  is obtained from  $t$  by replacing the occurrence of  $A$  at position  $u$  by  $s$ ,
- $x'_0 = V(u)$ ,
- $x'_1, \dots, x'_n, y'_1, \dots, y'_m$  are fresh variables,
- $\nu'$  is the valuation that maps every variable  $x$  distinct from the  $x'_i$ 's ( $i \neq 0$ ) and the  $y'_j$ 's to  $\nu(x)$  and that maps each variable  $x'_i$ , with  $1 \leq i \leq n$ , to  $ul_i$  when  $l_i$  is the position of the node that is tagged with  $x_i$  in  $s$ ,
- $1 \leq k \leq p$ , is the index of the possible realization chosen for the rule; it corresponds to the choice of a formula  $\psi_k(x_0 \dots x_n, y_1 \dots y_m)$  and the corresponding realization  $M_k$ ,
- $V'$  is equal to  $V$  for positions different from  $ul_1, \dots, ul_n$  and  $V'(ul_i) = x'_i$  for  $1 \leq i \leq n$ ,
- $E'$  is  $E$  to which we add the rule  $x'_0 \rightarrow M'_k$  and where  $M'_k$  is obtained from  $M_k$  by respectively substituting  $x'_1, \dots, x'_n, y'_1, \dots, y'_m$  for  $x_1, \dots, x_n, y_1, \dots, y_m$ .

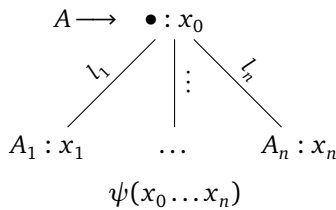


Figure 6:  
Constrained production  
with its associated  
linearization rule

$$\begin{array}{ccc}
 \psi_1(x_0 \dots x_n, y_1 \dots y_m) & \longrightarrow & M_1 \\
 x_0 := & \quad \vdots & \quad \vdots \\
 & \quad \vdots & \quad \vdots \\
 \psi_p(x_0 \dots x_n, y_1 \dots y_m) & \longrightarrow & M_p
 \end{array}$$

(Where the free variables in  $M_1 \dots M_p$  are  $x_1 \dots x_n y_1 \dots y_m$ .)

A rule  $A \rightarrow w$  that rewrites a non-terminal into a lexical entry  $w$ , assuming that this lexical entry is associated with the realization  $\bar{w}$ , can rewrite  $(E, V, t, \nu, \varphi)$  into  $(E', V, t', \nu, \varphi)$  where:

- $t'$  is obtained by replacing the occurrence of the non-terminal  $A$  at the position  $u$  by  $w$ ,
- $E'$  is the grammar obtained from  $E$  by adding the rule  $x \rightarrow \bar{w}$  if  $V(u) = x$ .

A complete derivation is a derivation such that  $(\emptyset, (\varepsilon, x), S, \emptyset, true)$  rewrites in several steps into  $(E, V, t, \nu, \varphi)$ , where  $t$  does not contain any occurrence of a non-terminal. The rules of  $E$  define a unique term, but this term may be not well-typed, or it may contain some free variables due to the external variables of some rule used in the course of the derivation. In the first case, we consider the derivation as invalid, in the second case, we need to give a meaning to the free variables of the term defined by  $E$ .

For this, we need a valuation  $\nu'$  that extends  $\nu$  by mapping the free variables of  $\varphi$  for which  $\nu$  is undefined to  $\nu(\text{dom}(E))$ , where  $\text{dom}(E)$  is the set of variables that are non-terminals of  $E$ . Thus  $\nu'$  extends  $\nu$  by mapping the external variables to nodes of  $t$  which are related to some node in the parse tree implicitly represented by  $E$ . Moreover,  $\nu'$  need to be such that  $t, \nu' \models \varphi$ . Notice that, in particular, this forces  $t$  to be a valid tree of the underlying regular grammar of abstract structures, as  $\varphi$  contains as one of its conjuncts the formula that  $t$  should satisfy in order to be valid. Now that  $\nu'$  is given, we may assign a semantics to the free variables in the term defined by  $E$ . For this, we follow our convention, by associating with an external variable  $y$  the realization of the node  $\nu(y)$ . Technically, it suffices to remark that, as  $E$  is a deterministic grammar, its rules induce a partial order on the non-terminals of  $E$ . Using this partial order, we replace each occurrence of a parameter  $y$  by the maximal non-terminal  $x$  in  $E$  so that  $\nu'(x) = \nu'(y)$ . If we do this for each parameter  $y$ , we obtain a deterministic grammar  $E'$ ; if this grammar defines a finite well-typed term, this term must be unique and we call it *realization* of  $(E, V, t, \nu, \varphi)$ . Nevertheless,  $E'$  may not define any finite term due to circular definitions, and it may also define badly typed terms.

Thus, in a nutshell, given the result  $(E, V, t, \nu, \varphi)$  of a complete derivation, its set of *realizations*  $M$  is given by every extension  $\nu'$  so that

$t, \nu' \models \varphi$ , so that the grammar  $E'$  that  $\nu'$  induces defines the well-typed term  $M'$ , whose normal form is  $M$ . Notice that, with this definition, a non-valid tree has an empty set of realizations.

More abstractly, we may see the linearization process as a Monadic Second-Order transduction (MSO-transduction) in the sense of Courcelle (1994) that turns the parse tree of a valid tree into a Directed Acyclic Graph (DAG) whose nodes are labelled with  $\lambda$ -terms. Then we take the unfolding of this DAG into a tree which can then be seen as a  $\lambda$ -term. The final step consists in  $\beta$ -normalizing this term provided it is well-typed. Courcelle and Engelfriet (1995) (see also Courcelle and Engelfriet 2012 for a more recent presentation of that result) showed that the class of languages definable with Hyperedge Replacement Grammars (HRG) is closed under MSO-transductions. As regular languages can easily be represented as HRGs, this shows that the language of DAGs output by the linearization process is definable by a Hyperedge Replacement Grammar (HRGs). Moreover, as shown by Engelfriet and Heyker (1992), the tree languages that are unfoldings of DAG languages definable with HRGs are output languages of attribute grammars, which in turn can be seen as almost linear 2-ACGs as showed by Kanazawa (2011, 2009). Thus, taking another homomorphism yields in general a non-linear 2-ACG. Nevertheless, when modelling the phonological realizations, we expect that the language we obtain is a linear 2-ACG. It is worthwhile to notice that the acyclicity of a graph is a definable property in MSO, and also that the well-typing of a tree labelled by  $\lambda$ -terms is MSO definable as well. Moreover, the translation of the linearization rules into a 2-ACG is such that the abstract syntactic structure can be read from the derivation trees of the 2-ACG we obtain with a simple relabelling. Indeed, when we showed how to construct a regular grammar recognizing the set of valid abstract structures, the abstract language of the 2-ACG was obtained by enriching the derivation trees of the regular grammar with information about the states of the automata corresponding to the logical formula or to the typing constraints. Therefore, the regular grammar with constraints and linearization rules can be effectively compiled into a 2-ACG. It is worthwhile to notice that the compiled grammar may be much larger than the original description.

## Handling optional arguments

We have proposed earlier to use  $(A)$  in regular tree productions to denote that an argument  $A$  is optional. Such an optional argument is then taken as a non-terminal that can rewrite into either  $A$  or  $\perp$ . When defining a linearization, the realization of  $(A)$  is taken to be that of the symbol that it rewrites to. We will set some default value as the realization of  $\perp$ , so that the realization of an empty argument is well defined.

For instance, in the example described by Figure 5, a sensible value for  $\perp$  is the empty string  $\varepsilon$ . With this default value, we may simply have one linearization rule  $c := true \longrightarrow s \vee o$ . Thus, when the verb is intransitive, the constraints we have put on the valid trees imply that the *obj* argument in the tree must be  $\perp$ . Taking the empty string as a default value, we get  $s \vee o = s \vee$ , which is the expected realization of an intransitive verb clause.

We can choose to provide each optional argument with its own default values, depending on what we consider a sensible realization of an empty optional argument. We could also conceivably associate a value with  $\perp$  that depends on its context in a stronger way, by using logical formulae. However, we have not found it useful in the models we have worked on; therefore, the linearization of optional arguments will only be  $\varepsilon$  for phonological linearizations, and a semantically empty argument when linearizing towards a sentence meaning representation.

## Additional macro syntax

The main goal of our approach is concision. To avoid redundancy in linearization rules, we introduce a syntactic mechanism that factors out some redundant constructions. Indeed, the number of linearization rules depend on the number of syntactic situations that may have an influence on the form the linearization takes. This number can be rather high, and just listing the situations may give the impression that one misses obvious generalizations, or structural dependencies between various cases.

The mechanism we propose to cope with this problem tries to give more structure to linearization rules. Mainly this mechanism is a form of simply typed  $\lambda$ -calculus designed to manipulate the linearization

rules. This calculus is parametrized with a finite set of variables  $X$  of the syntactic logical language. We write  $FO[X]$  as a shorthand for the set of logical formulae whose set of free variables is included in  $X$ . The set of types of the language is divided into two disjoint sets: *otypes* and  $\omega$ *types*. The set *otypes* is the set of simple types of the object language (i.e. the target language of the linearization) and  $\omega$ *types* are types of the form  $A_1 \rightarrow \dots \rightarrow A_n \rightarrow \omega$ , where  $\omega$  is an atomic type that is distinct from the atomic types used in *otypes* and  $A_1, \dots, A_n$  are *otypes*. As we have seen, one of the features of the linearization mechanism is that, depending on the context, the type of the realization may vary. Thus, in general, linearization rules are objects of the form  $[\varphi_1 \longrightarrow M_1, \dots, \varphi_n \longrightarrow M_n]$ , where  $M_1, \dots, M_n$  are terms that may have different types. The atomic type  $\omega$  is meant to type these objects.

The set of terms of the language,  $\mathcal{T}_{A,X}$ , is thus indexed with two parameters:  $A$  is either an *otypes* or an  $\omega$ *types* and  $X$  is the set of variables that are allowed to be free in the logical formula. The sets  $\mathcal{T}_{A,X}$  are inductively defined as follows:

- for  $A$  in *otypes*,  $x^A$  is in  $\mathcal{T}_{A,X}$ ,
- if  $M$  and  $N$  are respectively in  $\mathcal{T}_{A \rightarrow B,X}$  and in  $\mathcal{T}_{A,X}$ , then  $MN$  is in  $\mathcal{T}_{B,X}$ ,
- if  $M$  is in  $\mathcal{T}_{B,X}$  and  $A$  is in *otypes* then  $\lambda x^A.M$  is in  $\mathcal{T}_{A \rightarrow B,X}$ ,
- if  $M_1, \dots, M_n$  are all in  $\mathcal{T}_{A,X}$  with  $A$  in *otypes* and  $\varphi_1, \dots, \varphi_n$  are all in  $FO[X]$ , then  $[\varphi_1 \longrightarrow M_1, \dots, \varphi_n \longrightarrow M_n]$  is in  $\mathcal{T}_{A,X}$ ,
- if  $M_1, \dots, M_n$  are respectively in  $\mathcal{T}_{A_1,X}, \dots, \mathcal{T}_{A_n,X}$  with the  $A_1, \dots, A_n$  either being equal to  $\omega$  or being in *otypes*, then and  $\varphi_1, \dots, \varphi_n$  are all in  $FO[X]$ , then  $[\varphi_1 \longrightarrow M_1, \dots, \varphi_n \longrightarrow M_n]$  is in  $\mathcal{T}_{\omega,X}$ .

We adopt a call-by-value operational semantics for this language: as for IO languages (see Kobele and Salvati 2013), the notion of value coincides with that of the normal form. For this we need the notion of *pure terms*, that is  $\lambda$ -terms which contain no construct of the form  $[\varphi_1 \longrightarrow M_1, \dots, \varphi_n \longrightarrow M_n]$ . We shall denote pure terms in  $\beta$ -normal form with  $V$ , possibly with indices. A term is said to be in the normal form either when it is a pure term in the  $\beta$ -normal form, or when it is a term of the form  $[\varphi_1 \longrightarrow M_1, \dots, \varphi_n \longrightarrow M_n]$ , where  $M_1, \dots, M_n$  are pure terms in the  $\beta$ -normal form. From now on we shall write  $W$ , possibly with indices, for terms that are values, that is terms in normal

form. The computational rules of the calculus are as follows ( $M[V/x]$  denotes the capture-avoiding substitution of  $V$  for the free occurrences of  $x$  in  $M$ ):

- $(\lambda x.M)V \rightarrow M[V/x]$ ,
- $(\lambda x.M)[\varphi_1 \rightarrow V_1, \dots, \varphi_n \rightarrow V_n] \rightarrow$   
 $[\varphi_1 \rightarrow M[V_1/x], \dots, \varphi_n \rightarrow M[V_n/x]]$ ,
- $[\varphi_1 \rightarrow M_1, \dots, \varphi_n \rightarrow M_n]W \rightarrow [\varphi_1 \rightarrow M_1W, \dots, \varphi_n \rightarrow M_nW]$ ,
- $\lambda x.[\varphi_1 \rightarrow V_1, \dots, \varphi_n \rightarrow V_n] \rightarrow [\varphi_1 \rightarrow \lambda x.V_1, \dots, \varphi_n \rightarrow \lambda x.V_n]$ ,
- $V[\varphi_1 \rightarrow V_1, \dots, \varphi_n \rightarrow V_n] \rightarrow [\varphi_1 \rightarrow VV_1, \dots, \varphi_n \rightarrow VV_n]$  when  
 $V$  is not a  $\lambda$ -abstraction,
- $[\varphi_1 \rightarrow M_1, \dots, \varphi_k \rightarrow [\psi_1 \rightarrow V_1, \dots, \psi_n \rightarrow V_n],$   
 $\dots, \varphi_m \rightarrow M_m] \rightarrow$   
 $[\varphi_1 \rightarrow M_1, \dots, \varphi_k \wedge \psi_1 \rightarrow V_1, \dots, \varphi_k \wedge \psi_n \rightarrow V_n, \dots, \varphi_m \rightarrow M_m]$ .

The strong normalization of the simply typed  $\lambda$ -calculus induces the fact that computations in that calculus are terminating. The *subject reduction property* (the fact that the types of terms are invariant under reduction) is also inherited from the simply typed  $\lambda$ -calculus. We adopt in general a *right-most* reduction strategy which consists in rewriting the redex that is at the furthest right position in the term. This implements a call-by-value semantics for this language.

Finally, in the rest of the paper, we shall adopt some slight variation on the syntax of the language. In particular, we shall omit the typing annotations most of the time. We shall also write structures like  $[\varphi_1 \rightarrow M_1, \dots, \varphi_n \rightarrow M_n]$  as column vectors in which we omit the ‘,’ comma separator. We may also omit the square brackets or only put the left one to lighten the notation. We may write  $M$  where  $x_1 = M_1$  and ... and  $x_n = M_n$  in place of  $(\lambda x_1 \dots x_n.M)M_1 \dots M_n$ . Another abbreviation consists in simply writing  $M$  for  $[true \rightarrow M]$ . Finally when we write  $[\varphi_1 \rightarrow M_1, \dots, \varphi_n \rightarrow M_n, else \rightarrow M]$  we mean  $[\varphi_1 \rightarrow M_1, \dots, \varphi_n \rightarrow M_n, \neg\varphi_1 \wedge \dots \wedge \neg\varphi_n \rightarrow M]$ . Examples of this notation are used all along the next section.

## ILLUSTRATION

### 3.1

#### *Synchronous grammar*

We now illustrate the formalism we have introduced in the previous section by constructing a more complex grammar. This grammar will provide a superficial cover of several overlapping phenomena. It covers verbal clauses with subject, direct object, and complement clause arguments, taking into account verb valency. It also includes subject and object control verbs, and modification of noun phrases by relative clauses, with a *wh*-movement account of relative pronouns that takes island constraints into account. It also models a simplistic case of agreement that only restricts the use of the relative pronoun *that* to neuter antecedent. Linearization rules are provided that produce phonological realizations for English, German, and Dutch, in order to demonstrate the possibility of parametrizing the word order of realizations (including cross-serial ordering). Another set of linearization rules produces Montague-style  $\lambda$ -terms that represent the meaning of the covered sentences. Even though we have chosen our example so as to avoid a complete coverage of agreement, we hope that the treatment of *that* is illustrative enough to give a flavor of its rather straightforward extension to a realistic model of agreement.

Our goal when designing this grammar is to confront the methodology described so far against the task of dealing with the modeling of several interacting phenomena, along with both their syntactic and semantic linearizations, and evaluate the results in terms of expressiveness as well as concision.

#### Vocabulary

First, we construct a vocabulary for our grammar. The part-of-speech properties we use are:

*proper\_noun, noun, pronoun, determiner, verb.*

The other lexical properties are:

*pro\_rel, transitive, ctr\_subj, ctr\_obj, infinitive, masculine, feminine, neuter*

and designate respectively: relative pronouns, transitive verbs, subject control, and object control verbs, verbs in infinitive form, and gender marking.

Table 2:  
Excerpt  
vocabulary

| English | German | Dutch  | Semantic type                                 | Properties                                   |
|---------|--------|--------|-----------------------------------------------|----------------------------------------------|
| lets    | lässt  | laat   | $e \rightarrow e \rightarrow p \rightarrow p$ | <i>verb; ctr_obj; transitive</i>             |
| help    | helpen | helpen | $e \rightarrow e \rightarrow p \rightarrow p$ | <i>verb; ctr_obj; transitive; infinitive</i> |
| want    | willen | wilen  | $e \rightarrow p \rightarrow p$               | <i>verb; ctr_subj; infinitive</i>            |
| read    | lesen  | lezen  | $e \rightarrow e \rightarrow p$               | <i>verb; transitive; infinitive</i>          |
| that    | das    | dat    | $(e \rightarrow p) \rightarrow p$             | <i>pronoun; pro_rel; neuter</i>              |
| a       | ein    | een    | $(e \rightarrow p) \rightarrow p$             | <i>determiner; neuter</i>                    |
| book    | Buch   | boek   | $e \rightarrow p$                             | <i>noun; neuter</i>                          |
| John    | Hans   | Jan    | $e$                                           | <i>proper_noun; masculine</i>                |
| Mary    | Marie  | Marie  | $e$                                           | <i>proper_noun; feminine</i>                 |
| Ann     | Anna   | Anna   | $e$                                           | <i>proper_noun; feminine</i>                 |

The vocabulary is summed up in Table 2. The table also gives the expected phonological realizations of the individual lexical entries for English, German, and Dutch, along with the type of their semantic realization. Semantic types are based on  $e$  and  $p$ , which denote entities and propositions (truth values), respectively. Abstract structure leaves that are lexical entries will be written as their associated English realizations; and so will their semantic realizations, with the same type-setting conventions we used previously: e.g. *song* (abstract structure leaf) vs. *song* (semantic realization).

#### Regular over-approximation

We now present the regular grammar that over-approximates the set of valid abstract structures. It contains three non-terminal symbols  $C$ ,  $A$ , and  $M$ . The start symbol  $C$  corresponds to independent or subordinate clauses,  $A$  to noun phrases that are an argument of some clause, and  $M$  to modifiers.

The labels used on the edges of abstract structures belong to the list (*head, subj, obj, arg cl, det, mod*) and designate respectively the head of the (nominal or verbal) phrase, the nominal subject, the nominal direct object, an additional complement clause of the verbal predicate, the determiner in a noun phrase, and a modifier.

The production rules of the grammar are given in Figure 7. The production  $p_1$  constructs a clause with a verb as its head, along with its (optional) arguments;  $p_2$  recursively adds a modifier to an argument;  $p_3$  through  $p_5$  build an argument as a noun phrase, respectively in the



A logical approach to grammar description

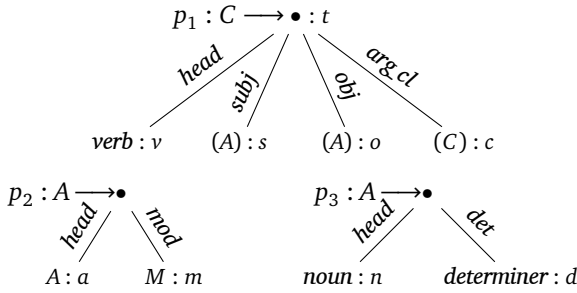


Figure 7:  
Regular over-  
approximation  
of valid  
sentences

$$p_4 : A \longrightarrow \text{proper\_noun} : pn \quad p_5 : A \longrightarrow \text{pronoun} : p \quad p_6 : M \longrightarrow C : r$$

form of a determiner/noun pair, a proper noun, and a pronoun; and finally  $p_6$  constructs a modifier as a verbal clause. Note that the only type of modifiers covered in the grammar are verbal clauses (which we shall restrict to be relative clauses), but other could be added by adding more productions that rewrite  $M$  as an adjective or a genitive construct.

Defining linguistic notions

We now use the logical language to construct predicates that model linguistic notions and relations. We shall use these relations both for constraining the regular grammar and to guide the linearization process. The predicates and relations we add are summed up in Table 3.

The first predicate recognizes a control verb which is simply a verb that has the subject control or object control lexical property:

$$\text{control\_verb}(v) := \text{verb}(v) \wedge (\text{ctr\_subj}(v) \vee \text{ctr\_obj}(v))$$

The second predicate defines a clause as a subtree whose head is a verb:

$$\text{clause}(cl) := \exists v. \text{verb}(v) \wedge \text{head}(cl, v)$$

Then, a controlled clause is a clause that serves as an argument of a control verb:

$$\begin{aligned} \text{controlled}(ctd) := & \text{clause}(ctd) \wedge \\ & \exists \text{ctr}. \text{control\_verb}(\text{ctr}) \wedge \text{head} \uparrow \text{arg\_cl}(\text{ctr}, \text{ctd}) \end{aligned}$$

We construct another predicate to identify verbs that expect an argument clause. In the context of our grammar, this is equivalent to

Table 3:  
Logical  
predicates  
modelling  
linguistic  
notions

$$\begin{aligned}
\text{control\_verb}(v) &:= \text{verb}(v) \wedge (\text{ctr\_subj}(v) \vee \text{ctr\_obj}(v)) \\
\text{clause}(vp) &:= \exists v. \text{verb}(v) \wedge \text{head}(vp, v) \\
\text{controlled}(ctd) &:= \text{clause}(ctd) \wedge \exists \text{ctr}. \text{control\_verb}(\text{ctr}) \wedge \text{head} \uparrow \text{arg\_cl}(\text{ctr}, \text{ctd}) \\
\text{clause\_verb}(v) &:= \text{control\_verb}(v) \\
\text{independent}(icl) &:= \text{clause}(icl) \wedge \forall \text{cl}. \text{clause}(\text{cl}) \Rightarrow \neg \text{any}^+(\text{cl}, icl) \\
\text{subordinate}(scl) &:= \text{clause}(scl) \wedge \exists p. (\text{mod} + \text{arg\_cl})(p, scl) \\
\text{relative}(rcl) &:= \text{subordinate}(rcl) \wedge \\
&\quad \exists \text{hd}. (\text{noun}(\text{hd}) \vee \text{proper\_noun}(\text{hd})) \wedge \text{head}^* \uparrow \text{mod}(\text{hd}, rcl) \\
\text{ext\_path}(cl, p) &:= (\text{subj} + \text{arg\_cl}^* \text{obj})(cl, p) \\
\text{ext\_obj}(obj) &:= \text{pro\_rel}(obj) \wedge \exists \text{cl}. \text{obj}(\text{cl}, obj) \\
\text{ext\_suj}(suj) &:= \text{pro\_rel}(suj) \wedge \exists \text{cl}. \text{subj}(\text{cl}, suj) \\
\text{ext\_cl}(cl) &:= \exists p. \text{ext\_path}(cl, p) \wedge \text{pro\_rel}(p) \\
\text{gd\_agr}(x, y) &:= (\text{masculine}(x) \wedge \text{masculine}(y)) \\
&\quad \vee (\text{feminine}(x) \wedge \text{feminine}(y)) \\
&\quad \vee (\text{neuter}(x) \wedge \text{neuter}(y)) \\
\text{antecedent}(\text{ant}, \text{pro}) &:= (\text{noun}(\text{ant}) \vee \text{proper\_noun}(\text{ant})) \wedge \text{pro\_rel}(\text{pro}) \\
&\quad \wedge \exists \text{rcl}. \text{relative}(\text{rcl}) \wedge \text{head}^* \uparrow \text{mod}(\text{ant}, \text{rcl}) \\
&\quad \wedge \text{ext\_path}(\text{rcl}, \text{pro})
\end{aligned}$$

the verb being a (subject or object) control verb:

$$\text{clause\_verb}(v) := \text{control\_verb}(v)$$

This predicate could be extended to include verbs that expect other forms of complement clauses besides the infinitival clauses associated with control verbs.

The following predicates enable us to distinguish between different types of clauses.

First, an independent clause is a clause that is not dominated (through any non-empty sequence of edges) by any other clause:

$$\text{independent}(icl) := \text{clause}(icl) \wedge \forall \text{cl}. \text{clause}(\text{cl}) \Rightarrow \neg \text{any}^+(\text{cl}, icl)$$

By contrast, a subordinate clause is a clause that serves as a complement or modifier:

$$\text{subordinate}(scl) := \text{clause}(scl) \wedge \exists p. (\text{mod} + \text{arg\_cl})(p, scl)$$

Then, a relative clause is a subordinate clause that modifies a noun phrase (a subtree whose head is a common or proper noun):

$$\begin{aligned} \text{relative}(rcl) &:= \text{subordinate}(rcl) \wedge \\ &\quad \exists hd. (\text{noun}(hd) \vee \text{proper\_noun}(hd)) \wedge \\ &\quad \text{head}^* \uparrow \text{mod}(hd, rcl) \end{aligned}$$

We then add a predicate to identify objects that undergo a wh-movement (which we call *extracted*). This covers all relative pronouns that fill an object role in a clause. We also provide a similar predicate for extracted subjects, following the usual analysis of generative grammars (Chomsky 1981):

$$\begin{aligned} \text{ext\_obj}(obj) &:= \text{pro\_rel}(obj) \wedge \exists cl. \text{obj}(cl, obj) \\ \text{ext\_suj}(suj) &:= \text{pro\_rel}(suj) \wedge \exists cl. \text{subj}(cl, suj) \end{aligned}$$

Next, we add a relation that links an *insertion site* and its corresponding *extraction site*, taking into account island constraints as defined by Ross (1967). We recall that, in the generative tradition, the extraction site is the position at which a wh-word would be realized given its syntactic role according to the canonical word order of a language. By contrast, the insertion site corresponds to its actual position in the sentence. In our grammar, complying with island constraints means that only *arg cl* edges are allowed for traversal before we reach a distant extracted object:

$$\text{ext\_path}(cl, p) := (\text{subj} + \text{arg } cl^* \text{ obj})(cl, p)$$

Using this relation, we define another predicate, which denotes that a complement clause contains an extraction of some form; this corresponds to the clause containing a relative pronoun at the end of a valid extraction path:

$$\text{ext\_cl}(cl) := \exists p. \text{ext\_path}(cl, p) \wedge \text{pro\_rel}(p)$$

Note that the straightforward definition of *ext\_path* we have given does not, purposefully, guarantee that the first position given is an insertion site, nor that the second one is an extraction site. It simply ensures that island constraints are not violated for long-distance extractions in the context of our grammar. However, since we are only going to use

it in contexts where both its arguments must satisfy the other prerequisites for wh-movement, this simple definition will be sufficient for our needs.

We add another relation that verifies gender agreement between two nodes. This relation is simply true if and only if both of the involved nodes have the same gender property:

$$\begin{aligned} gd\_agr(x, y) := & masculine(x) \wedge masculine(y) \\ & \vee feminine(x) \wedge feminine(y) \\ & \vee neuter(x) \wedge neuter(y) \end{aligned}$$

This relation could be extended so as to account for more agreement phenomena such as number, case, etc.

Finally, we define one last relation that links a relative pronoun to its antecedent. This relation is built upon *ext\_path* and links the head of a noun phrase to the relative pronoun of the relative clause that modifies it:

$$\begin{aligned} antecedent(ant, pro) := & (noun(ant) \vee proper\_noun(ant)) \wedge pro\_rel(pro) \\ & \wedge \exists rcl.relative(rcl) \wedge head^* \uparrow mod(ant, rcl) \wedge ext\_path(rcl, pro) \end{aligned}$$

This relation will allow us to verify that relative pronouns agree with their antecedents.

#### Logical constraints

In order to refine the over-approximation given in Figure 7, we now add logical constraints to production rules. We recall that only the abstract structures which satisfy these formulae are considered valid according to the grammar, thus filtering out many ill-formed structures.

We first consider  $p_1$ , for which four additional constraints are given in Figure 8.

Constraints (1) and (2) deal with verb valency, ensuring that the produced clause has an object if and only if the head is a transitive verb, and a complement clause if and only if the head is a verb that expects one. Constraint (3) equates the lack of an explicit subject argument with the fact that a clause is controlled. From a syntactic point of view, our model uses clauses without an explicit subject. We shall see later how logic allows us to associate its subject with a controlled clause. Finally, constraint (4) ensures that verbs in controlled clauses

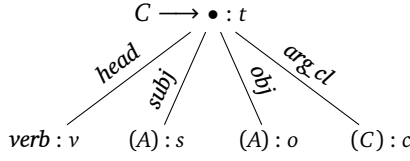


Figure 8:  
Logical constraints  
for  $p_1$

- $$\begin{aligned} \text{transitive}(v) &\Leftrightarrow \text{some}(o) & (1) \\ \text{clause\_verb}(v) &\Leftrightarrow \text{some}(c) & (2) \\ \text{controlled}(t) &\Leftrightarrow \text{none}(s) & (3) \\ \text{controlled}(t) &\Leftrightarrow \text{infinitive}(v) & (4) \end{aligned}$$

are in an infinitive form and, since the grammar does not handle other kinds of infinitive clauses, we assume that all infinitive verbs are controlled.

We now consider the relation between relative clauses and relative pronouns. We want to ensure that the valid abstract structures show a one-to-one relation between relative pronouns and the relative clauses they belong to. These pronouns should also be found at positions in their relative clause that are consistent with island constraints. This is guaranteed by a pair of symmetrical constraints on productions  $p_5$  and  $p_6$ :

$$p_5 : A \longrightarrow \text{pronoun} : p \qquad p_6 : M \longrightarrow C : r$$

$$\text{pro\_rel}(p) \Rightarrow \exists! r. \text{relative}(r) \wedge \text{ext\_path}(r, p) \qquad \exists! p. \text{pro\_rel}(p) \wedge \text{ext\_path}(r, p)$$

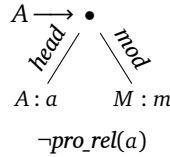
$$\begin{aligned} \text{pro\_rel}(p) &\Rightarrow \\ &\exists \text{ant}. \text{antecedent}(\text{ant}, p) \wedge \text{gd\_agr}(\text{ant}, p) \end{aligned}$$

The production  $p_5$  rewrites an argument  $A$  as a pronoun. The first constraint associated with it ensures that, if it is a relative pronoun, there is a unique relative clause to which this pronoun corresponds. Conversely,  $p_6$  produces a relative clause and ensures that there is a unique relative pronoun that corresponds to it. Both constraints use the  $\text{ext\_path}$  relation to make sure that the path between the top of the relative clause and its corresponding pronoun is valid and does not violate island constraints.

Then, the second constraint on  $p_5$  ensures that a relative pronoun has an antecedent, and that both of them are in agreement. With this, we rule out constructs like “Mary that ...” in English, “Marie das ...” in German or “Marie dat ...” in Dutch. This illustrates how agreement can be added to the grammar. However, as languages may have different sets of gender and agreement rules, when dealing with synchronous grammars, it is better to model agreement by refining a core common grammar for each target language. We here avoid this complication for the sake of keeping the illustration of the formalism rather simple. This is why we only give a simplistic treatment of the neuter gender that behaves similarly in English, Dutch, and German for the particular set of sentences we model.

Finally, one last syntactic restriction that we want to add is to forbid the addition of a modifier to a relative pronoun (which would be ungrammatical). The corresponding constraint is added to  $p_2$  as shown in Figure 9.

Figure 9:  
Logical constraint of  $p_2$



The added logical constraint simply ensures that a modified argument  $a$  cannot be a relative pronoun.

#### Phonological linearizations

Finally, we turn to the process of describing four linearizations (towards English, German and Dutch on one hand, and semantics on the other hand) of the grammar. We will begin with the phonological linearizations, starting with the most straightforward linearization rules, until we have covered all the productions in the grammar. A complete representation of the grammar, including production rules, logical constraints, and all the linearization rules, is available in Figures 17 and 18.

First, we consider the phonological linearization rules for productions  $p_2$  through  $p_5$ , which are given in Figure 10.

The linearization rules are the same for all target languages (i.e. English, German, and Dutch):  $p_4$  and  $p_5$ , being unary terminal rules,

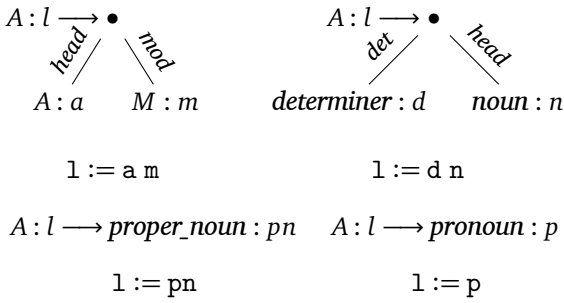


Figure 10:  
Phonological linearization  
rules for productions  $p_2$   
through  $p_5$

are simply realized with the string value associated with the lexical entry of the leaf they rewrote into. The linearization of productions  $p_2$  and  $p_3$  is obtained by concatenating the string values of the lexical entries in the expected order, which means that the determiner is followed by the noun for  $p_3$  and the whole noun phrase is followed by the relative clause for  $p_2$ .

We then turn to the linearization of  $p_6$ :

$$M : l \longrightarrow C : r$$

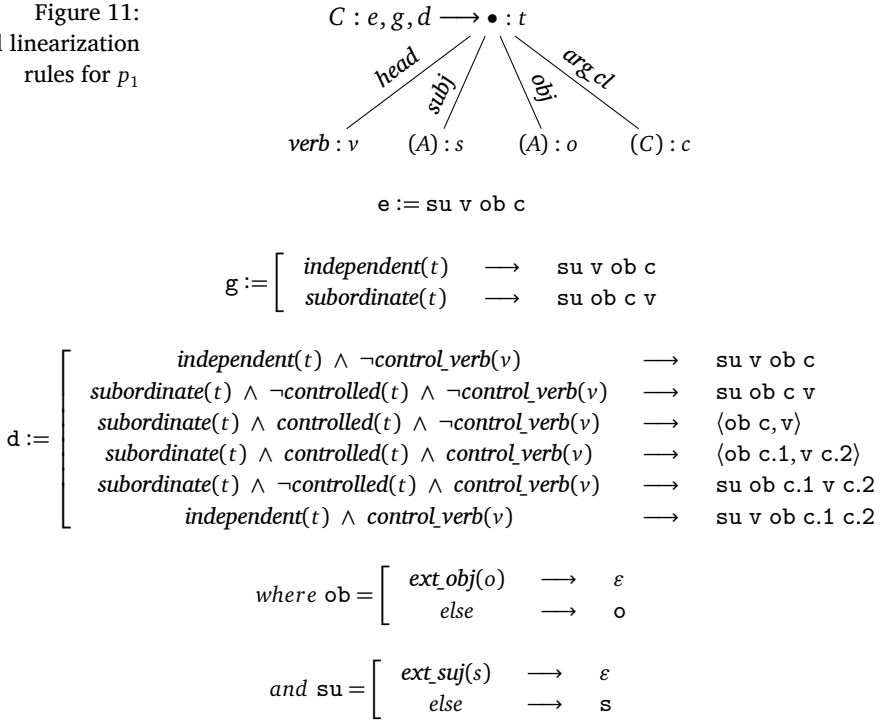
$$l := \text{ext\_path}(r, \mathbf{p}) \wedge \text{pro\_rel}(\mathbf{p}) \longrightarrow \mathbf{p} r$$

This production enables us to rewrite a modifier as a relative clause  $r$ . Once again, the linearization remains the same cross-linguistically. However, it uses an external variable  $\mathbf{p}$ , which corresponds to the relative pronoun of this relative clause. Let us recall that this grammar treats relative pronouns as *wh*-elements which appear in the abstract structure at the position corresponding to their syntactic function, which means they have to be phonologically realized at a distant position in the tree in order to precede the rest of the relative clause. The linearization rule thus calls for the realization of the (unique) relative pronoun  $\mathbf{p}$  which satisfies  $\text{ext\_path}(r, \mathbf{p})$ , and places it before the rest of the relative clause in the realization.

Note that the logical constraint given earlier for this production guarantees that such a pronoun does exist (and that it is unique) in all valid abstract structure trees. Hence, this linearization rule always produces a realization.

Finally, we consider the more sophisticated linearization rule associated with  $p_1$ , depicted in Figure 11.

Figure 11:  
Phonological linearization  
rules for  $p_1$



Having to deal with the linear ordering of the clause arguments, this production uses different linearization rules for each target language. We use different labels  $e, g, d$  for each one in the figure, which stand for English, German, and Dutch, respectively.

First, we take the linearization of empty optional leaves ( $\perp$ ) to be the empty string for all non-terminals.

Then, consider the final *where* statements, which are the same in all three linearizations (we have only written them once in order to clear up the figure). They describe two variable constructs ( $\text{su}, \text{ob}$ ), which are slightly more abstract versions of the arguments they correspond to ( $s, o$ ): these constructs denote the local realizations of the subject and object arguments which can be either the empty string  $\varepsilon$ , if the subject or object is a *wh*-pronoun marked for extraction, or simply the realization of the argument itself otherwise. We use this abstraction in place of the actual subject and object variables everywhere else in the linearization rules.



Now, the linearization rule for English simply concatenates the verb and its arguments in the usual SVO order, with the complement clause at the end.

The linearization rule for German, on the other hand, relies on the context to pick an appropriate word order: when the current clause is an independent clause, it uses the same SVO word order as English; however, if it is a subordinate clause, then the verb is rejected at the end, as expected in German sentences. Note that this linearization does not account for the scrambling phenomenon that occurs in German subordinate clauses. A possibility for modelling this phenomenon would be to define linearization in the algebra for free word orders proposed in Kirman and Salvati (2013).

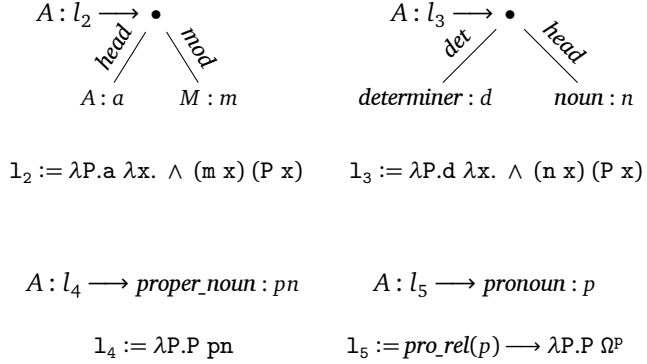
Finally, we turn to the more complex linearization rule for Dutch. The first two cases, which cover independent or subordinate clauses in which no control is involved, have the same realization as in German. The third logical clause builds a realization in the case of a subordinate clause which is controlled by its parent clause. Controlled clauses, rather than being realized as a string, are realized as a pair of strings so as to produce the expected cross-serial word order of Dutch. The first element of the pair accumulates object arguments, while the second one stacks the verbs. The next logical clause covers the case of a verb in a subordinate clause which exerts a control while being controlled itself; this is the “intermediate” step in cross-serial constructions. It builds up the stack of objects by concatenating its object argument before the first projection of the realization of its argument clause, and does the same for verbs on the second projection, producing a pair of strings similar to the one it received from its argument clause. Finally, the last two clauses of the linearization complete a cross-serial construct by concatenating both projections of the pair of strings they receive in the expected order, according to whether the topmost clause in the series is an independent or subordinate clause.

As a last remark, note that it is easily verified that the given set of linearization rules provides a linearization for all valid abstract structures.

#### Semantic linearization

We now turn to the semantic linearization rules. Let us recall the semantic types given in the vocabulary from Table 2. We work with

Figure 12:  
Semantic linearization  
rules for productions  $p_2$   
through  $p_5$



simple types built from the basic types  $e$ , which denotes entities, and  $p$  which denotes propositions. We add two constants  $\Omega^e$  and  $\Omega^p$  to the object language, which are empty semantic values for these base types. We follow a straightforward version of the usual Montague-style interpretations of syntactic categories. We take the semantic value of nouns and modifiers to have type  $e \rightarrow p$ , building a proposition from an entity. The interpretation of proper nouns simply refers directly to the entity they correspond to (type  $e$ ). Determiners have the type of a quantifier  $(e \rightarrow p) \rightarrow p$ . The type of verbs depends on the type and number of arguments they expect: they can be either  $e \rightarrow p$  (for intransitive verbs),  $e \rightarrow e \rightarrow p$  (transitive),  $e \rightarrow p \rightarrow p$  (expecting a complement clause), and  $e \rightarrow e \rightarrow p \rightarrow p$  (both transitive and expecting a complement clause). Finally, the type of a clause can be either  $p$  for an independent clause,  $e \rightarrow p$  for controlled clauses,  $e \rightarrow p$  for clauses on an extraction path, or  $e \rightarrow e \rightarrow p$  for clauses both controlled and on an extraction path. These four types account for all the possible cases of missing arguments; including a missing subject (in the case of a control or a subject relative clause), or a possibly distant object (in the case of an object relative clause). These abstracted arguments will be provided either by the parent (controlling) clause or the antecedent of the relative clause. Finally, we take the convention that the upper-case letter variables bound in our semantic  $\lambda$ -terms have type  $e \rightarrow p$ , while lower-case letter ones have type  $e$ .

We consider first the linearization rules that produce an argument, that is those depicted in Figure 12. The left-hand side of each production  $p_i$  in the figure is labelled with  $l_i$  to help identify them.

First we consider production  $p_4$ : its linearization constructs an argument from an element by abstracting the predicate to which it will eventually be applied, using the type-raising construction usual in Montague semantics. Production  $p_5$ , when realizing a relative pronoun, produces an “empty” argument with no semantic value. As the corresponding argument in the clause will have to be linked with the relative’s antecedent, the corresponding term will be deleted during the linearization of the clause that dominates  $l_5$ . Finally, productions  $p_2$  and  $p_3$  construct the semantics of noun-phrase using the continuation passing style that is usual in Montague semantics.

Next, we consider the linearization of  $p_6$ , given below:

$$M : l \longrightarrow C : r$$

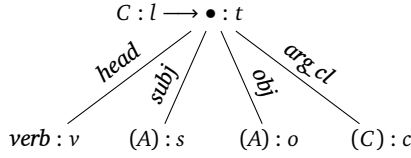
$$l := r$$

This linearization rule builds a modifier from a clause. Since a relative clause must contain exactly one extracted pronoun and cannot be controlled (as it is not an argument of a controller verb), the resulting realization has the type of a predicate. This element will later be used to modify the meaning of its antecedent with the rule  $p_2$ .

Last, we consider the linearization rule for  $p_1$ , given in Figure 13. This linearization rule takes into account all the possible contexts and arguments of a clause, and builds a realization accordingly. It relies on the reification of several concepts that are described separately using *where* statements, then plugged together as needed to build the realization. These constructs are denoted by the variables  $su, ob, cl$ ; in addition, the variable  $fcl$  is used to avoid repetitions by factorizing a major part of the term that constitutes the final realization.

First, the variables  $su$  and  $ob$  correspond respectively to the subject and object type-raised arguments of the clause. In most cases, these arguments are simply the realization of the *subj* and *obj* arguments of the clause, that is  $s, o$ . However, either of them may actually be a relative pronoun, and the corresponding element should be left abstract, so as to allow the antecedent to use the relative clause as a modifier. In such cases (*ext\_suj, ext\_obj*), the actual subject or object element is left as a free variable  $e$ ; and  $su$  or  $ob$  are obtained with a simple type-raising construction, thus behaving exactly as a normal subject or object argument would. Finally, it may also be that the *subj*

Figure 13:  
Semantic linearization  
rule for  $p_1$



$$\begin{aligned}
 l := & \begin{cases} \neg \text{ext\_cl}(t) \rightarrow \begin{cases} \neg \text{controlled}(t) \rightarrow \text{fcl } \Omega^e \Omega^e \\ \text{controlled}(t) \rightarrow \lambda s'. \text{fcl } s' \Omega^e \end{cases} \\ \text{ext\_cl}(t) \rightarrow \begin{cases} \neg \text{controlled}(t) \rightarrow \lambda e. \text{fcl } \Omega^e e \\ \text{controlled}(t) \rightarrow \lambda s'. \lambda e. \text{fcl } s' e \end{cases} \end{cases} \\
 \text{where } \text{fcl} = \lambda s'. \lambda e. \text{su } \lambda x. \text{ob } \lambda y. & \begin{cases} \neg \text{transitive}(v) \rightarrow \begin{cases} \neg \text{clause\_verb}(t) \rightarrow v \ x \\ \text{clause\_verb}(t) \rightarrow v \ x \ \text{cl} \end{cases} \\ \text{transitive}(v) \rightarrow \begin{cases} \neg \text{clause\_verb}(t) \rightarrow v \ x \ y \\ \text{clause\_verb}(t) \rightarrow v \ x \ y \ \text{cl} \end{cases} \end{cases} \\
 \text{where } \text{cl} = & \begin{cases} \text{ext\_cl}(c) \rightarrow \begin{cases} \text{ctr\_subj}(v) \rightarrow c \ x \ e \\ \text{ctr\_obj}(v) \rightarrow c \ y \ e \\ \text{else} \rightarrow c \ e \end{cases} \\ \text{else} \rightarrow \begin{cases} \text{ctr\_subj}(v) \rightarrow c \ x \\ \text{ctr\_obj}(v) \rightarrow c \ y \\ \text{else} \rightarrow c \end{cases} \end{cases} \\
 \text{where } \text{su} = & \begin{cases} \text{ext\_subj}(s) \rightarrow \lambda P.P \ e \\ \text{controlled}(t) \rightarrow \lambda P.P \ s' \\ \text{else} \rightarrow s \end{cases} \\
 \text{and } \text{ob} = & \begin{cases} \text{ext\_obj}(o) \rightarrow \lambda P.P \ e \\ \text{else} \rightarrow o \end{cases}
 \end{aligned}$$

argument is non-existent when the current clause is controlled. The construct in this case is the same as for an extracted subject, except that the free variable corresponding to a controlled subject is, by convention,  $s'$  instead of  $e$ .

Then, the  $cl$  construct denotes the clause argument of a verb that requires one (*clause\_verb*). Such verbs expect an argument of type  $p$ , corresponding to a proposition; however, as we have seen, a clause may have a more abstract type, with one or two missing elements. There are two independent reasons for a subordinate clause to expect an element. The first one is if the clause is on an extraction path (*ext\_cl*). If this is the case, the expected element corresponds to the missing object at the end of the extraction path, and it is provided

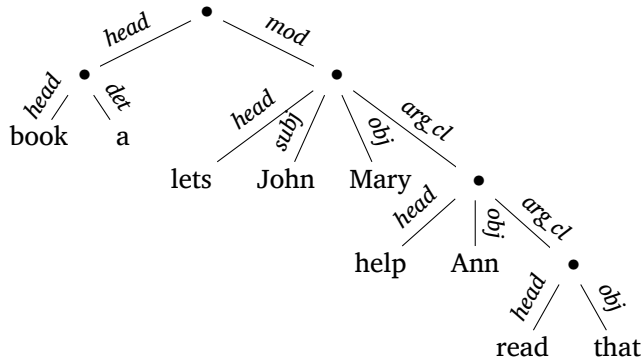
to the argument clause in the form of a free variable  $e$ . The second reason for subordinate clauses to expect an argument is control (*ctr\_subj, ctr\_obj*): indeed, a controlled clause has a missing subject, which is identified with the subject or object of the controller clause. Depending on the type of control, the missing element is supplied as either  $x$  or  $y$ , which are free variables denoting respectively the subject or direct object element of the verb in the current (controlling) clause. Finally, note that the construction of *cl* also covers the case where an argument clause is present without the head of the current clause being a control verb; this cannot currently be the case in our valid abstract structures, as we have currently defined that the verbs which expect a complement clause (*clause\_verb*) are exactly the control verbs. Nevertheless, we decided to add a default rule (using *else*) for the sake of completeness. Should we decide to introduce verbs that expect other forms of complement clauses in our vocabulary and alter our definition of *clause\_verb*, this linearization rule would yield the expected semantics for the corresponding sentences.

Using these three constructs, we can now build *fcl*, that is the (factorized)  $\lambda$ -term that represents the meaning of the clause. This term abstracts the free variables  $s'$  and  $e$  that denote a missing subject (in the case of a control) and an extracted element (in the case of a relative clause), regardless of whether or not they occur in the term. The *su* and *ob* constructs are then applied to the verb cluster, with abstracted variables  $x$  and  $y$  for the subject and object elements; these constructs behave exactly as normal, type-raised arguments. The verb cluster itself is constructed according to the valency properties of the verb (*transitive, clause\_verb*), by applying the verb to its arguments  $x$ ,  $y$ , and *cl*.

Finally, the linearization of the whole clause, depending on whether there is an ongoing extraction or control (*ext\_cl, controlled*), provides empty elements to *fcl*, or abstracts the corresponding variables again in order to yield the expected type for the realization. The logical preconditions ensure that the empty elements  $\Omega^e$  are provided exactly when the term *fcl* does not depend on that argument. We thus obtain the intended meaning of a clause.

The semantic linearization of a clause may, at first sight, look rather involved. However, it still represents a shorter representation over the mere enumeration of all the possible cases covered by the

Figure 14:  
Abstract structure *abs*



linearization. The grammar indeed covers 39 different cases (taking into account the interactions between valency, relevant classes of context, control, and extraction). Compared to a direct implementation of all the cases with an actual grammar, this model is arguably simple. Moreover, the abstraction provided by logic makes the model rather intuitive.

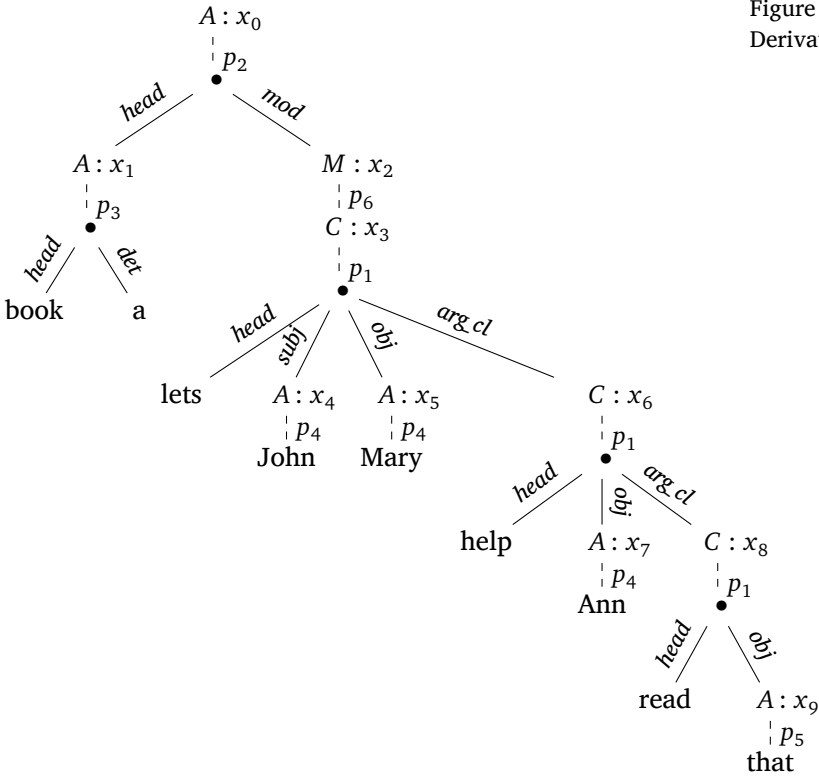
3.2 *Example*

We now construct an example sentence for the synchronous grammar we have just described, and show how the grammar asserts its grammaticality and assigns it a realization according to the linearization rules.

In order to demonstrate the interaction between the different phenomena covered by the grammar, we consider a “worst-case” example phrase that exhibits long-distance movement in a relative clause across a sequence of control verbs. Though its acceptability may be questionable, it should serve as a good support for describing the inner works of our formalism.

The abstract structure *abs* of the example we consider is depicted in Figure 14. It consists of an argument formed by a common noun and a determiner modified by a relative clause where the relative object pronoun is reached across two nested subordinate clauses. The antecedent (*a book*) is thus identified with the object of the last controlled verb (*read*). The subject of this verb is provided by the object control verb above (*help*), which is itself controlled by the head of the relative clause above (*lets*).

Figure 15:  
Derivation tree of *abs*



For the purpose of the example we consider that the starting symbol of the regular over-approximation is *A* instead of *C*. Indeed, all the interesting phenomena we wish to illustrate in the example can arise in noun-phrases and embedding this noun-phrase example in a complete sentence would only lengthen our explanations with unnecessary details.

Regular tree grammar derivability

First, we will show that *abs* is in the language of the regular over-approximation of our synchronous grammar. The corresponding derivation is depicted in Figure 15. The non-terminals are written as (labelled) nodes in the derivation tree, and rewrites are represented as dashed edges, labelled with the name of the production used to rewrite the non-terminal. The corresponding right-hand side is then drawn directly below, without the unused optional nodes. We recall that the

full definition of our synchronous grammar is summed up in Figures 17 and 18.

Note that, though there is only one derivation for this abstract structure *abs*, the regular over-approximation of our grammars need not be unambiguous. Had there been several different derivations that produced the given abstract structure, we would have considered all of them.

Satisfaction of logical constraints

To verify that the tree *abs* is valid, we need to ensure that, in addition to the regular over-approximation, the abstract structure tree also satisfies the logical constraints associated with the productions. Traversing the *abs* tree in prefix order, we consider the logical conditions associated with each production and instantiate them with the corresponding positions in the tree.

The first production used in the derivation of *abs* is  $p_2$ . It has an associated logical constraint, stated as:

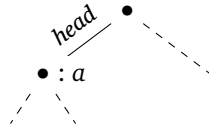
$$\neg pro\_rel(a)$$

This constraint ensures that the modified noun phrase is not just a relative pronoun, with *a* being the *head* argument of the right-hand side of the production. As can be seen in Figure 16, the node corresponding to *a* is not a lexical entry and hence cannot have the *pro\_rel* property, so the constraint is satisfied, and the structure remains grammatical.

The production  $p_3$  on the left branch has no associated constraints, so it is trivially valid. On the right branch, on the other hand, the first production used in the rewrite is  $p_6$ , which expects that there exists a unique relative pronoun *p* at the end of a valid extraction path starting at the current position *r*:

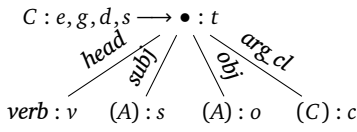
$$\exists ! p. pro\_rel(p) \wedge ext\_path(r, p)$$

Figure 16:  
Instantiation of the logical  
constraint for  $p_2$





A logical approach to grammar description



$$\begin{aligned}
 \text{transitive}(v) &\Leftrightarrow \text{some}(o) \\
 \text{clause\_verb}(v) &\Leftrightarrow \text{some}(c) \\
 \text{controlled}(t) &\Leftrightarrow \text{none}(s) \\
 \text{controlled}(t) &\Leftrightarrow \text{infinitive}(v)
 \end{aligned}$$

Figure 17:  
First part  
of the full  
synchronous  
grammar,  
with logical  
constraints and  
linearizations

$$e := su \ v \ ob \ c$$

$$g := \begin{cases} \text{independent}(t) & \longrightarrow \text{su } v \ ob \ c \\ \text{subordinate}(t) & \longrightarrow \text{su } ob \ c \ v \end{cases}$$

$$d := \begin{cases} \text{independent}(t) \wedge \neg \text{control\_verb}(v) & \longrightarrow \text{su } v \ ob \ c \\ \text{subordinate}(t) \wedge \neg \text{controlled}(t) \wedge \neg \text{control\_verb}(v) & \longrightarrow \text{su } ob \ c \ v \\ \text{subordinate}(t) \wedge \text{controlled}(t) \wedge \neg \text{control\_verb}(v) & \longrightarrow \langle ob \ c, v \rangle \\ \text{subordinate}(t) \wedge \text{controlled}(t) \wedge \text{control\_verb}(v) & \longrightarrow \langle ob \ c.1, v \ c.2 \rangle \\ \text{subordinate}(t) \wedge \neg \text{controlled}(t) \wedge \text{control\_verb}(v) & \longrightarrow \text{su } ob \ c.1 \ v \ c.2 \\ \text{independent}(t) \wedge \text{control\_verb}(v) & \longrightarrow \text{su } v \ ob \ c.1 \ c.2 \end{cases}$$

$$\text{where } ob = \begin{cases} \text{ext\_obj}(o) & \longrightarrow \varepsilon \\ \text{else} & \longrightarrow o \end{cases} \text{ and } su = \begin{cases} \text{ext\_subj}(s) & \longrightarrow \varepsilon \\ \text{else} & \longrightarrow s \end{cases}$$

$$s := \begin{cases} \neg \text{ext\_cl}(t) & \longrightarrow \begin{cases} \neg \text{controlled}(t) & \longrightarrow \text{fcl } \Omega^e \ \Omega^e \\ \text{controlled}(t) & \longrightarrow \text{fcl } s' \ \Omega^e \end{cases} \\ \text{ext\_cl}(t) & \longrightarrow \begin{cases} \neg \text{controlled}(t) & \longrightarrow \text{fcl } \Omega^e \ e \\ \text{controlled}(t) & \longrightarrow \text{fcl } s' \ e \end{cases} \end{cases}$$

$$\text{where } \text{fcl} = \lambda s'. \lambda e. \text{su } \lambda x. ob \ \lambda y. \begin{cases} \neg \text{transitive}(v) & \longrightarrow \begin{cases} \neg \text{clause\_verb}(t) & \longrightarrow v \ x \\ \text{clause\_verb}(t) & \longrightarrow v \ x \ cl \end{cases} \\ \text{transitive}(v) & \longrightarrow \begin{cases} \neg \text{clause\_verb}(t) & \longrightarrow v \ x \ y \\ \text{clause\_verb}(t) & \longrightarrow v \ x \ y \ cl \end{cases} \end{cases}$$

$$\text{where } cl = \begin{cases} \text{ext\_cl}(c) & \longrightarrow \begin{cases} \text{ctr\_subj}(v) & \longrightarrow c \ x \ e \\ \text{ctr\_obj}(v) & \longrightarrow c \ y \ e \\ \text{else} & \longrightarrow c \ e \end{cases} \\ \neg \text{ext\_cl}(c) & \longrightarrow \begin{cases} \text{ctr\_subj}(v) & \longrightarrow c \ x \\ \text{ctr\_obj}(v) & \longrightarrow c \ y \\ \text{else} & \longrightarrow c \end{cases} \end{cases}$$

$$\text{where } su = \begin{cases} \text{ext\_subj}(s) & \longrightarrow \lambda P.P \ e \\ \text{controlled}(t) & \longrightarrow \lambda P.P \ s' \\ \text{else} & \longrightarrow s \end{cases} \text{ and } ob = \begin{cases} \text{ext\_obj}(o) & \longrightarrow \lambda P.P \ e \\ \text{else} & \longrightarrow o \end{cases}$$

Figure 18:  
Second part  
of the full  
synchronous  
grammar,  
with logical  
constraints and  
linearizations

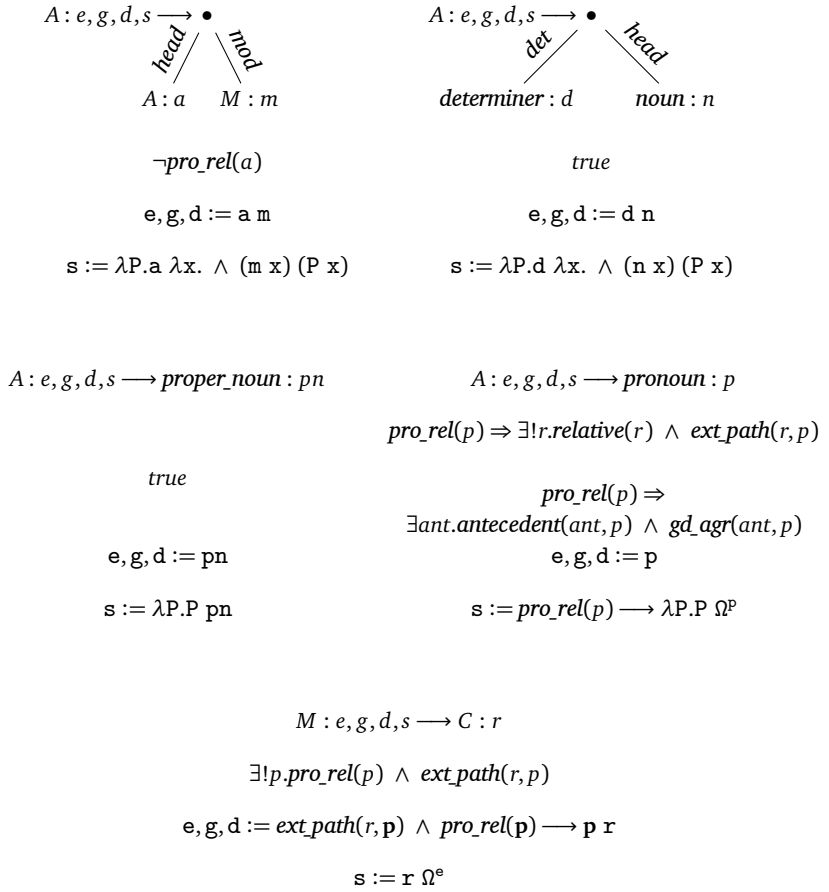


Figure 19 shows in solid edges all the paths that link  $r$  to another node  $x$  such that  $\textit{ext\_path}(r, x)$  is true. Of all these candidate nodes, only the one labelled with  $p$  satisfies  $\textit{pro\_rel}(x)$ , ensuring its existence and uniqueness and thus satisfying the constraint.

The next production to occur in the derivation tree is  $p_1$ . This production has four constraints:

$$\begin{array}{l}
 \textit{transitive}(v) \Leftrightarrow \textit{some}(o) \\
 \textit{control\_verb}(v) \Leftrightarrow \textit{some}(c) \\
 \textit{controlled}(t) \Leftrightarrow \textit{none}(s) \\
 \textit{controlled}(t) \Leftrightarrow \textit{infinitive}(v)
 \end{array}$$

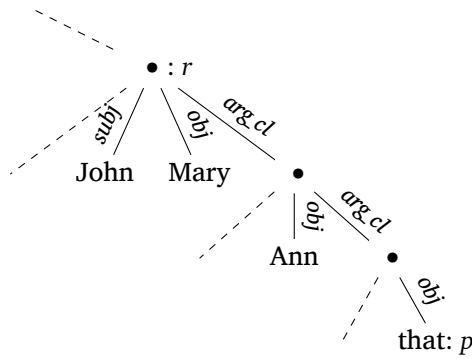


Figure 19:  
Instantiation of the logical  
constraint for  $p_6$

These four constraints ensure that the verb valency corresponds to the arguments provided by the abstract structure, and that the controlled verbs are in an infinitive form and have no redundant subject. As shown in Figure 20, in this case, all three arguments *subj*, *obj*, and *arg-cl* are present. Looking at the head “lets”, we can check that it has the properties *transitive* and *ctr\_obj*, satisfying the two first constraints. Since the edge above *t* is labelled with *mod*, we can infer from the definition of *controlled* that  $\neg\text{controlled}(t)$ , which verifies the third constraint. Finally, since “lets” does not have the property *infinitive*, the fourth constraint is also satisfied.

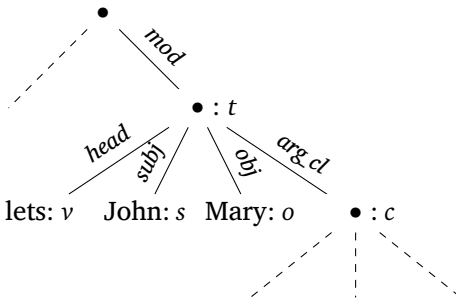
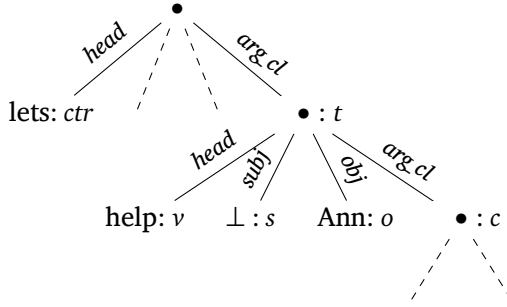


Figure 20:  
Instantiation of the logical  
constraints for the first  
occurrence of  $p_1$

The next two rewrites use the production  $p_4$ , which has no additional constraints. Then, the production  $p_1$  is used to rewrite the non-terminal labelled  $x_6$ , with the same four constraints as before. The valency constraints are satisfied in the same way (the verb expects – and gets – its optional arguments *obj* and *arg-cl* as it is both transitive and a control verb). On the other hand, the *controlled* predicate is

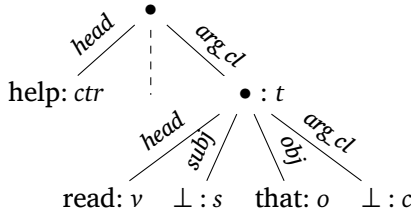
true for the node  $t$ , and hence the verb must be in the infinitive form and the *subj* argument should be  $\perp$ . As both these conditions are verified, all the constraints are again satisfied. The corresponding subtree (including the  $\perp$  leaf for *s*) is drawn in Figure 21.

Figure 21:  
Instantiation of the logical constraints for the second occurrence of  $p_1$



The next production, being  $p_4$ , has no associated constraints. Then there is one last occurrence of the production  $p_1$  which is satisfied in the same fashion as before, except that the *arg-cl* argument is absent as the leaf node  $v$  does not satisfy the *control\_verb* predicate. The corresponding tree is found in Figure 22.

Figure 22:  
Instantiation of the logical constraints for the last occurrence of  $p_1$



Finally, the last production in the derivation tree is  $p_5$  which has two constraints to satisfy:

$$pro\_rel(p) \Rightarrow \exists ! r.relative(r) \wedge ext\_path(r, p)$$

$$pro\_rel(p) \Rightarrow \exists ant.antecedent(ant, p) \wedge gd\_agr(ant, p),$$

where the variable  $p$  is instantiated with the leaf “that” which has the *pro\_rel* property. For the first constraint we consider the candidate nodes for  $r$  along the path described by *ext\_path*, to find that only the topmost one (labelled  $r$  in Figure 23) satisfies the predicate *relative*

(being a modifier of a noun phrase). Then, for the second constraint, the node labelled with *ant* in the figure constitutes a valid candidate for the existential quantifier, and verifies both relations with *p* (since *ant* and *p* are both lexical entries that have the *neuter* property).

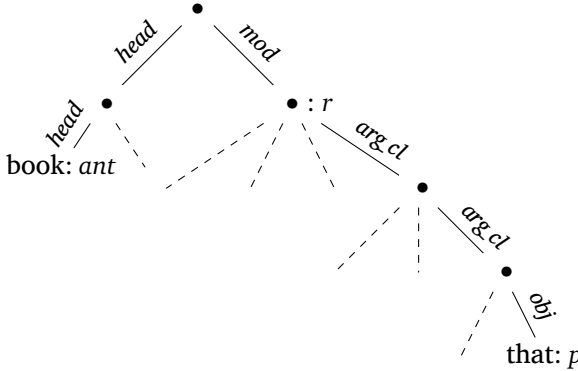


Figure 23:  
Instantiation of the logical constraints for  $p_5$

#### Linearization towards Dutch

Since the constraints of the productions used in the derivation are satisfied, then *abs* is a valid abstract structure. We can thus look at the linearization rules associated with the productions in its derivation, and construct the realization that our grammar associates with *abs*. We consider in this example the phonological linearization towards Dutch.

We construct the realization bottom-up, describing the realization associated with the left-hand side of each production by referring to the labels  $x_i$  that we have attached to the non-terminals in Figure 15.

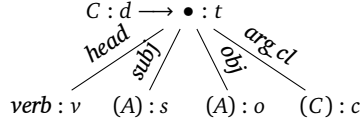
We first consider the non-terminal node labelled with  $x_9$ . It is rewritten using the production  $p_5$ , whose attached linearization rule simply yields the string representation of the terminal lexical entry in the right-hand side, namely *dat*. The realization attached to the nodes  $x_4$ ,  $x_5$ , and  $x_7$  is obtained similarly, considering the linearization rule attached to the production  $p_4$ , and yields respectively the realizations *Jan*, *Marie*, and *Anna*.

Then we consider the non-terminal node labelled with  $x_1$ , rewritten with the production  $p_3$ . The attached linearization rule combines

the realizations of the two resulting lexical entries, with the *det* argument first and the *head* argument next, yielding the string *een boek* for  $x_1$ .

We now describe the linearization of the successive clauses along the derivation tree. We recall the corresponding production  $p_1$  and the associated linearization rule for Dutch in Figure 24.

Figure 24:  
Dutch linearization for  
production  $p_1$



$$d := \left[ \begin{array}{ll} \text{independent}(t) \wedge \neg \text{control\_verb}(v) & \longrightarrow \text{su } v \text{ ob } c \\ \text{subordinate}(t) \wedge \neg \text{controlled}(t) \wedge \neg \text{control\_verb}(v) & \longrightarrow \text{su } \text{ob } c \text{ v} \\ \text{subordinate}(t) \wedge \text{controlled}(t) \wedge \neg \text{control\_verb}(v) & \longrightarrow \langle \text{ob } c, v \rangle \\ \text{subordinate}(t) \wedge \text{controlled}(t) \wedge \text{control\_verb}(v) & \longrightarrow \langle \text{ob } c.1, v \text{ c}.2 \rangle \\ \text{subordinate}(t) \wedge \neg \text{controlled}(t) \wedge \text{control\_verb}(v) & \longrightarrow \text{su } \text{ob } c.1 \text{ v } c.2 \\ \text{independent}(t) \wedge \text{control\_verb}(v) & \longrightarrow \text{su } v \text{ ob } c.1 \text{ c}.2 \end{array} \right.$$

$$\text{where } \text{ob} = \left[ \begin{array}{ll} \text{ext\_obj}(o) & \longrightarrow \varepsilon \\ \text{else} & \longrightarrow o \end{array} \right] \text{ and } \text{su} = \left[ \begin{array}{ll} \text{ext\_subj}(s) & \longrightarrow \varepsilon \\ \text{else} & \longrightarrow s \end{array} \right]$$

The first clause we consider is the one labelled with  $x_8$ . Its *subj* and *arg cl* arguments are missing, as depicted in Figure 22. We consider the logical preconditions for the linearization, starting with the *where* statements. The condition  $\text{ext\_obj}(o)$  is true (since the *obj* argument of the current clause has the property *pro\_rel*), while  $\text{ext\_subj}(s)$  is not (the *subj* argument of the clause is  $\perp$ ). Hence, we get  $\text{su} = s$  and  $\text{ob} = \varepsilon$ .

Looking at the context, the other logical conditions have the following values:  $\text{independent}(t)$  is false but  $\text{subordinate}(t)$  is true (there is another clause directly above in *abs*);  $\text{controlled}(t)$  is true (as the clause above  $t$  has the object control verb “*help*” as its head); and  $\text{control\_verb}(v)$  is false (the verb “*read*” does not have the *ctr\_subj* or *ctr\_obj* property). Hence, the only possible linearization for this clause is the third one, which yields the pair of strings  $\langle \text{ob } c, v \rangle$ . We have seen that  $\text{ob} = \varepsilon$ , and the optional argument  $c$  is not present, and therefore its realization is also taken to be the empty string  $\varepsilon$ . Hence,  $d$  has exactly one possible value that satisfies the linearization rule, which is:  $\langle \varepsilon, \text{lezen} \rangle$ .

The next clause, labelled with  $x_6$ , is rewritten using the same production and linearization rule. We recall that its instantiated labels and its context in *abs* are depicted in Figure 21.

There is no extraction *ext\_obj* or *ext\_suj* involved, hence  $ob = o$  and  $su = s$ . The node  $t$  corresponds again to a clause that satisfies both the *subordinate* and *controlled* predicates; however, the *head* argument “*help*” has the *ctr\_obj* property and hence verifies *control\_verb*( $v$ ). The selected linearization will therefore be  $\langle ob\ c.1, v\ c.2 \rangle$ , where  $c.1$  and  $c.2$  denote the first and second projection of the pair that constitutes the realization of  $c$ . Building on our previous observations, we have  $o = \text{Anna}$  and  $c = \langle \varepsilon, \text{lezen} \rangle$ . Hence, the realization associated with  $x_6$  is  $\langle \text{Anna}, \text{helpen lezen} \rangle$ .

The last clause, labelled with  $x_3$  and depicted in Figure 20, has the same logical preconditions as  $x_6$  except for the fact that it is not controlled (the edge that dominates  $t$  is labelled with *mod*). The selected realization is then the fourth one in Figure 24, that is:  $su\ ob\ c.1\ v\ c.2$ , with  $su = s$  and  $ob = o$ . The realization associated so far with the right-hand side non-terminals is such that  $s = \text{Jan}$ ,  $o = \text{Marie}$ , and  $c = \langle \text{Anna}, \text{helpen lezen} \rangle$ . Thus, the topmost clause in the relative clause is realized as *Jan Marie Anna laat helpen lezen*, with the expected Dutch cross-serial ordering.

To carry on the linearization process, we now establish the realization associated with  $x_2$ . It is constructed with the following rule:

$$d := \text{ext\_path}(r, \mathbf{p}) \wedge \text{pro\_rel}(\mathbf{p}) \longrightarrow \mathbf{p}\ r,$$

where  $r$  corresponds to the clause labelled with  $x_3$  that we have just linearized, and  $\mathbf{p}$  is any external node that satisfies the given logical precondition. As imposed by the logical constraint depicted in Figure 19, there is exactly one candidate node that satisfies this condition, that is the lexical entry “*that*”, which rewrites  $x_9$ . Note that its realization was not used in the construction of the realization of the node  $x_8$ . The realization of  $\mathbf{p}$  is then *dat*, and the full realization associated with  $x_2$  is obtained by concatenating  $\mathbf{p}$  and  $r$ , yielding: *dat Jan Marie Anna laat helpen lezen*.

Finally, the realization of the whole *abs* subtree, which does not depend on the context above  $x_0$ , is obtained by concatenating those of the nodes  $x_1$  and  $x_2$  as demanded by the linearization rule for  $p_2$ . The resulting string is: *een boek dat Jan Marie Anna laat helpen lezen*.

This paper explores the possibility of designing high-level grammars by means of Model Theoretic Syntax. We try to anchor high-level descriptions in formal methods and more particularly in logic. This allows us to obtain a precise meaning for the grammatical descriptions. Moreover, our whole methodology is favoured by the wealth of difficult results that the literature provides. Indeed, informed by those results, we have designed a logical language that seems to suit the needs of linguistic descriptions and that is also weaker than Monadic Second-Order Logic ensuring that the properties expressed by that logical language can be captured by finite state automata. Moreover, inspired by the work of Courcelle (1994), we use the flexibility of logical transduction so as to obtain an arguably simple model of extraction. Finally, all these design choices make the languages described with our system belong to the class of mildly context sensitive languages. More specifically, the grammars we obtain are 2-ACGs. We chose this grammatical model for the fact that, in their linear version, they exactly capture mildly context sensitive languages and that they allow one to model both syntax and semantics with the same set of primitives.

After Rogers (2003a), our methodology offers another way for Model Theoretic Syntax to describe languages that are outside the class of context free grammars. It can be seen as a refinement of the two step approach of Kolb *et al.* (2003) and Morawietz (2003). Moreover, this methodology can be adapted to define other formalisms: it is possible not to use a regular approximation and encode recursion directly in the logic; the logical language can be changed as long as it is weaker than MSO; and one can use grammars based on other operations and objects (such as graphs or hypergraphs). As an example, free-word order languages can be modeled within this framework by using an adapted algebra allowing one to represent free-word ordering as proposed in Kirman and Salvati (2013).

We illustrate our formalism with a small subset of interleaved phenomena that deal with extraction. The formalisation is still technical, but we argue that this technicality is mostly of linguistic nature. Indeed, the interplay of these phenomena raises a number of particular cases one eventually needs to describe. The advantage of our approach



is that it reduces the difficulty of describing this set of situations. The small macro language we have designed to deal with the parts that are common to various situations seems to be sufficient to provide linguistic generalisations.

On the semantic side, the traditional continuation passing style used in the Montagovian approach to semantics makes it hard to express the semantics in a natural way. Indeed, one would wish to simply use the logical relations on the abstract structure so as to find the argument of each predicate. But this would amount to seeing formulae as graphs and would thus break down an interesting feature of Montague semantics: the fact that it gives semantics for each constituent of a sentence. A possible way out could be the result of Kanazawa (2011) which demonstrates a link between hypergraphs and  $\lambda$ -calculus. Taking into consideration the result of Courcelle and Engelfriet (1995) that shows that hyperedge replacement grammars are closed under MSOL transductions, it could be the case that the formulae generated as graphs could then give rise to a 2-ACG providing a semantics to each constituent, and thus recovering compositional semantics.

In future work, we shall model larger fragments of natural language, by incorporating several phenomena. Moreover, as our formalism seems to adapt well to the description of synchronous grammars, we shall see how we can refine linguistic descriptions so as to allow a modular development of those grammars. The case of agreement, that may greatly vary between languages that otherwise share many syntactic constructs, as for the languages we have chosen (English, German, and Dutch) pushes us in that direction. Moreover, another direction is of course to submit our approach to experiments and more specifically to implement a compiler from high-level descriptions to actual grammars. It is indeed well-known that the automata verifying whether some constraints are verified may have a non-elementary size with respect to the size of the formula. Thus, compiling these grammatical descriptions to actual grammars may be quite challenging. Nevertheless, if these descriptions are realistic, they should be rendered by wide coverage grammars which, even though huge, can be handled by modern computers.

## REFERENCES

- Henk P. BARENDREGT (1984), *The Lambda Calculus: Its Syntax and Semantics*, volume 103, Studies in Logic and the Foundations of Mathematics, North-Holland Amsterdam, revised edition.
- Philippe BLACHE (2001), *Les grammaires de propriétés. Des contraintes pour le traitement automatique des langues naturelles*, number 2-7462-0236-0 in Technologies et cultures, Hermes Science Publications.
- Anudhyan BORAL and Sylvain SCHMITZ (2013), Model-Checking Parse Trees, in *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '13, pp. 153–162, IEEE Computer Society, Washington, DC, USA.
- Joan BRESNAN (2001), *Lexical-functional syntax*, volume 16 of *Blackwell textbooks in linguistics*, Blackwell.
- Norbert BRÖKER (1998), Separating Surface Order and Syntactic Relations in a Dependency Grammar, in *Proceedings of COLING-ACL98*, pp. 174–180.
- Marie-Hélène CANDITO (1999), *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien.*, Ph.D. thesis, Université Paris 7.
- Noam CHOMSKY (1981), Lectures on Government and Binding, in *The Pisa Lectures*, Foris Publications, Holland.
- Thomas CORNELL and James ROGERS (1998), Model theoretic syntax, *The Glot International State of the Article Book*, 1:101–125.
- Bruno COURCELLE (1994), Monadic second-order definable graph transductions: a survey, *Theoretical Computer Science*, 126:53–75.
- Bruno COURCELLE and Joost ENGELFRIET (1995), A Logical Characterization of the Sets of Hypergraphs Defined by Hyperedge Replacement Grammars, *Mathematical Systems Theory*, 28(6):515–552.
- Bruno COURCELLE and Joost ENGELFRIET (2012), *Graph Structure and Monadic Second-Order Logic*, Encyclopedia of Mathematics and its Applications, Cambridge University Press.
- Benoit CRABBÉ, Denys DUCHIER, Claire GARDENT, Joseph LE ROUX, and Yannick PARMENTIER (2013), XMG: eXtensible MetaGrammar, *Computational Linguistics*, 39(3):591–629.
- Haskell B. CURRY (1961), Some Logical Aspects of Grammatical Structure, in Roman JAKOBSON, editor, *Structure of Language and Its Mathematical Aspects*, pp. 56–68, AMS Bookstore.
- Mary DALRYMPLE (2001), *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics*, Academic Press, New York.

Philippe DE GROOTE (2001), Towards Abstract Categorical Grammars, in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, ACL '01*, pp. 252–259, Association for Computational Linguistics, Stroudsburg, PA, USA, doi:10.3115/1073012.1073045.

Philippe DE GROOTE and Sylvain POGODALLA (2004), On the expressive power of Abstract Categorical Grammars: Representing context-free formalisms, *Journal of Logic, Language and Information*, 13(4):421–438.

Ralph DEBUSMANN, Denys DUCHIER, and Geert-Jan KRUIJFF (2004), Extensible Dependency Grammar: A New Methodology, in *Recent Advances in Dependency Grammars*, pp. 78–85.

John DONER (1965), Decidability of the weak second-order theory of two successors, *Notices of the American Mathematical Society*, 12:365–468.

Denys DUCHIER, Thi-Bich-Hanh DAO, and Yannick PARMENTIER (2014), Model-theory and implementation of property grammars with features., *Journal of Logic and Computation*, 24(2):491–509.

Denys DUCHIER, Thi-Bich-Hanh DAO, Yannick PARMENTIER, and Willy LESAIN (2012), Property Grammar Parsing Seen as a Constraint Optimization Problem., in Philippe DE GROOTE and Mark-Jan NEDERHOF, editors, *Formal Grammar – 15th and 16th International Conferences, FG 2010–2012*, volume 7395, pp. 82–96, Springer.

Denys DUCHIER, Jean-Philippe PROST, and Thi-Bich-Hanh DAO (2009), A model-theoretic framework for grammaticality judgements, in *Conference on Formal Grammar (FG 2009)*, pp. 1–14.

Joost ENGELFRIET and Linda HEYKER (1992), Context-free hypergraph grammars have the same term-generating power as attribute grammars, *Acta Informatica*, 29(2):161–210.

Kilian FOTH, Wolfgang MENZEL, and Ingo SCHRÖDER (2005), Robust parsing with weighted constraints, *Natural Language Engineering*, 11(01):1–25.

J. Roger HINDLEY and Jonathan P. SELDIN (2008), *Lambda-Calculus and Combinators*, Cambridge University Press.

Aravind K. JOSHI (1985), Tree-adjointing grammars: How much context sensitivity is required to provide reasonable structural descriptions?, in David DOWTY, Lauri KARTTUNEN, and Arnold M. ZWICKY, editors, *Natural Language Parsing*, pp. 206–250, Cambridge University Press.

Makoto KANAZAWA (2009), A lambda calculus characterization of MSO definable tree transductions, *The Bulletin of Symbolic Logic*, 15(2):250–251.

Makoto KANAZAWA (2011), Parsing and Generation as Datalog Query Evaluation, Technical report, National Institute of Informatics.

Jérôme KIRMAN and Sylvain SALVATI (2013), On the Complexity of Free Word Orders, in *Proceedings of the 17th and 18th International Conferences on Formal Grammar, FG 2012, Opole, Poland, August 2012, FG 2013, Düsseldorf, Germany, August 2013, Revised Selected Papers*, volume 8036 of *Lecture Notes in Computer Science*, pp. 209–224, Springer.

Gregory M. KOBELE and Sylvain SALVATI (2013), The IO and OI Hierarchies Revisited, in *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP 2013, Part II)*, volume 7966 of *Lecture Notes in Computer Science*, pp. 336–348, Springer.

Hans-Peter KOLB, Jens MICHAELIS, Uwe MÖNNICH, and Frank MORAWIETZ (2003), An operational and denotational approach to non-context-freeness, *Theoretical Computer Science*, 293(2):261–289.

Markus KRACHT (1995), Syntactic codes and grammar refinement, *Journal of Logic, Language, and Information*, 4(1):41–60.

Richard MONTAGUE (1974), English as a Formal Language, in Richmond H. THOMASON, editor, *Formal philosophy: Selected Papers of Richard Montague*, Yale University Press, New Haven.

Frank MORAWIETZ (2003), *Two-Step Approaches to Natural Language Formalism*, number 64 in *Studies in Generative Grammar*, De Gruyter.

Geoffrey K. PULLUM (2007), The evolution of model-theoretic frameworks in linguistics, in *Proceedings of the ESSLLI 2007 Workshop on Model-Theoretic Syntax*, volume 10, pp. 1–10.

Geoffrey K. PULLUM and Barbara C. SCHOLZ (2001), On the distinction between model-theoretic and generative-enumerative syntactic frameworks, in *Proceedings of the International Conference on Logical Aspects of Computational Linguistics*, volume Complete the number of volume of Complete the title of the series, pp. 17–43, Springer.

Geoffrey K. PULLUM and Barbara C. SCHOLZ (2005), Contrasting applications of logic in natural language syntactic description, in *Logic, methodology and philosophy of science: Proceedings of the twelfth international congress*, pp. 481–503.

Michael O. RABIN (1969), Decidability of Second-Order Theories and Automata on Infinite Trees, *Transaction of the American Mathematical Society*, 141:1–35.

James ROGERS (1996), A model-theoretic framework for theories of syntax, in *Proceedings of the 34th annual meeting of the Association for Computational Linguistics*, pp. 10–16, Association for Computational Linguistics.

James ROGERS (1998), *A descriptive approach to language-theoretic complexity*, *Studies in Logic, Language & Information*, CSLI Publications, distributed by the University of Chicago Press.

*A logical approach to grammar description*

James ROGERS (2003a), Syntactic Structures as Multi-Dimensional Trees, *Research on Language and Computation*, 1(3–4):265–305.

James ROGERS (2003b), wMSO theories as grammar formalisms, *Theoretical Computer Science*, 293(2):291–320.

John Robert ROSS (1967), *Constraints on variables in syntax*, Ph.D. thesis, Massachusetts Institute of Technology.

Sylvain SALVATI (2005), *Problèmes de filtrage et problèmes d’analyse pour les grammaires catégorielles abstraites*, Ph.D. thesis, Institut National Polytechnique de Lorraine.

Sylvain SALVATI (2007), Encoding second order string ACG with Deterministic Tree Walking Transducers, in Shuly WINTNER, editor, *Proceedings of the 11th Conference on Formal Grammar (FG 2006)*, FG Online Proceedings, pp. 143–156, CSLI Publications.

Sylvain SALVATI (2009), A Note on the Complexity of Abstract Categorical Grammars, in Marcus KRACHT, Gerald PENN, and Ed STABLER, editors, *The Mathematics of Language, 10th and 11th Biennial Conference, MOL 10, Los Angeles, CA, USA, July 28–30, 2007, and MOL 11, Bielefeld, Germany, August 20–21, 2009, Revised Selected Papers*, pp. 266–271.

Sylvain SALVATI (2010), On the membership problem for non-linear ACGs, *Journal of Logic Language and Information*, 19(2):163–183.

Stuart M. SHIEBER (1985), Evidence Against the Context-Freeness of Natural Language, *Linguistics and Philosophy*, 8:333–343.

François THOMASSET and Éric Villemonte DE LA CLERGERIE (2005), Comment obtenir plus des Méta-Grammaires, in *Proceedings of TALN’05*, ATALA, Dourdan, France.

David J. WEIR (1988), *Characterizing mildly context-sensitive grammar formalisms*, Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*

<http://creativecommons.org/licenses/by/3.0/>





# A syntactic component for Vietnamese language processing

*Phuong Le-Hong*<sup>1</sup>, *Azim Roussanaly*<sup>2</sup>, and *Thi Minh Huyen Nguyen*<sup>1</sup>

<sup>1</sup> VNU University of Science, Hanoi, Vietnam

<sup>2</sup> LORIA, Université de Lorraine, Nancy, France

## ABSTRACT

This paper presents the development of a grammar and a syntactic parser for the Vietnamese language. We first discuss the construction of a lexicalized tree-adjoining grammar using an automatic extraction approach. We then present the construction and evaluation of a deep syntactic parser based on the extracted grammar. This is a complete system that produces syntactic structures for Vietnamese sentences. A dependency annotation scheme for Vietnamese and an algorithm for extracting dependency structures from derivation trees are also proposed. This is the first Vietnamese parsing system capable of producing both constituency and dependency analyses. It offers encouraging performance: accuracy of 69.33% and 73.21% for constituency and dependency analysis, respectively.

*Keywords:*  
*language,*  
*parsing,*  
*segmentation,*  
*syntactic*  
*component,*  
*tagging,*  
*tree-adjoining*  
*grammar,*  
*Vietnamese*

1

## INTRODUCTION

Natural language processing (NLP) often depends on a syntactic representation of text. Software that can generate such a representation is usually composed of both a grammar and a parser for a given language.

For decades, NLP research has mostly concentrated on English and other well-studied languages. Recently there has been increased interest in languages for which fewer resources exist, notably because of their growing presence on the Internet. Vietnamese, which is among the top 20 most spoken languages (Paul *et al.* 2014), is one such lan-

guage attracting increased attention. Obstacles remain, however, for NLP research in general and grammar development in particular: Vietnamese does not yet have vast and readily available constructed linguistic resources upon which to build effective statistical models, nor does it have reference works upon which new ideas may be experimented.

Moreover, most existing NLP research concerning Vietnamese has been focused on testing the applicability of existing methods and tools developed for English or other Western languages, under the assumption that their logical or statistical well-foundedness might offer cross-language validity; whereas assumptions about the structure of a language are usually made in such tools, and must be amended to adapt them to different linguistic phenomena. For an isolating language such as Vietnamese, techniques developed for inflectional languages cannot be applied “as is”.

Our goal is to develop a syntactic parser for the Vietnamese language. We believe that a wide-coverage grammar that incorporates rich statistical information would contribute to the development of basic linguistic resources and tools for automatic processing of Vietnamese written text.

Syntactic parsing is a fundamental task in natural language processing. For Vietnamese, there have been few published works dealing with this problem. This paper presents the construction and evaluation of a deep syntactic parser based on Lexicalized Tree-Adjoining Grammars (LTAG) for the Vietnamese language.

The remainder of the paper is organized as follows. The next section introduces some preliminary concepts of different types of syntactic representation, a brief introduction of the Vietnamese language and the tree-adjoining grammar formalism. Section 3 then presents the construction of a tree-adjoining grammar – the first part of the syntactic component. This grammatical resource is extracted automatically from the Vietnamese treebank. Next, Section 4 discusses the construction of a deep parser based on the extracted grammar. The parser is evaluated in Section 5. Section 6 concludes the paper and suggests some directions for future work.



Constituency structure and dependency structure are two types of syntactic representation of a natural language sentence. While a constituency structure represents a nesting of multi-word constituents, a dependency structure represents dependencies between individual words of a sentence. The syntactic dependency represents the fact that the presence of a word is licensed by another word which is its governor. In a typed dependency analysis, grammatical labels are added to the dependencies to mark their grammatical relations, for example *subject* or *indirect object*.

Recently, there have been many published works on dependency analysis for well-studied languages, such as English (Kübler *et al.* 2009) or French (Candito *et al.* 2009b). The dependency parsers developed for these languages are usually probabilistic and trained on corpora available in the language of interest. We can classify the architecture of such parsers into two main types:

- parsers that employ a machine learning method on dependency corpora extracted automatically from treebanks and that directly produce dependency parses (Nivre 2003, McDonald and Pereira 2006, Johansson and Nugues 2008, Candito *et al.* 2010);
- parsers that rely on a sequential process where constituency parses are produced first and then dependency parses are extracted (Candito *et al.* 2009b, de Marneffe *et al.* 2006).

This second type is motivated by the fact that dependency corpora are not readily available for many languages, as in the case of Vietnamese. In such an architecture, we need a module which takes as input constituency parses given by a constituency parser and converts these parses into typed dependency parses as illustrated in Figure 1 and Figure 2 for the English sentence “*A hearing is scheduled on the issue today*” (Nivre and McDonald 2008).

In this section we present some general characteristics of the Vietnamese language; these are adopted from Hạo (2000), Hữu *et al.* (1998) and Nguyen *et al.* (2006).

Figure 1:  
Constituency analysis  
of an English sentence

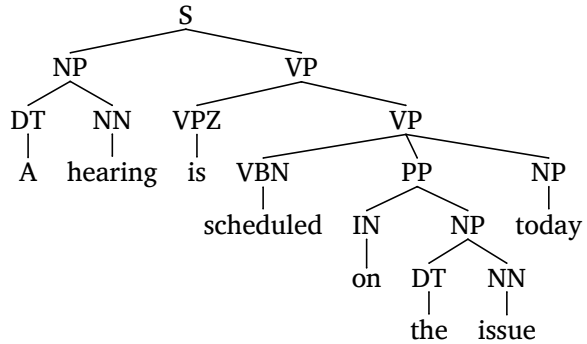
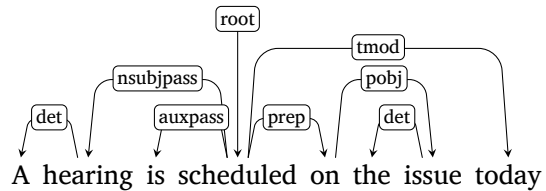


Figure 2:  
Dependency analysis  
of an English sentence



Vietnamese belongs to the VietMuong group of the Mon-Khmer branch, which in turn belongs to the Austro-Asiatic language family. Vietnamese is also similar to languages in the Tai family. The Vietnamese vocabulary features a large number of Sino-Vietnamese words which are derived from Chinese (Alves 1999). This vocabulary was originally written with Chinese characters that were used in the Vietnamese writing system, but like all written Vietnamese, is now written with the Latin-based Vietnamese alphabet that was adopted in the early 20<sup>th</sup> century. Moreover, by being in contact with the French language, Vietnamese was enriched not only in vocabulary but also in syntax by the calque (or loan translation) of French grammar. Thus, for example, the Subject-Verb-Object structure gained prevalence over the natively more common Theme-Rheme construction.

Vietnamese is an isolating language,<sup>1</sup> which means that it is characterized by the following traits:

- it is a monosyllabic language;
- its word forms never change, unlike occidental languages that use morphological variations (e.g. plural form, conjugation);

<sup>1</sup> It is noted that Chinese is also isolating; Chinese is classified in a branch of Sino-Tibetan language family.

- hence, all grammatical relations are manifested by word order and function words.

Vietnamese has a special unit called “*tiếng*” that corresponds at the same time to a syllable with respect to phonology, a morpheme with respect to morpho-syntax, and a word with respect to sentence constituent creation. For convenience, we call these “*tiếng*” syllables. The Vietnamese vocabulary contains:

- simple words, which are monosyllabic (e.g. *mưa* (rainy) *nắng* (sunny));
- reduplicated words composed by phonetic reduplication (e.g. *trắng* (white) – *trắng trắng* (whitish))
- compound words composed by semantic coordination (e.g. *quần* (trousers), *áo* (shirt) – *quần áo* (clothes))
- compound words composed by semantic subordination (e.g. *xe* (vehicle), *đạp* (to pedal) – *xe đạp* (bicycle));
- some compound words whose syllable combination is no longer recognizable (e.g. *bồ nông* (pelican))
- complex words phonetically transcribed from foreign languages (e.g. *cà phê* (coffee), from the French *café*).

The issue of syntactic category classification for Vietnamese is still in debate in the linguistic community. That lack of consensus is due to the unclear limit between the grammatical roles or syntactic functions of many words as well as the very frequent phenomenon of syntactic category mutation, by which a verb may for example be used as a noun, or even as a preposition. Vietnamese dictionaries (Hoàng 2002) use a set of 8 parts of speech proposed by the Vietnam Committee on Social Sciences (1983).

As for other isolating languages, the most important syntactic information source in Vietnamese is word order. The basic word order is Subject–Verb–Object. There are prepositions but no postpositions. In a noun phrase the main noun precedes the adjectives and the genitive follows the governing noun. These phenomena are subsumed under the term “head-initiality”.

The other syntactic means are function words, reduplication, and, in the case of spoken language, intonation.

From the point of view of functional grammar, the syntactic structure of Vietnamese follows a topic-comment structure. It belongs to the class of topic-prominent languages as described by Li and Thompson (1976). In those languages, topics are coded in the surface structure and they tend to control co-referentiality (e.g. *Cây đố lá to nên tôi không thích* (*Tree that leaves big so I not like*), which means *This tree, its leaves are big, so I don't like it*); the topic-oriented “double subject” construction is a basic sentence type (e.g. *Tôi tên là Nam, sinh ở Hà Nội* (*I name be Nam, born in Hanoi*), which means *My name is Nam, I was born in Hanoi*), while such subject-oriented constructions as the passive and “dummy” subject sentences are rare or non-existent (e.g. *There is a cat in the garden* should be translated as *Có một con mèo trong vườn* (*exist one < animal-classifier > cat in garden*)).

### 2.3 *Tree-adjoining grammars*

In the TAG formalism (Joshi and Schabes 1997), the grammar is defined by a set of elementary trees. A TAG parsing system rewrites nodes of trees rather than symbols of strings as in context-free grammars (CFG). The nodes of these trees are labelled with nonterminals and terminals. Starting from the elementary trees, larger trees are derived using composition operations of substitution and adjunction. In the case of an adjunction, the tree being adjoined has exactly one leaf node that is marked as the foot node (marked with an asterisk). Such a tree is called an *auxiliary tree*. Elementary trees that are not auxiliary trees are called *initial trees*. Each derivation starts with an initial tree. Substituting a tree  $\alpha$  in a tree  $\beta$  simply replaces a frontier substitution node in  $\beta$  with  $\alpha$ , under the convention that the non-terminal symbol of the substitution node is the same as the root node of  $\alpha$ . Only initial trees and derived trees can be substituted in another tree. Adjoining an auxiliary tree  $\beta$  at some node  $n$  of a derived tree  $\gamma$  proceeds as follows: the sub-tree  $t$  of  $\gamma$  rooted by  $n$  is removed from  $\gamma$ , and  $\beta$  is substituted for it instead, where  $t$  is substituted in the foot node of  $\beta$ . In the final derived tree, all leaves must have terminal labels.

In TAG, the derived tree does not give enough information to determine how it was constructed. The derivation tree is an object that specifies uniquely how a derived tree was constructed. The root of a derivation tree is labelled by a sentence-type initial tree. All other nodes in the derivation tree are labelled by auxiliary trees in the case

of adjunction or initial trees in the case of substitution. We use the following convention when depicting a derivation tree: trees that are adjoined to their parent tree are linked by a solid line to their parent, and trees that are substituted are linked by a dashed line.

In order to represent natural languages, TAGs are enriched with additional linguistic conventions or principles. First, a TAG for natural languages is *lexicalized* (Schabes 1990), which means that each elementary tree has a lexical anchor (usually unique, but in some cases, there is more than one anchor). Second, the elementary trees of a lexicalized TAG (LTAG) represent extended projections of lexical items (the anchors) and encapsulate all syntactic arguments of the lexical anchor; that is, they contain slots (nonterminal leaves) for all arguments. Furthermore, elementary trees are minimal in the sense that only the arguments of the anchor are encapsulated; all recursion is factored away. This amounts to the *condition on elementary tree minimality* from Frank (2002).

Because of these principles, in linguistic applications, combining two elementary trees corresponds to the application of a predicate to an argument (in case of substitution) or to the addition of modifiers (in case of adjunction). The derivation tree then reflects the predicate-argument structure of the sentence. This is why most approaches to semantics in TAG use the derivation tree as an interface between syntax and semantics.

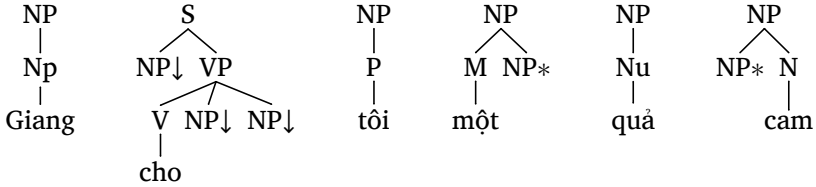
Figure 3 gives a simple Vietnamese TAG and an analysis of a sentence. The first half of the figure shows the elementary trees of the grammar and the second half shows the derived tree and its corresponding derivation tree, where the notation < anchor > represents the elementary tree corresponding to a lexical anchor. A derivation tree in TAG specifies how a derived tree was constructed.

TAG has several advantages over CFG. First, it provides an extended domain of locality. Second, the adjunction operation permits us to model long-distance relationships in single elementary trees due to the factoring of recursion.<sup>2</sup> Third, TAG derivation trees show semantic dependencies between entities in a sentence, as the tree branches rep-

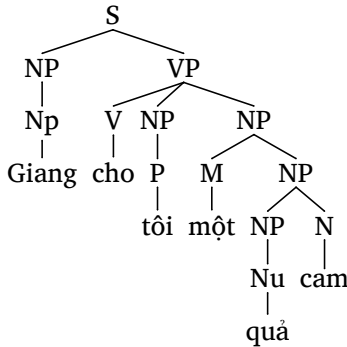
---

<sup>2</sup>These two properties follow from the mathematical properties of TAGs. TAGs belong to the class of mildly context-sensitive grammars. Context-free languages form a proper subset of tree-adjoining languages (TALs), which in turn form a proper subset of context-sensitive languages.

Elementary trees



Derived tree



Derivation tree

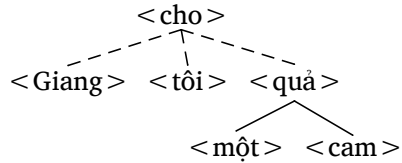


Figure 3: A TAG analysis of the sentence “Giang cho tôi một quả cam” (*Giang gave me an orange*)

represent their combination type (dashed or continuous line for substitution or adjunction, respectively, in Figure 3). In addition, in LTAG, lexical entries naturally capture constraints associated with lexical items, which is not possible in CFG. TAG and LTAG are formally equivalent; however, from the linguistic perspective, LTAG is the system we shall be concerned with in this paper.

3

GRAMMAR EXTRACTION

Since the development of hand-crafted grammars is a time-consuming and labour-intensive task, many studies on automatic and semi-automatic grammar development have been carried out during recent decades. A semi-automatic approach to building a large computational grammar is to rely on a formal language capable of describing the target grammar, e.g. a meta-grammar formalism. Many meta-grammar engineering environments were developed to support the construction

of large computational grammars for natural language. Most of them were used to build large grammars for occidental languages. A typical example is the XMG (eXtensible MetaGrammar) system which supports rapid prototyping of tree-based grammars (Crabbé *et al.* 2013). An alternative approach for obtaining grammars is to extract grammars from a treebank containing syntactically annotated sentences. This is the approach that we chose to rapidly develop a large computational grammar for Vietnamese.

We present in this section a system that automatically extracts lexicalized tree adjoining grammars from treebanks. We first discuss in detail the extraction algorithms and compare them to previous work. We then report the first results for LTAG extraction for Vietnamese, using the recently released Vietnamese treebank.

### 3.1 *Extracting grammars from treebanks*

There has been much work done on extracting treebank grammars in general and LTAG grammars in particular from annotated corpora, but all of these works are for common languages. Xia *et al.* (2000) and Xia (2001) developed the uniform method of grammar extraction for English, Chinese and Korean. Chiang (2000) developed a system for extracting an LTAG grammar from the English Penn Treebank and used it for statistical parsing with LTAG. Chen and Vijay-Shanker (2000) and Chen *et al.* (2006) extracted TAGs and there are other works based on Chen's approach such as Johansen (2004) and Nasr (2004) for French, and Habash and Rambow (2004) for Arabic. Neumann (2003) extracted lexicalized tree grammars for English from the English Penn Treebank and for German from the NEGRA treebank. Bäcker and Harbusch (2002) extracted an LTAG grammar for German – also from the NEGRA corpus – and used it for supertagging. Kaeshammer (2012) presented a grammar and a lexicon for PLTAG using the German Tiger corpus. Finally, Park (2006) extracted LTAG grammars for Korean from Korean Sejong Treebank.

### 3.2 *Vietnamese treebank*

Recently, a group of Vietnamese computational linguists has been involved in developing a treebank for Vietnamese (Nguyen *et al.* 2009). This is the treebank we used for our extraction system.

Table 1:  
Some Vietnamese treebank tags

| No. | Category | Description               |
|-----|----------|---------------------------|
| 1.  | S        | simple declarative clause |
| 2.  | VP       | verb phrase               |
| 3.  | NP       | noun phrase               |
| 4.  | PP       | preposition phrase        |
| 5.  | N        | common noun               |
| 6.  | V        | verb                      |
| 7.  | P        | pronoun                   |
| 8.  | R        | adverb                    |
| 9.  | E        | preposition               |
| 10. | CC       | coordinating conjunction  |

The construction of a Vietnamese treebank is a branch project of a national project which aims to develop basic resources and tools for Vietnamese language and speech processing.<sup>3</sup> The raw texts of the treebank are collected from the social and political sections of the Youth online daily newspaper. The corpus is divided into three sets corresponding to three annotation levels: word-segmented, POS-tagged and syntax-annotated set. The syntax-annotated corpus, a subset of the POS-tagged set, is currently composed of 10 471 sentences (225 085 tokens). Sentences range from 2 to 105 words, with an average length of 21.75 words. There are 9314 sentences of length 40 words or less. The tagset of the treebank has 38 syntactic labels (18 part-of-speech tags, 17 syntactic category tags, 3 empty categories) and 17 function tags. For details, please refer to Nguyen *et al.* (2009).<sup>4</sup> The meanings of the tags that appear in this paper are listed in Table 1.

### 3.3

#### *Extraction algorithms*

In general, our work on extracting an LTAG grammar for Vietnamese follows closely the method of grammar extraction originally proposed by Xia (2001). The extraction process has three steps: first, phrase-structure trees are converted into LTAG derived trees; second, the derived trees are decomposed into a set of elementary trees conforming to their three predefined prototypes; and third, invalid extracted elementary trees are filtered out using linguistic knowledge.

<sup>3</sup>The VLSP project, <http://vlsp.vietlp.org:8080/demo/>.

<sup>4</sup>All the resources are available at the website of the VLSP project.



The phrase structures in the Vietnamese treebank follow the English Penn Treebank (PTB) bracketed style format which are not suitable for LTAG extraction due to two reasons. First, the PTB trees do not distinguish heads, arguments and adjuncts as required in derived trees of an LTAG. Second, for each PTB tree, it is not trivial to recover a derivation tree generating it if it is not in a proper format of derived tree.

Therefore, we first have to convert the phrase structures of the treebank into derived trees by augmenting them with additional information needed for extraction.

In this step, we first classify each node in a phrase-structure tree as one of three types: head, argument or modifier. We then build a derived tree by adding intermediate nodes so that at each level of the tree, the nodes satisfy exactly one of the following relations (Xia 2001):

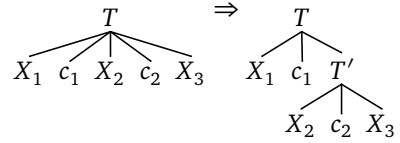
- *predicate-argument relation*: there is one (or more) node(s), where one is the head, and the rest are its arguments;
- *modification relation*: there are exactly two nodes, where one node is modified by the other;
- *coordination relation*: there are exactly three nodes, in which two nodes are coordinated by a conjunction.

In order to find heads of phrases, we have constructed a *head percolation table* (Magerman 1995; Collins 1997) for the Vietnamese treebank. This table is used to select the head child of a node. In addition, we have also constructed an *argument table* to determine the types of arguments that a head child can take. The argument table helps explicitly mark each sibling of a head child as either an argument or an adjunct according to the tag of the sibling, the tag of the head child, and the position of the sibling with respect to the head child. Together with *the tagset table*, these three tables constitute the Vietnamese treebank-specific information that is required for the extraction algorithms.

Since the conjunction structures are different from the argument and modifier structures, we first recursively bracket all conjunction groups of a treebank tree by Algorithm 1 and then build the full derived tree for the resulting tree by Algorithm 2. A conjunction group

Algorithm 1: **Data:** A syntactic tree  $T$ .  
 ProcessConj( $T$ ) **Result:**  $T$  whose conjunction groups are processed.  
**for**  $K \in T.children$  **do**  
     **if**  $IsPhrasal(K)$  **then**  
          $K \leftarrow ProcessConj(K)$ ;  
 $(\mathcal{C}_1, \dots, \mathcal{C}_k) \leftarrow ConjGroups(T.children)$ ;  
**for**  $i = 1$  **to**  $k$  **do**  
     **if**  $\|\mathcal{C}_i\| > 1$  **then**  
          $InsertNode(T, \mathcal{C}_i)$ ;  
**if**  $k > 2$  **then**  
     **for**  $i = k$  **downto**  $3$  **do**  
          $\mathcal{L} \leftarrow \mathcal{C}_{i-1} \cup c_{i-1} \cup \mathcal{C}_i$ ;  
          $T' \leftarrow InsertNode(T, \mathcal{L})$ ;  
          $\mathcal{C}_{i-1} \leftarrow T'$ ;  
**return**  $T$ ;

Figure 4:  
Transformation of  
conjunction groups



is a group of coordinating words or phrases connected by one or more coordinating conjunction. The form of a conjunction group is either “A and B” or “A or B”.<sup>5</sup> Figure 4 shows a tree with conjunction groups before and after being processed by Algorithm 1 where  $c_i$  are coordinating conjunctions and  $X_i$  are conjunction groups. Figure 5 shows a realisation of Algorithm 2 where  $A_i$  are arguments of the head child  $H$  of  $T$  and  $M_i$  are modifiers of  $H$ . These two algorithms use the function  $InsertNode(T, \mathcal{L})$  to insert an intermediate node between a node  $T$  and a list of its child nodes  $\mathcal{L}$ . This new node is a child of  $T$ , has the same label as  $T$  and has  $\mathcal{L}$  as the list of its children. The function  $IsPhrasal(X)$  checks whether  $X$  is a phrasal node or not.<sup>6</sup> The function

<sup>5</sup>In the treebank, there are no conjunctions which use the coordinating punctuation; that is, a structure like “A, B and C” is not present.

<sup>6</sup>A phrasal node is defined to be a node which is not a leaf or a preterminal. This means that it must have two or more children, or one child that is not a leaf.

**Data:** A tree  $T$  whose conjunction groups have been processed.

Algorithm 2:  
BuildDerivedTree( $T$ )

**Result:** A derived tree whose root is  $T$ .

```

if (not IsPhrasal(T)) then
 | return T ;
 $H \leftarrow$ HeadChild(T);
if not IsLeaf(H) then
 | for $K \in T.children$ do
 | | $K \leftarrow$ BuildDerivedTree(K);
 | $\mathcal{A} \leftarrow$ ArgNodes(H, \mathcal{L});
 | $\mathcal{M} \leftarrow$ ModNodes(H, \mathcal{L});
 | $m \leftarrow \|\mathcal{M}\|$;
 | if $m > 0$ then
 | | $\mathcal{L} \leftarrow \{H\} \cup \mathcal{A}$;
 | | $T' \leftarrow$ InsertNode(T, \mathcal{L});
 | | $(M_1, M_2, \dots, M_m) \leftarrow \mathcal{M}$;
 | | for $i \leftarrow 1$ to $m - 1$ do
 | | | $\mathcal{L} \leftarrow \{M_i, T'\}$;
 | | | $T'' \leftarrow$ InsertNode(T, \mathcal{L});
 | | | $T' \leftarrow T''$;
 | return T ;

```

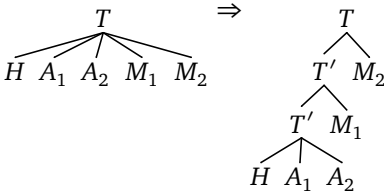
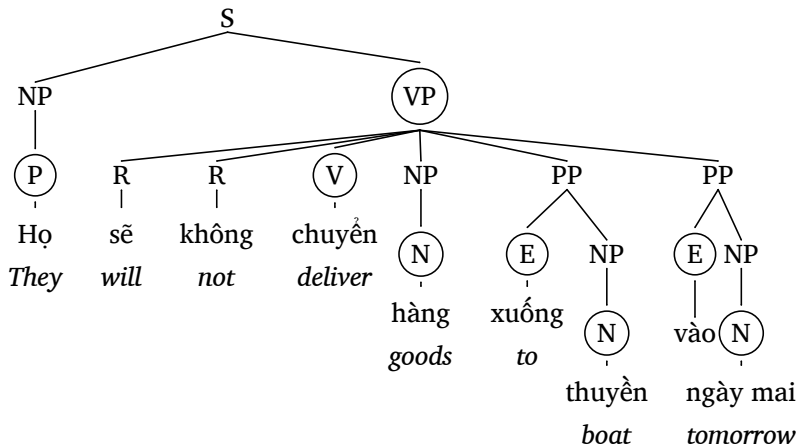


Figure 5:  
An example of derived  
tree realisation

ConjGroups( $\mathcal{L}$ ) returns  $k$  groups of components  $\mathcal{C}_i$  of  $\mathcal{L}$  which are separated by  $k - 1$  conjunctions  $c_1, \dots, c_{k-1}$ , which have a special POS tag in the treebank (CC).

Algorithm 2 uses several simple functions. The HeadChild( $X$ ) function selects the head child of a node  $X$  according to a head percolation table. The function IsLeaf( $X$ ) checks whether a node  $X$  is a

Figure 6:  
A parse tree of  
the Vietnamese  
treebank



leaf node or not. The functions  $\text{ArgNodes}(H, \mathcal{L})$  and  $\text{ModNodes}(H, \mathcal{L})$  each return a list of nodes which are arguments and modifiers, respectively, of a node  $H$ . The list  $\mathcal{L}$  contains all sisters of  $H$ .

For example, Figure 6 shows the phrase structure of a sentence extracted from the Vietnamese treebank “*Họ sẽ không chuyên hàng xuống thuyền vào ngày mai.*” (They will not deliver the goods to the boat tomorrow.) The head children of phrases are circled.

The derived tree of the sentence once processed by Algorithm 2 is shown in Figure 7, wherein the inserted nodes are marked by the quotation mark symbol (').

### 3.3.2 Building elementary trees

At this step, each derived tree is decomposed into a set of elementary trees. The recursive structures of the derived tree are factored out and will become auxiliary trees, and the remaining non-recursive structures will be extracted as initial trees.

Extracted elementary trees fall into one of three prototypes as determined by the relation between the anchor and other nodes, as shown in Figure 8. The extraction process involves copying nodes from the derived tree for building elementary trees. The result of the extraction process is three sets of elementary trees:  $\mathcal{S}$  contains spine trees,  $\mathcal{M}$  contains modifier trees and  $\mathcal{C}$  contains conjunction trees.

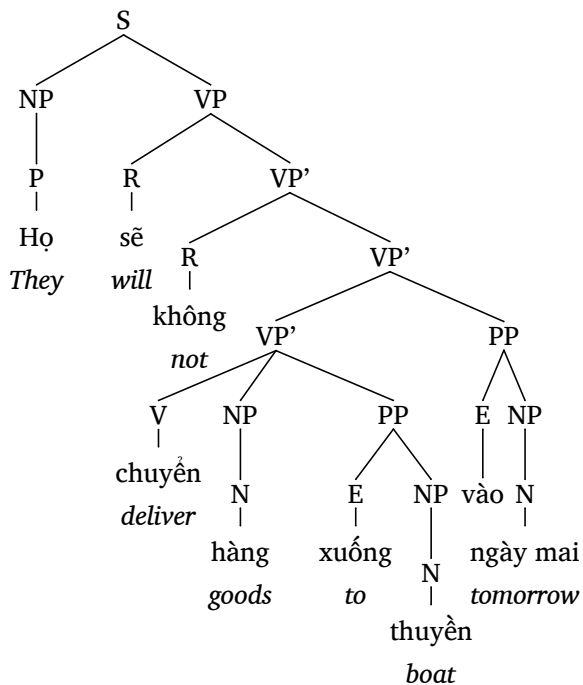


Figure 7:  
The derived tree of the  
treebank tree in Figure 6

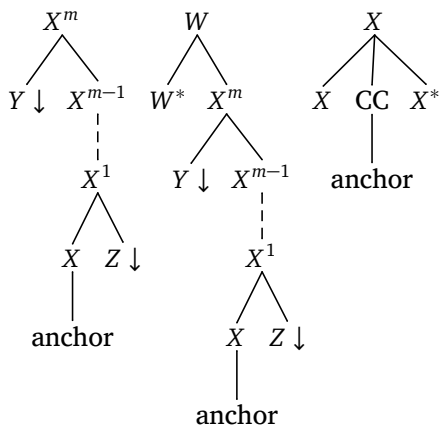


Figure 8:  
Prototypes of spine trees  
and auxiliary trees

Algorithm 3: **Data:**  $T$  is a derived tree.  
**BuildElementaryTrees**( $T$ ) **Result:** Sets  $\mathcal{S}, \mathcal{M}, \mathcal{C}$  of elementary trees.

```

if (not IsPhrasal(T)) then
 return ;
 $\{H_0, H_1, \dots, H_n\} \leftarrow$ HeadPath(T);
 $ok \leftarrow$ false;
 $P \leftarrow H_0$;
for $j \leftarrow 1$ to n do
 $\mathcal{L} \leftarrow$ Sisters(H_j);
 if $|\mathcal{L}| > 0$ then
 $Rel \leftarrow$ GetRelation(H_j, \mathcal{L});
 if $Rel =$ Coordination then
 $\mathcal{C} \leftarrow \mathcal{C} \cup$ BuildConjTree(P);
 if $Rel =$ Modification then
 $\mathcal{M} \leftarrow \mathcal{M} \cup$ BuildModTree(P);
 if $j = 1$ then
 $\mathcal{S} \leftarrow \mathcal{S} \cup$ BuildSpineTree(P);
 $ok \leftarrow$ true;
 if $Rel =$ Argument then
 if not ok and not IsLinkNode(P) then
 $\mathcal{S} \leftarrow \mathcal{S} \cup$ BuildSpineTree(P);
 $ok \leftarrow$ true;
 else
 if not IsLinkNode(P) and IsPhrasal(P) then
 $\mathcal{S} \leftarrow \mathcal{S} \cup$ BuildSpineTree(P) ;
 $P \leftarrow H_j$;

```

To build elementary trees from a derived tree  $T$ , we first find the head path<sup>7</sup>  $\{H_0, H_1, \dots, H_n\}$  of  $T$ . For each parent  $P$  and its head child  $H$ , we get the list  $\mathcal{L}$  of sisters of  $H$  and determine the relation between  $H$  and  $\mathcal{L}$ . If the relation is coordination, a conjunction tree will

---

<sup>7</sup> A *head path* starting from a node  $T$  in a derived tree is the unique path from  $T$  to a leaf node where each node except  $T$  is the head child of its parents. Here  $H_0 \equiv T$  and  $H_j$  is the parent of its head child  $H_{j+1}$ . A node on the head path is called a *link node* if its label is the same as that of its parent.

A syntactic component for Vietnamese language processing

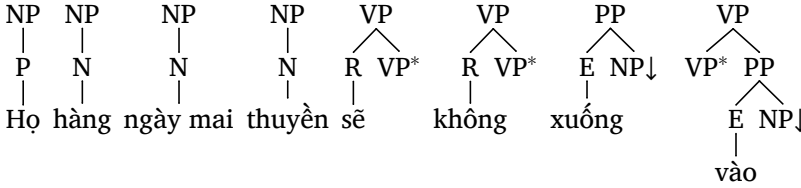


Figure 9: Extracted elementary trees

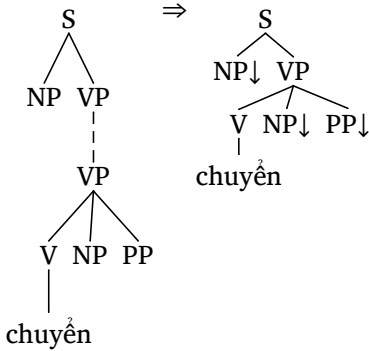


Figure 10: Merge link nodes to get a spine tree

be extracted; if the relation is modification, a modifier tree will be extracted; otherwise, the relation is predicate-argument and a spine tree will be extracted. Algorithm 3 shows the complete extraction algorithm. This algorithm uses additional functions as follows:

- BuildSpineTree( $T$ ) which creates a spine tree;
- MergeLinkNodes( $T$ ) which merges all link nodes of a spine tree into one node (see Figure 10 for an example);
- BuildModTree( $T$ ) which creates a modifier tree;
- BuildConjTree( $T$ ) which creates a conjunction tree.

As an example, from the derived tree shown in Figure 7, nine trees are extracted by algorithms as shown in Figure 9 and Figure 10.

### 3.3.3 Filtering out invalid trees

Annotation errors may be present in any particular treebank. The errors in parse trees will result in incorrect elementary trees. An elementary tree is called invalid if it does not satisfy some linguistic requirement. We have constructed some linguistic rules for filtering out invalid elementary trees. For example, in Vietnamese, an adjective (or an adjectival phrase) can be an argument of a noun (or a noun phrase); however, it must always be to the right of the noun. For instance, in

Algorithm 4: **Data:**  $T$  is a derived tree.  
**BuildSpineTree**( $T$ ) **Result:** A spine tree.

```

 $T_c \leftarrow \text{Copy}(T);$
 $P \leftarrow T_c;$
 $H \leftarrow \text{NULL};$
repeat
 $H \leftarrow \text{HeadChild}(P);$
 $\mathcal{L} \leftarrow \text{Sisters}(H);$
 if $|\mathcal{L}| > 0$ then
 $\text{Rel} \leftarrow \text{GetRelation}(H, \mathcal{L});$
 if $\text{Rel} = \text{Argument}$ then
 for $A \in \mathcal{L}$ do
 $\text{BuildElementaryTrees}(A);$
 $A.\text{children} \leftarrow \emptyset;$
 $A.\text{type} \leftarrow \text{Substitution};$
 else
 for $A \in \mathcal{L}$ do
 $P.\text{children} \leftarrow P.\text{children} \setminus A;$
 $P \leftarrow H;$
until $(H = \text{NULL});$
return $\text{MergeLinkNodes}(T_c);$

```

Algorithm 5: **Data:**  $T$  is a derived tree  
**BuildModTree**( $T$ ) **Result:** a modifier tree

```

 $T_c \leftarrow \text{Copy}(T);$
 $H \leftarrow \text{HeadChild}(T_c);$
 $H.\text{children} \leftarrow \emptyset;$
 $H.\text{type} \leftarrow \text{Foot};$
 $M \leftarrow \text{Modifier}(H);$
 $T' \leftarrow \text{BuildSpineTree}(M);$
if $|M.\text{children}| > 1$ then
 $\text{BuildElementaryTrees}(M);$
 $M \leftarrow T';$
return $T_c;$

```



**Data:**  $T$  is a derived tree.  
**Result:** A conjunction tree.  
 $T_c \leftarrow \text{Copy}(T)$ ;  
 $H \leftarrow \text{HeadChild}(T_c)$ ;  
 $\text{BuildElementaryTrees}(H)$ ;  
 $K \leftarrow \text{Coordinator}(H)$ ;  
 $\text{BuildElementaryTrees}(K)$ ;  
 $H.\text{children} \leftarrow \emptyset$ ;  
 $H.\text{type} \leftarrow \text{Foot}$ ;  
 $K.\text{children} \leftarrow \emptyset$ ;  
 $K.\text{type} \leftarrow \text{Substitution}$ ;  
**return**  $T_c$ ;

Algorithm 6:  
 $\text{BuildConjTree}(T)$

the noun phrase *cô gái đẹp* (*beautiful girl*), the adjective *đẹp* (*beautiful*) must go after the noun *cô gái* (*girl*). Thus if there is an adjective on the left of a noun of an extracted spine tree, the tree is invalid and it must be filtered out.

#### 3.4 *Comparison with previous work*

As mentioned above, our approach for LTAG extraction follows the uniform method of grammar extraction proposed by Xia (2001). Nevertheless, there are some differences between our design and implementation of extraction algorithms and that of Xia.

First, in the step in which we build the derived tree, we first recursively bracket all conjunction groups of the tree before fully bracketing the arguments and modifiers of the resulting tree. We think that this approach is easier to understand and implement since conjunction structures are different from argument and modifier structures. Second, in the elementary tree decomposition step, we do not split each node in the derived tree into the top and bottom parts as was done in Xia's approach of Xia. In our implementation, the nodes are directly copied to build extracted trees. Third, the tree extraction process is separated into functions; each function builds a particular type of elementary tree; and these functions can call each other to repeat the extraction process for the subtrees whose roots are not yet visited. In spite of using recursive functions, our extraction algorithms are carefully designed to avoid redundant or repeating function calls:

Table 2:  
Some tags in the Vietnamese treebank  
tagset are merged into a single tag

| Category            | Original tags | Tags in $G_2$ |
|---------------------|---------------|---------------|
| noun phrases        | NP/WHNP       | NP            |
| adjective phrases   | AP/WHAP       | AP            |
| adverbial phrases   | RP/WHRP       | RP            |
| preposition phrases | PP/WHPP       | PP            |
| clauses             | S/SQ          | S             |

Table 3:  
Two LTAG grammars extracted from  
the Vietnamese treebank

| Type              | # of trees    | # of templates |
|-------------------|---------------|----------------|
| $G_1$             | <b>46 382</b> | <b>2317</b>    |
| Spine trees       | 24 973        | 1022           |
| Modifier trees    | 21 309        | 1223           |
| Conjunction trees | 100           | 72             |
| $G_2$             | <b>46 102</b> | <b>2113</b>    |
| Spine trees       | 24 884        | 952            |
| Modifier trees    | 21 121        | 1093           |
| Conjunction trees | 97            | 68             |

each node is assured to be visited one time. The “divide and conquer” approach seems to be reasonably efficient and is easy to optimise.

### 3.5 An LTAG for Vietnamese

We ran extraction algorithms on the Vietnamese treebank and extracted two treebank grammars. The first one,  $G_1$ , uses the original tagset of the treebank. The second one,  $G_2$ , uses a reduced tagset, where some sets of tags in the treebank are consolidated, as shown in Table 2. The grammar  $G_2$  is smaller than  $G_1$  and it is presumed that the sparse data problem is less severe when  $G_2$  is used.

We count the number of elementary trees and tree templates. The sizes of the two grammars are in Table 3. Recall that a template is an elementary tree without the anchor word.

There are 15 035 unique words in the treebank and the average number of elementary trees that a word anchors is around 3.07. We also count the number of context-free rules of the grammars where the rules are simply read off the templates in an extracted LTAG. The extracted grammars  $G_1$  and  $G_2$  have 851 and 727 context-free rules, respectively.

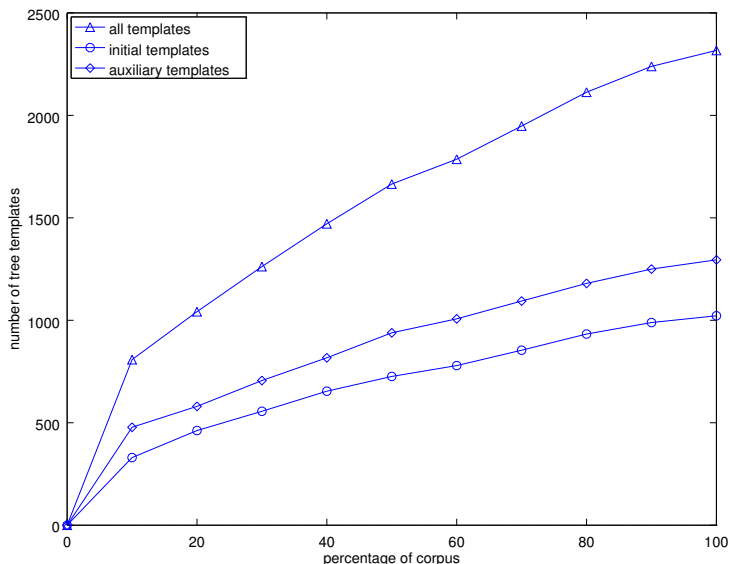


Figure 11:  
The growth of tree templates

In order to evaluate the coverage of the Vietnamese treebank, we count the number of extracted tree templates with respect to the size of the treebank. Figure 11 shows that the number of templates converges very slowly as the size of the corpus grows, implying that there are many unseen templates. This experiment also implies that the size of the current Vietnamese treebank is not large enough to cover all the grammatical templates of the Vietnamese language.

We have developed a software package<sup>8</sup> that implements the presented algorithms for extracting an LTAG for Vietnamese. The software is written in the Java programming language and is freely distributed under the GNU/GPL license. The software is very efficient in term of extraction speed: it takes only 165 seconds to extract the entire grammar  $G_1$  on an ordinary personal computer.<sup>9</sup> It should be straightforward to extend the software in order to extract LTAGs from treebanks of other languages since the language-specific information is intentionally factored out of the general framework. In order to use the software on a treebank of a given language, a user would need to provide the treebank-specific information for that language: a tagset, a head percolation table, and an argument table.

<sup>8</sup> <http://mim.hus.vnu.edu.vn/phuonglh/software/vnLExtractor>

<sup>9</sup> On an Intel Core 2 Duo CPU U9600 with 4GB RAM.

In this section, we have presented a system that automatically extracts LTAGs from treebanks. The system has been used to extract an LTAG for the Vietnamese language from the recently released Vietnamese treebank. The extracted Vietnamese LTAG covers the corpus; that is, the corpus can be seen as a collection of derived trees for the grammar and can be used to train statistical LTAG parsers directly.

The number of templates extracted from the current Vietnamese treebank converges slowly. This implies that there are many new templates outside the corpus and the current Vietnamese treebank is not large nor typical enough to cover all the grammatical templates of the Vietnamese language.

We are currently experimenting with extracting a French LTAG from a French treebank (Abeillé *et al.* 2003). We also plan to compare quantitatively syntactic structures of French and Vietnamese. We believe that a quantitative comparison of the two grammars may reveal interesting relations between them.

We present in this section the construction of a deep syntactic parser for Vietnamese. Our parser is able to produce both constituency and dependency analyses for a given sentence.

Before being parsed, a text is fed to a chain of preprocessing modules including a sentence segmenter, a word tokenizer and a tagger. In particular, we have integrated the following preprocessing modules into the parser:

- **vnSentDetector** – a sentence detector which segments a text into sentences;
- **vnTokenizer** – a tokenizer which segments sentences into words or lexical units (Le-Hong *et al.* 2008);
- **vnTagger** – a part-of-speech tagger which tags each word of a sentence with its most appropriate syntactic category (Le-Hong *et al.* 2010).

We have adapted an LTAG parser developed at the LORIA<sup>10</sup> laboratory to construct a deep syntactic parser for Vietnamese. This parser was initially used to parse French text (Roussanaly *et al.* 2005). Given a sentence, the parser outputs all possible constituency parses and their corresponding derivation trees. The most important improvement we made to the parser is the refactoring and introduction of general interfaces and modules for preprocessing tasks (sentence detection, word segmentation, POS tagging) which naturally depend on specific languages. We have also enriched the parser by adding a supplementary module which extracts dependency parses from constituency parses given by the parser.<sup>11</sup> This module implements the dependency analysis extraction algorithm which will be described in the next subsections.

#### 4.2 *Dependency annotation schema*

There exist many schema for dependency annotation. Examples include the Stanford Dependency (SD) annotation scheme (de Marneffe *et al.* 2006), created via an automated conversion of the English Penn Treebank; the PARC 700 scheme (King *et al.* 2003), inspired by functional structures of lexical functional grammars; and the GR scheme (Carroll *et al.* 1998) or EASy (Paroubek *et al.* 2005) for French. Recently, McDonald *et al.* (2013) presented a universal treebank with homogeneous syntactic dependency annotation for six languages: German, English, Swedish, Spanish, French and Korean. The multiplicity of these different annotation schema is due to different linguistic and practical choices. We prefer defining an annotation scheme of surface dependency for the Vietnamese language which can be not only convertible to different standards cited above but also enlargeable to finer dependency schema if necessary. The current scheme contains 13 grammatical relations representing principal functional dependencies between Vietnamese words. All these dependencies use the syntactic categories defined in the Vietnamese treebank (Nguyen *et al.* 2009) and they are divided into three groups.

The first group, *arg*, represents the relationship between a head word and its argument. There are two types of arguments: subject

---

<sup>10</sup><http://www.loria.fr/>

<sup>11</sup><http://mim.hus.vnu.edu.vn/phuonglh/software/vnLTAGParser>

(*subj*) or object (*obj*). It is worth noting that Vietnamese is a topic-prominent language where sentences are structured around topics rather than subjects and objects. In many cases, we cannot identify the subject and the object of a Vietnamese sentence by their respective positions. The distinction between subject and object of a Vietnamese sentence is thus not a trivial task, especially in an automatic process. Therefore, at the moment, we do not distinguish the two relations *subj* and *obj* in our evaluations. The second group, *mod*, represents modification relations of a word and its head word (or its governor). According to the syntactic category of the modifier, we distinguish nine modification relations named *modN* (nominal modifier), *modM* (numeral modifier), *modA* (adjective modifier), *modR* (adverbial modifier), *modE* (prepositional modifier), *modV* (verbal modifier), *modL* (determinant modifier), *modP* (pronominal modifier) and *modC* (subordinating coordination modifier). The third group, *coord*, represents dependencies of each lexical head of two coordinating phrases on the conjunction.

Having defined a dependency annotation scheme for Vietnamese, we now propose an algorithm for automatically extracting dependency analyses from TAG derivation trees.

#### 4.3 *Dependency relation extraction*

It has been shown that the TAG formalism shares many important similarities with the dependency grammar formalism (Rambow and Joshi 1994). A derivation tree of TAG can be converted superficially into a dependency tree in the case of lexicalized grammars (Kallmeyer and Kuhlmann 2012). The main idea is to transform each derivation operation into a dependency relation. A derivation operation between a source tree  $t_1$  and a target tree  $t_2$  results in a dependency relation between the head word of  $t_1$  as governor and the head of  $t_2$  as dependent word.

The dependency analysis corresponding to the analysis in Figure 3 is shown in Figure 12. We see that the derivation tree can be transformed into the dependency tree by a simple transformation in which each node of the derivation tree (representing an elementary tree) is replaced with its lexical node. Here, we want to extract typed dependencies where each one is labelled by a grammatical relation following the annotation scheme defined above. We thus need to consider the

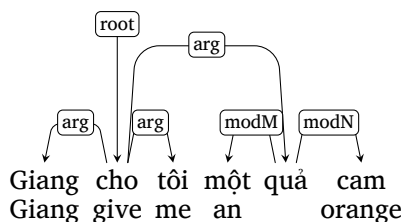


Figure 12:  
Dependency tree  
corresponding to the  
analysis in Figure 3

operation done at each node of the derivation tree. If it is a substitution, a relation of type *arg* will be created; if it is an adjunction, a relation of type *mod* will be created and its label can be determined by examining the syntactic category of the concerned word at the lexical node of the derivation tree.

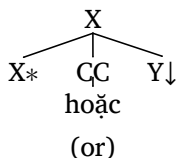
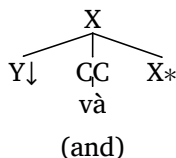


Figure 13:  
Examples of coordination  
auxiliary trees

The most difficult case is the construction of coordination relations where we must consider three related nodes and two combination operations at the same time since an auxiliary tree for conjunctions in TAG has a specific form having a substitution node and a foot node, as illustrated in example trees in Figure 13. We propose an algorithm for the automatic extraction of dependency relations from a derivation tree given by a constituency parser. The algorithm  $\text{ExtractRelations}(N)$  (Algorithm 7) shows the extraction procedure in detail.

This algorithm uses some supplementary functions as follows. The function  $\text{LexicalNode}(N)$  returns the lexical head of a node of an input derivation tree  $N$ , while the function  $\text{POSNode}(N)$  returns the part-of-speech of a lexical head. The functions  $\text{IsSubst}()$  and  $\text{IsAdj}()$  are called at each node of the derivation tree to verify whether the node is about a substitution or an adjunction. Finally, the function  $\text{NewRelation}(\text{type}, w_1, w_2)$  creates and returns a new relation of type  $\text{type}$  between two lexical units  $w_1$  and  $w_2$ .

Algorithm 7: **Data:** A derivation tree  $N$ .  
**Result:** A set  $\mathcal{R}$  of dependency relations.

```

ExtractRelations(N)
 $w_n \leftarrow \text{LexicalNode}(N)$;
 $t_n \leftarrow \text{POSNode}(N)$;
 for $K \in N.\text{children}$ do
 $w_k \leftarrow \text{LexicalNode}(K)$;
 $t_k \leftarrow \text{POSNode}(K)$;
 if $K.\text{IsSubst}()$ then
 if $t_n = \text{CC}$ then
 $\mathcal{R} \leftarrow \mathcal{R} \cup \text{NewRelation}(\text{coord}, w_n, w_k)$;
 else
 $\mathcal{R} \leftarrow \mathcal{R} \cup \text{NewRelation}(\text{arg}, w_n, w_k)$;
 else
 if $K.\text{IsAdj}()$ then
 if $t_k \in \{A, N, R, V, E, L, M, P, C\}$ then
 $\mathcal{R} \leftarrow \mathcal{R} \cup \text{NewRelation}(\text{mod}t_k, w_n, w_k)$;
 if $t_k = \text{CC}$ then
 $\mathcal{R} \leftarrow \mathcal{R} \cup \text{NewRelation}(\text{coord}, w_k, w_n)$;
 // Recursively extract relations from tree K ;
 ExtractRelations(K);
 return \mathcal{R} ;

```

For example, the application of this algorithm on the input derivation tree in Figure 1 results in the following relations:  $\text{arg}(\text{cho}, \text{Giang})$ ,  $\text{arg}(\text{cho}, \text{tôi})$ ,  $\text{arg}(\text{cho}, \text{quả})$ ,  $\text{mod}M(\text{quả}, \text{một})$ ,  $\text{mod}N(\text{quả}, \text{cam})$ .

5

## PARSER EVALUATION

In this section, we evaluate the parser on a test corpus. The parser performance is considered in two versions, with and without using part-of-speech (POS) tagging.

The grammar used to evaluate the parser is an LTAG extracted from the Vietnamese Treebank (Nguyen *et al.* 2009) containing 10 163 sentences (225 085 words, about 22.14 words per sentence on average). Figure 14 shows the distribution of the number of sentences ac-



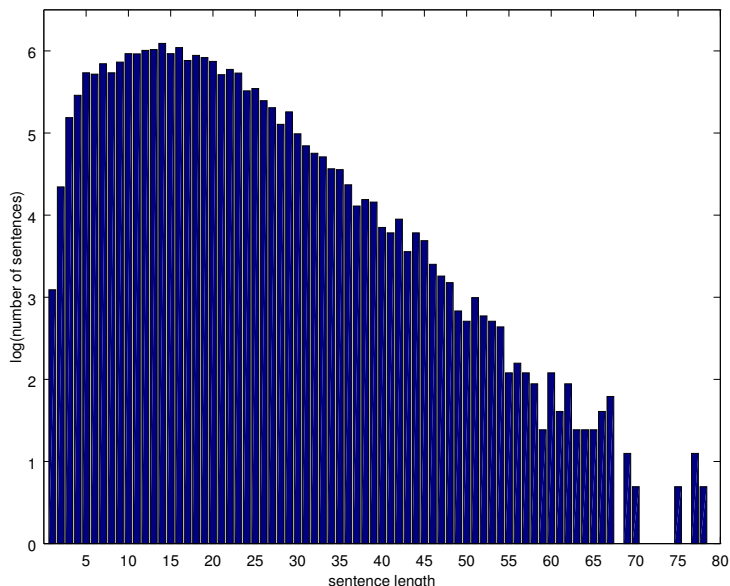


Figure 14:  
The distribution of the number of sentences according to their length

according to their lengths. We see that most of the sentences have a length between 5 and 30 words.

We choose a subset of the treebank containing 8808 sentences of length 30 words or less as an evaluation corpus. This corpus is divided into two sets: a training set (95% of the corpus, 8367 sentences) and a test set (5% of the corpus, 441 sentences). We use **vnLEXtractor** to extract an LTAG for Vietnamese from the training set. This grammar contains 35 655 elementary trees instantiated from 1658 tree templates. The size of this grammar is shown in Table 4.

| Type              | Number of trees | Number of templates |
|-------------------|-----------------|---------------------|
| Spine trees       | 19 708          | 741                 |
| Modifier trees    | 15 868          | 860                 |
| Conjunction trees | 79              | 57                  |
| <b>Total</b>      | <b>35 655</b>   | <b>1658</b>         |

Table 4:  
Size of the LTAG extracted from the training corpus

To evaluate the parser, we make use of two measures: *tree accuracy* (or *T-accuracy*) and *dependency accuracy* (or *D-accuracy*).<sup>12</sup>

<sup>12</sup>In computing these scores, unanalyzable sentences and punctuations are not taken into account.

Table 5:  
Performance of the  
constituency analysis  
without or with POS  
tagging

| <i>T</i> -accuracy         | All    |              | ≤ 10 words |              |
|----------------------------|--------|--------------|------------|--------------|
|                            | No POS | POS          | No POS     | POS          |
| Precision                  | 67.98  | 69.15        | 71.28      | 71.60        |
| Recall                     | 68.40  | 69.52        | 71.39      | 72.30        |
| <i>F</i> -measure          | 68.19  | <b>69.33</b> | 71.33      | <b>71.95</b> |
| Complete match             | 13.00  | 16.67        | 17.57      | 20.69        |
| Average crossing           | 2.66   | 2.39         | 1.80       | 1.69         |
| No crossing                | 23.00  | 27.78        | 29.73      | 32.76        |
| Fewer than three crossings | 55.00  | 54.17        | 68.92      | 65.52        |
| Tagging accuracy           | 87.72  | 95.25        | 87.34      | 95.43        |

Table 6:  
Performance of the  
dependency analysis  
without or with POS  
tagging

| <i>D</i> -accuracy | With type |       | Without type |       |
|--------------------|-----------|-------|--------------|-------|
|                    | No POS    | POS   | No POS       | POS   |
| Precision          | 70.83     | 71.81 | 74.02        | 73.21 |
| Complete match     | 15.87     | 20.00 | 23.37        | 25.45 |

When there are multiple parse trees for a sentence (which is very often the case even with short sentences), we choose one of *the derivation trees whose derived trees have smallest number of nodes* because these parses correspond to the most specific tree.<sup>13</sup>

### 5.1 Performance without POS tagging

First, the parser is evaluated without using a POS tagger. That is, the module **vnTagger** is not integrated into the parser. In this setting, each word occurrence of an input sentence is tagged with all possible tags that have been assigned to it in the training set. Unknown words are tagged as common nouns (label N). We first evaluate the performance of the constituency analysis. The results are shown in Table 5.<sup>14</sup>

In addition to the familiar precision and recall ratios, other measures are reported to help analyze the results:<sup>15</sup>

<sup>13</sup>In case of equality by this criterion, we take the first result returned by the parser.

<sup>14</sup>The presented evaluation results are calculated automatically by EVALB, a tool used frequently for the evaluation of syntactic constituency analysis which is distributed freely at <http://nlp.cs.nyu.edu/evalb/>.

<sup>15</sup>The *F*-measure is the harmonic mean of precision and recall and is computed as  $F = 2 \frac{PR}{P+R}$ .

- Complete match ratio is the percentage of sentences where recall and precision are both 100%. About 13% of the test sentences match completely. The complete match ratio for sentences of 10 words or less is 17.57%.
- The average crossing ratio is the number of constituents crossing a test constituent divided by the number of sentences of the test corpus.
- The no crossing ratio is the percentage of sentences which have zero crossing brackets. There are 23% of the test sentences that do not have any crossing (29.73% for the sentences of 10 words or less). There are 55% (respectively 68.92%) of the test sentences which have fewer than three crossings.
- The tagging accuracy is the percentage of correct POS tags (without punctuations). It is interesting to note that the tagging accuracy declines slightly when shorter test sentences are used.

The performance of dependency analysis is evaluated in two versions: with and without type. In the first version, two typed dependencies  $type_1(u_1, v_1)$  and  $type_2(u_2, v_2)$  are considered equal if three corresponding parts of these dependencies are all equal, that is  $type_1 \equiv type_2, u_1 \equiv u_2, v_1 \equiv v_2$ . In the second version, we compare only two pairs of concerned words without using their dependency types. The  $D$ -accuracy of the two evaluations are given in Table 6.<sup>16</sup> Table 7 shows the system's performance for each dependency type.

We see that the parser works perfectly on coordination structures, as they are inherently unambiguous in both the grammar and the extraction algorithm. The performance on the dependencies of type *argument* is much better than that of type *modifier*. These results justify a higher ambiguity of the adjunction operation of the LTAG formalism (which is related to auxiliary trees) in comparison with the substitution operation (which is related to initial trees).

We observe that the parser could not parse about 16.6% of the test corpus. We believe that there may be two main reasons that some sentences can not be analysed. First, there is an insufficient coverage of the underlying LTAG grammar used by the parser. That is, the gram-

---

<sup>16</sup>Note that when evaluating the accuracy of a dependency analysis, we do not need to compute precision or recall ratios since they are equal: the number of relations given by the parser always matches the number of correct relations.

Table 7:  
Performance of  
dependency  
analysis by type  
without or with  
POS tagging

| Type         | Precision |        | Recall |        | F-measure |        |
|--------------|-----------|--------|--------|--------|-----------|--------|
|              | No POS    | POS    | No POS | POS    | No POS    | POS    |
| <i>arg</i>   | 87.57     | 87.18  | 79.02  | 80.95  | 83.08     | 83.95  |
| <i>coord</i> | 100.00    | 100.00 | 100.00 | 100.00 | 100.00    | 100.00 |
| <i>modA</i>  | 48.57     | 59.09  | 62.96  | 65.00  | 54.84     | 61.90  |
| <i>modC</i>  | 46.67     | 66.67  | 43.75  | 60.00  | 45.16     | 63.16  |
| <i>modE</i>  | 50.00     | 35.71  | 56.52  | 35.71  | 53.06     | 35.71  |
| <i>modL</i>  | 72.73     | 100.00 | 47.06  | 50.00  | 57.14     | 66.67  |
| <i>modM</i>  | 80.00     | 81.82  | 53.33  | 75.00  | 64.00     | 78.26  |
| <i>modN</i>  | 50.00     | 58.54  | 66.67  | 68.57  | 57.14     | 63.16  |
| <i>modR</i>  | 64.10     | 47.06  | 60.98  | 42.11  | 62.50     | 44.44  |
| <i>modV</i>  | 52.63     | 58.33  | 62.50  | 87.50  | 57.14     | 70.00  |

mar extracted from the training corpus does not contain the syntactic structure (elementary trees) of a given sentence to be parsed. Secondly, our heuristic choice of tagging all the new words as a common noun may effectively introduce errors prior to the analysis, which may result in analysis failures. We have not yet thoroughly investigated these causes.

The ambiguity and the duration of parsing are strongly dependent on the length of sentences, as shown in Figure 15. It seems that the number of parses has an exponential growth with respect to the length of the sentence.<sup>17</sup>

## 5.2 Performance with POS tagging

The results reported in the previous subsection make possible a preliminary evaluation of the grammar and the performance of the parser. Nevertheless, the condition under which the experimentation is carried out is rather harsh since the parser has to try all possible syntactic categories of each word of an input sentence. The experiments in this subsection are closer to real use conditions, in that each sentence is first processed by a tagger to remove POS-tagging ambiguity – each word is assigned a unique tag. We have thus a sole sequence of

<sup>17</sup>For some considerably long sentences, the parser could not give any result after a fixed time-out predefined at 3 minutes. We limit the sentence length to 15 words in the experiments with the symbolic parser.

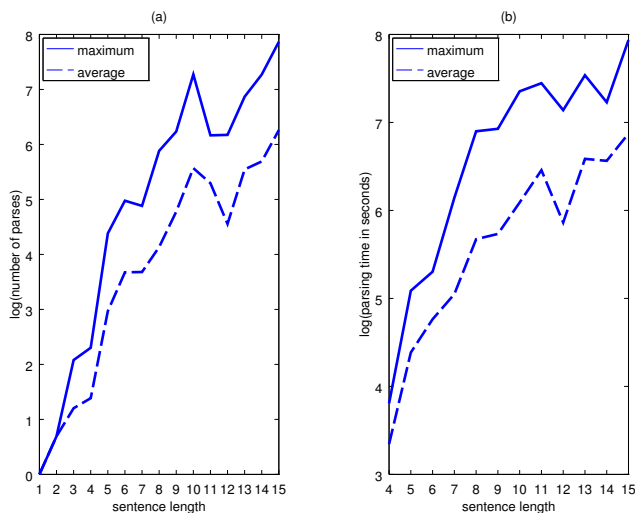


Figure 15: The ambiguity (a) and duration (b) of analysis, average and maximum, according to the length of sentences

words/tags and it is used as input to the syntactic parser. The tagging is done by the **vnTagger** module.

We proceed with the evaluation of this parser version in a similar way as presented for the previous version without POS tagging. We first give constituency parsing results, then dependency parsing results and finally the ambiguity and duration of the parsing.

The *T*-accuracy of the system is shown in Table 5. By integrating a POS tagger, the tagging accuracy is greatly improved, from 87.72% to 95.25%.<sup>18</sup> This helps improve all the scores of the system, notably the complete match ratio, from 13.00% to 16.67% (and that for sentences of length 10 words or less improves to 20.69%).

The dependency analysis performance both with and without type is shown in Table 6 and the performance of particular dependency types is shown in Table 7.

We see that the performance of the system is improved slightly in comparison with the system without tagging. However, the most important benefit of the parser with the integrated tagger is a strong reduction of analysis ambiguity and time, shown in Figure 16. The tag-

<sup>18</sup>Recall that the test corpus only contains sentences of 30 words or less.

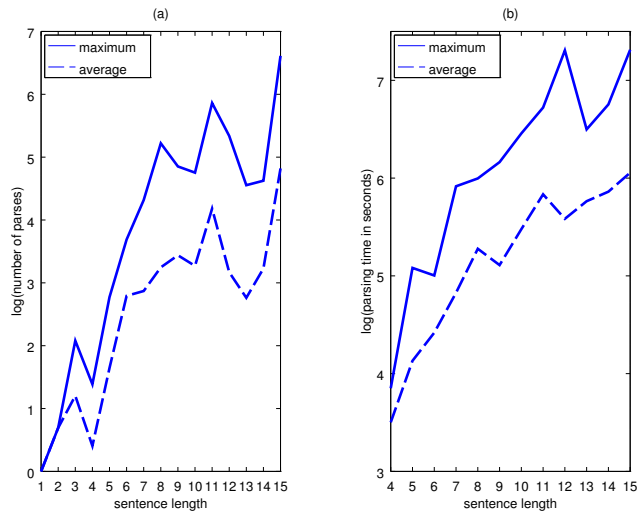


Figure 16: The ambiguity (a) and duration (b) of analysis, average and maximum, with an integrated tagger

ger helps reduce analysis ambiguity fivefold on average and reduces analysis duration three times in comparison with the required time of the parser without prior tagging. Nevertheless, we observe that the integration of the tagger results in a higher number of sentences that the parser could not parse, to 40% of the test corpus. This result is to be expected because in this version the parser uses only a syntactic category (the most probable POS) given by the tagger for each word. (We note also that the precision of the tagger at sentence level is about 32% (Le-Hong *et al.* 2010); that is, the tagger can give correct tags for all the words of a sentence to be parsed only one third of the time).

### 5.3

#### *Discussion*

In the previous section, we evaluated a syntactic analysis system based on LTAG for Vietnamese. The best results obtained are 73.21% (dependency accuracy, or D-accuracy) and 69.33% (*F*-measure of constituency accuracy, or T-accuracy, measured by EVALB) on a test corpus.

It is worth noting that these are the first results of syntactic analysis of Vietnamese based on LTAG. To our knowledge, to date there

have been few published works on the syntactic analysis of Vietnamese. The most complete report on parser performance for Vietnamese is an empirical study of applying probabilistic CFG parsing models by Collins (2003); its best result on constituency analysis is 78% *T*-accuracy on a test corpus, while there is no result reported for dependency analysis. Concerning the constituency parsing result, their parser is slightly better than ours. However, these results are not directly comparable since the parsing models are trained and tested on a different corpus.

Our first results of the syntactic parsing of Vietnamese are rather good although they are still significantly weaker than parsing results for well-studied languages like English (whose *T*-accuracy is 91.10% (Carreras *et al.* 2008) and whose *D*-accuracy is 92.93% (Koo and Collins 2010) on the Penn Treebank) or French (*T*-accuracy is 86.41% (Candito *et al.* 2009a) and *D*-accuracy is 85.55% on a French treebank (Candito *et al.* 2010)). However, we can improve our results by correcting three main sources of errors identified by the experiments; we examine each such type of error presently.

The principal source of parsing errors is the selection of parse. When there are multiple parses for a sentence, only the parse whose derivation tree contains the fewest nodes is selected. Although the returned tree corresponds to the most specific analysis, it is obvious that this selection method is purely heuristic and fragile. However, the use of a probabilistic parser does not improve significantly the parsing accuracy. We think that the parameters of the statistical parser are currently not optimised for parsing Vietnamese or the Vietnamese grammar is not large enough in order for the statistical parser to be effective. Consequently, optimising parameters of the statistical parsing model could help improve the parsing performance.

The second source of parsing errors is the POS tagging. In the experiments with a tagger integrated, we use only the most confident prediction generated by **vnTagger** as input to the parser. We have seen that the tagger often makes errors at the sentence level; perfectly tagged sentences are rare. A tagging error may effectively introduce one or more parsing errors. An improvement in tagging performance is thus another necessary condition to improve the performance of the parser.

The third source of parsing errors concerns the coverage of the grammar used in the experiments. In general, the proportion of test sentences having at least one word that the grammar does not recognize is rather high, at about 15%. Consequently, the parser could not build the correct analysis for these sentences. A straightforward solution to this problem is to enlarge the coverage of the LTAG grammar, which in turn necessitates an enlargement of the Vietnamese treebank. However, developing such a corpus is an expensive and labour-intensive task. In addition, this may lead to the typical problem of a symbolic syntactic parser: the tradeoff between its performance and its efficiency. This is an interesting problem in itself, which we shall investigate in future works.

In this article, we have presented a complete syntactic component for Vietnamese language processing. The component comprises two essential resources: a lexicalized tree-adjoining grammar for Vietnamese and a set of software tools that are chained together to produce syntactic structures from Vietnamese raw text. The grammar is extracted automatically from a treebank by an efficient algorithm. The software includes necessary modules for detecting sentence boundaries, tokenizing word units, part-of-speech tagging and syntactic parsing. This syntactic component is the first system capable of generating both constituency and dependency analyses for the language with encouraging performance.

Syntactic dependency representation of natural sentences has gained a wide interest in the natural language processing community and has been successfully applied to many problems and applications such as machine translation (Ding and Palmer 2004), ontology construction (Snow *et al.* 2005) and automatic question answering (Lin and Pantel 2001). A primary advantage of dependency representation is its natural mechanism for representing discontinuous constructions or long distance dependencies which are common in Vietnamese. We think that the presence of a good dependency schema and a dependency parser for Vietnamese will be very helpful in a wide range of tasks for Vietnamese processing.



We have seen in recent years a rapid increase of research on data-driven dependency parsers, especially the rise of statistical methods in natural language processing where dependency annotated corpora exist. These parsers use one of two predominant paradigms for data-driven dependency parsing which are often called graph-based and transition-based dependency parsing. However, the constituency parser and dependency parser developed in this work are currently purely symbolic in that they do not make use of any probabilistic evidence to discriminate good parses from bad ones for a given sentence, regardless of its grammaticality. An initial investigation of statistical dependency parsing for Vietnamese has shown encouraging results (Nguyen *et al.* 2013). We believe that there is room to improve the performance of dependency parsers in general and of our dependency parser in particular by employing a hybrid approach: use elementary trees of an lexicalized tree-adjoining grammar as good syntactic features in a statistical dependency parser. This is an interesting problem that we plan to work on in the future.

#### ACKNOWLEDGEMENTS

This research is funded by the Vietnam National University, Hanoi (VNU) under project number QG.15.04. Any opinions, findings and conclusion expressed in this paper are those of the authors and do not necessarily reflect the view of VNU.

We are grateful to our three anonymous reviewers for their insightful comments, which helped us improve the quality of the article in terms of both presentation and content. Finally, we thank the copy editors of the Journal of Language Modelling for their great job on the manuscript.

## REFERENCES

- Anne ABEILLÉ, Lionel CLÉMENT, and François TOUSSENEL (2003), Building a treebank for French, in Anne ABEILLÉ, editor, *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*, pp. 165–187, Springer Netherlands.
- Mark ALVES (1999), What’s so Chinese about Vietnamese?, in *Proceedings of the Ninth Annual Meeting of the Southeast Asian Linguistics Society*, pp. 221–224, University of California, Berkeley, USA.
- Jens BÄCKER and Karin HARBUSCH (2002), Hidden Markov model-based supertagging in a user-initiative dialogue system, in *Proceedings of TAG + 6*, pp. 269–278, Università di Venezia, Italy.
- Marie CANDITO, Benoît CRABBÉ, and Djamé SEDDAH (2009a), On statistical parsing of French with supervised and semi-supervised strategies, in *Proceedings of EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference*, pp. 49–57, Athens, Greece.
- Marie CANDITO, Benoît CRABBÉ, and Pascal DENIS (2010), Statistical French dependency parsing: treebank conversion and first results, in *Proceedings of LREC 2010*, pp. 19–21, Valletta, Malta.
- Marie CANDITO, Benoît CRABBÉ, Pascal DENIS, and François GUÉRIN (2009b), Analyse syntaxique du français : des constituants aux dépendances (Syntactic Parsing of French: from constituents to dependencies), in *Actes de Traitement Automatique des Langues*, pp. 40–49, Senlis, France.
- John CAROLL, Ted BRISCOE, and Antonio SANFILIPPO (1998), Parser evaluation: a survey and a new proposal, in *Proceedings of LREC 1998*, Granada, Spain.
- Xavier CARRERAS, Michael COLLINS, and Terry KOO (2008), TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing, in *Proceedings of CoNLL 2008*, pp. 9–16, Manchester, UK.
- John CHEN, Srinivas BANGALORE, and K. VIJAY-SHANKER (2006), Automated extraction of tree-adjointing grammars from treebanks, *Natural Language Engineering*, 12(3):251–299.
- John CHEN and K. VIJAY-SHANKER (2000), Automated extraction of TAGs from the Penn treebank, in *Proceedings of the Sixth International Workshop on Parsing Technologies*.
- David CHIANG (2000), Statistical parsing with an automatically extracted tree adjoining grammar, in *Proceedings of ACL*, pp. 456–463, Morristown, New Jersey, USA.
- Michael COLLINS (1997), Three generative, lexicalised models for statistical parsing, in *Proceedings of ACL*, pp. 16–23, Association for Computational Linguistics, Stroudsburg, Pennsylvania, USA.

- Michael COLLINS (2003), Head-driven statistical models for natural language parsing, *Computational Linguistics*, 29(4):589–637.
- Benoît CRABBÉ, Denys DUCHIER, Claire GARDENT, Joseph Le ROUX, and Yannick PARMENTIER (2013), XMG: eXtensible MetaGrammar, *Computational Linguistics*, 39(3):591–629.
- Marie-Catherine DE MARNEFFE, Bill MACCARTNEY, and Christopher D. MANNING (2006), Generating typed dependency parses from phrase structure parses, in *Proceedings of LREC 2006*, pp. 449–454, Genoa, Italy.
- Yuan DING and Martha PALMER (2004), Synchronous dependency insertion grammars: a grammar formalism for syntax-based statistical machine translation, in *Workshop on Recent Advances in Dependency Grammars*, pp. 90–97, Geneva, Switzerland.
- Robert FRANK (2002), *Phrase structure composition and syntactic dependencies*, MIT Press, Boston, USA.
- Nizar HABASH and Owen RAMBOW (2004), Extracting a tree-adjoining grammar from the Penn Arabic treebank, in *Actes de Traitement Automatique des Langues*, pp. 50–55, Fez, Morocco.
- Cao Xuân HẠO (2000), *Vietnamese – Some Questions on Phonetics, Syntax and Semantics (in Vietnamese)*, NXB GD, Hanoi, Vietnam.
- Phê HOÀNG (2002), *Vietnamese Dictionary*, NXB DN, Danang, Vietnam.
- Đạt HỮU, Trí Dõi TRẦN, and Thanh Lan ĐÀO (1998), *Basis of Vietnamese (in Vietnamese)*, NXB GD, Hanoi, Vietnam.
- Ane-Dybro JOHANSEN (2004), *Extraction des grammaires LTAG à partir d'un corpus étiqueté syntaxiquement*, Master's thesis, Université Paris 7, Paris, France.
- Richard JOHANSSON and Pierre NUGUES (2008), Dependency-based syntactic-semantic analysis with PropBank and NomBank, in *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pp. 183–187, Manchester, UK.
- Aravind K. JOSHI and Yves SCHABES (1997), Tree Adjoining Grammars, in Grzegorz ROZENBERG and Arto SALOMAA, editors, *Handbooks of Formal Languages and Automata*, pp. 69–123, Springer-Verlag, New York, USA.
- Miriam KAESHAMMER (2012), *A German treebank and lexicon for tree-adjoining grammars*, Master's thesis, Universität des Saarlandes, Saarlandes, Germany.
- Laura KALLMEYER and Marco KUHLMANN (2012), A formal model for plausible dependencies in lexicalized tree adjoining grammar, in *Proceedings of TAG + 11*, pp. 108–116, Paris, France.
- Tracy Holloway KING, Richard CROUCH, Stefan RIEZLER, Mary DALRYMPLE, and Ronald M. KAPLAN (2003), The PARC 700 dependency bank, in *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora*, pp. 1–8, Budapest, Hungary.

- Terry KOO and Michael COLLINS (2010), Efficient third-order dependency parsers, in *Proceedings of ACL*, pp. 1–11, Uppsala, Sweden.
- Sandra KÜBLER, Ryan MCDONALD, and Joakim NIVRE (2009), *Dependency parsing*, Morgan & Claypool Publishers.
- Phuong LE-HONG, Thi Minh Huyen NGUYEN, Azim ROUSSANALY, and Tuong Vinh HO (2008), A hybrid approach to word segmentation of Vietnamese texts, in *Proceedings of LATA, LNCS 5196*, pp. 240–249, Springer.
- Phuong LE-HONG, Azim ROUSSANALY, Thi Minh Huyen NGUYEN, and Mathias ROSSIGNOL (2010), An empirical study of maximum entropy approach for part-of-speech tagging of Vietnamese texts, in *Actes de Traitement Automatique des Langues*, pp. 50–61, Montreal, Canada.
- Charles N. LI and Sandra A. THOMPSON (1976), Subject and topic: a new typology of language, in *Subject and topic*, pp. 457–489, London/New York: Academic Press.
- Dekang LIN and Patrick PANTEL (2001), Discovery of inference rules for question answering, *Natural Language Engineering*, 7(4):343–360.
- David M. MAGERMAN (1995), Statistical decision-tree models for parsing, in *Proceedings of ACL*, pp. 276–283, Stroudsburg, Pennsylvania, USA.
- Ryan MCDONALD, Joakim NIVRE, Yvonne QUIRMBACH-BRUNDAGE, Yoav GOLDBERG, Dipanjan DAS, Kuzman GANCHEV, Keith HALL, Slav PETROV, Hao ZHANG, Oscar TÄCKSTRÖM, Claudia BEDINI, Nuria Bertomeu CASTELLÓ, and Jungmee LEE (2013), Universal dependency annotation for multilingual parsing, in *Proceedings of ACL*, pp. 92–97, Sofia, Bulgaria.
- Ryan MCDONALD and Fernando PEREIRA (2006), Online learning of approximate dependency parsing algorithms, in *Proceedings of EACL*, pp. 81–88, Trento, Italy.
- Alexis NASR (2004), *Analyse syntaxique probabiliste pour grammaires de dépendances extraites automatiquement*, Habilitation à diriger des recherches, Université Paris 7, Paris, France.
- Günter NEUMANN (2003), A uniform method for automatically extracting stochastic lexicalized tree grammar from treebank and HPSG, in Anne ABEILLÉ, editor, *Treebanks: Building and Using Parsed Corpora*, volume 20 of *Text, Speech and Language Technology*, pp. 351–365, Springer Netherlands.
- Phuong Thai NGUYEN, Luong Vu XUAN, Thi Minh Huyen NGUYEN, Van Hiep NGUYEN, and Phuong LE-HONG (2009), Building a large syntactically-annotated corpus of Vietnamese, in *Proceedings of the 3rd Linguistic Annotation Workshop, ACL-IJCNLP*, pp. 182–185, Suntec City, Singapore.
- Thi Luong NGUYEN, My Linh HA, Viet Hung NGUYEN, Thi Minh Huyen NGUYEN, and Phuong LE-HONG (2013), Building a treebank for Vietnamese dependency parsing, in *The 10th IEEE RIVF*, pp. 147–151, IEEE, Hanoi, Vietnam.

- Thi Minh Huyen NGUYEN, Laurent ROMARY, Mathias ROSSIGNOL, and Xuan Luong VU (2006), A lexicon for Vietnamese language processing, *Language Resources and Evaluation*, 40(3–4).
- Jaokim NIVRE and Ryan McDONALD (2008), Integrating graph-Based and transition-Based dependency parsers, in *Proceedings of ACL-08*, pp. 950–958, ACL, Columbus, Ohio, USA.
- Joakim NIVRE (2003), An efficient algorithm for projective dependency parsing, in *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pp. 149–160, Nancy, France.
- Jungyeul PARK (2006), Extraction of tree adjoining grammars from a treebank for Korean, in *Proceedings of COLING-ACL Student Research Workshop*, pp. 73–78, Morristown, New Jersey, USA.
- Patrick PAROUBEK, L. G. POUILLOT, I. ROBBA, and Anne VILNAT (2005), EASY : Campagne d'évaluation des analyseurs syntaxiques (EASY Evaluation campagne of syntactic parsers), in *Actes de Traitement Automatique des Langues*, pp. 3–12, Dourdan, France.
- Lewis M. PAUL, Gary F. SIMONS, and Charles D. Fennig (EDS.) (2014), *Ethnologue: Languages of the World, Seventeenth edition*, SIL International, Dallas, Texas, USA.
- Owen RAMBOW and Aravind JOSHI (1994), A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena, in *Current Issues in Meaning-Text Theory*, pp. 1–20, Pinter, London, UK.
- Azim ROUSSANALY, Benoît CRABBÉ, and Jérôme PERRIN (2005), Premier bilan de la participation du LORIA à la campagne d'évaluation EASY, in *Actes de Traitement Automatique des Langues*, pp. 49–52, Dourdan, France.
- Yves SCHABES (1990), *Mathematical and computational aspects of lexicalized grammars*, Ph.D. thesis, University of Pennsylvania, Pennsylvania, USA.
- Rion SNOW, Dan JURAFSKY, and Andrew Y. NG (2005), Learning syntactic patterns for automatic hypernym discovery, in *Advances in Neural Information Processing Systems*, pp. 1297–1304, Vancouver, Canada.
- VIETNAM COMMITTEE ON SOCIAL SCIENCES, editor (1983), *Vietnamese Grammar (in Vietnamese)*, NXB KHXH, Hanoi, Vietnam.
- Fei XIA (2001), *Automatic grammar generation from two different perspectives*, Ph.D. thesis, University of Pennsylvania, Pennsylvania, USA.
- Fei XIA, Martha PALMER, and Aravind JOSHI (2000), A uniform method of grammar extraction and its applications, in *Proceedings of the joint SIGDAT conference on empirical methods in NLP and very large corpora*, pp. 53–62, Morristown, New Jersey, USA.

*Phuong Le-Hong et al.*

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*

<http://creativecommons.org/licenses/by/3.0/>



# Implementing semantic frames as typed feature structures with XMG

Timm Lichte<sup>1</sup> and Simon Petitjean<sup>2</sup>

<sup>1</sup> University of Düsseldorf, Germany

<sup>2</sup> Université d'Orléans, LIFO, France

## ABSTRACT

This work<sup>1</sup> presents results on the integration of frame-based representations into the framework of eXtensible MetaGrammar (XMG). Originally XMG allowed for the description of tree-based syntactic structures and underspecified representations of predicate-logical formulae, but the representation of frames as a sort of typed feature structure, particularly type unification, was not supported. Therefore, we introduce an extension that is capable of handling frame representations directly by means of a novel <frame>-dimension. The aim is not only to make possible a straightforward specification of frame descriptions, but also to offer various ways to specify constraints on types, be it as a connected type hierarchy or a loose set of feature structure con-

*Keywords:*  
*metagrammar,*  
*typed feature*  
*structures,*  
*models of type*  
*constraints,*  
*type hierarchy,*  
*unification*

---

<sup>1</sup> This article is a substantially revised and extended version of Lichte *et al.* (2013). It has greatly benefited from discussions with Laura Kallmeyer, Rainer Osswald, Christof Rumpf and Yulia Zinova. We also thank both the reviewers of the ESSLLI 2013 workshop on High-level Methodologies in Grammar Engineering (HMGE) and the the reviewers of the JLM for detailed and helpful comments. The work presented in this paper was financed by the Deutsche Forschungsgemeinschaft (DFG) within the CRC 991. It was also financially supported by the scientific network PARSEME (COST Action IC1207) with a travel grant for Simon Petitjean. When finishing this article we learned with great sadness of the passing of our friend and colleague Christof Rumpf. We dedicate this article to his memory.

straints. The presented extensions to XMG are fully operational in a new prototype.

1

## INTRODUCTION

Recent work (Kallmeyer and Osswald 2012a,b, 2013; Zinova and Kallmeyer 2012) has shown increasing interest in coupling a frame-based semantics with a tree-based syntax such as Tree Adjoining Grammar (TAG, Joshi and Schabes 1997). While having led to promising results on the theoretic side, it is still unclear how to implement these ideas with existing grammar engineering tools, let alone how to bring them alive in natural language parsing. In this article, we present results on the integration of frame-based representations into the framework of eXtensible MetaGrammar (XMG, Crabbé *et al.* 2013).<sup>2</sup> XMG originally allowed for the description of tree-based syntactic structures and underspecified representations of predicate-logical formulae, but the representation of frames as a sort of typed feature structure, particularly type unification, was not supported. Therefore we extend XMG by a novel <frame>-dimension, among other tools, that makes it capable of handling frame representations as formalized in Petersen (2007) and Kallmeyer and Osswald (2013), i.e. as extended typed feature structures, directly.<sup>3</sup> This capability also paves the way for implementing recent work on morphological decomposition, such as in Zinova and Kallmeyer (2012), where morphemes are linked to a frame-semantic representation.

These efforts might seem redundant, given that there exists a multitude of works on dealing with typed feature structures in grammar implementation and parsing, notably in the framework of HPSG (e.g. Carpenter 1992; Götz *et al.* 1997; Malouf *et al.* 2000; Flickinger 2000; Copestake 2002; Carpenter *et al.* 2003). As far as we can see, however, our approach differs from previous work in several respects: first and

---

<sup>2</sup><https://sourcesup.renater.fr/xmg/>

<sup>3</sup>The reviewers suggested to focus more on typed feature structures and less on semantic frames. While this is a reasonable advice, we consider the application to be the driving force of this whole endeavour, which also crucially motivates our choice of a typed feature structure logic that is, from our perspective, uncommon in linguistic applications.



foremost, it allows for a choice between minimal and maximal models of type constraints. As a consequence of this, our approach also allows for anonymous types, hence types that result from the conjunction of defined types, but that are not defined themselves in the type signature. We will precisely explicate this distinction in Section 4. Finally, our work aims at providing the grammar writer with multiple means of expression, for example, in permitting him/her to make use of loose type constraints or connected type hierarchies, or both. This degree of flexibility is often not found in other grammar implementation frameworks that support typed feature structures. We hypothesize that other frameworks are mainly concerned with the syntactic component of the grammar, while we focus on the conceptual semantics whose structure (and how it comes about) seems to be much less predetermined.

The paper proceeds as follows. The next section briefly illustrates the grammatical objects that we are concerned with, and Section 3 then shows the proposed factorization, which crucially guides the implementation with XMG and also justifies the employment of frame unification. After this, we explain the formalization of frames as a sort of typed feature structure in Section 4, and the basic concepts of XMG in Section 5. This will be essential to understand the usage and compilation details of the new `<frame>`-dimension, the presentation of which follows in Section 6. Finally, Section 7 concludes the article. Moreover, a complete code example based on analyses from Section 2 and Section 3 is presented in the Appendix.

## 2 A FRAME-BASED SEMANTICS FOR LTAG

We will use the state-of-the-art work of Kallmeyer and Osswald (2013) on integrating frame semantics into Lexicalized Tree-Adjoining Grammars (LTAG) as a starting point. One important motivation for developing a frame-based semantics for LTAG is found in the straightforward account for the well-established distinction between lexical and constructional contributions to the overall meaning. An example of this sort of contrast is displayed in (1):

- (1) a. The ball rolled into the goal.
- b. John rolled the ball into the goal.

Both sentences involve the same verb of directed motion, *rolled*, but the two instances nevertheless differ with respect to the linking of the subject with the conveyed event semantics. While in (1a) the subject *the ball* is the moved object, in (1b) the subject *John* is causing the motion rather than undergoing it. The trigger for this semantic shift seems to be the lack or existence of the direct object (both under the presence of a directional PP), hence the construction type, as this constitutes the crucial syntactic difference between (1a) and (1b).

In the framework of Kallmeyer and Osswald (2013), the syntax of these constructions lies in the scope of the LTAG component. An LTAG consists of a finite set of phrase structure trees, the *elementary trees*, that can be combined (by means of two basic operations, substitution and adjunction) to generate larger trees.<sup>4</sup> Since we are dealing with a lexicalized TAG, each elementary tree must include at least one non-terminal leaf, the *lexical anchor*. Furthermore elementary trees are constrained through the valency properties of their anchors. Usually each non-terminal leaf corresponds to exactly one syntactic argument, and vice versa.

The proposed LTAG analyses of (1a) and (1b), shown in Figure 1, differ with respect to the elementary trees that are associated with the two instances of *rolled*, say the intransitive *rolled<sub>int</sub>* and the transitive *rolled<sub>tr</sub>*:<sup>5</sup> since syntactic arguments are represented as non-terminal leaves, the elementary tree for *rolled<sub>int</sub>* lacks the object NP slot that the elementary tree for *rolled<sub>tr</sub>* has. The difference in meaning is therefore attributed to the different elementary trees that a given verb may anchor.

Since elementary trees, but not the anchoring verbs, are held responsible for different linking patterns, it is straightforward to abstract away from the concrete anchor by just considering the yet unanchored elementary tree, which is commonly called the *tree template*. An example of such a tree template is shown on the left in Figure 2, wherein the site of lexical insertion, here of *rolled*, is marked by the  $\diamond$ -symbol. The

---

<sup>4</sup>Since in the present paper we mainly focus on single elementary trees, we skip most details of the formalism here. See Joshi and Schabes (1997) or Abeillé and Rambow (2000a) for comprehensive presentations.

<sup>5</sup>Note that dashed arrows indicate combinatorial operations, which in this case only involve substitution, i.e. the rewriting of a leaf node in the target tree.

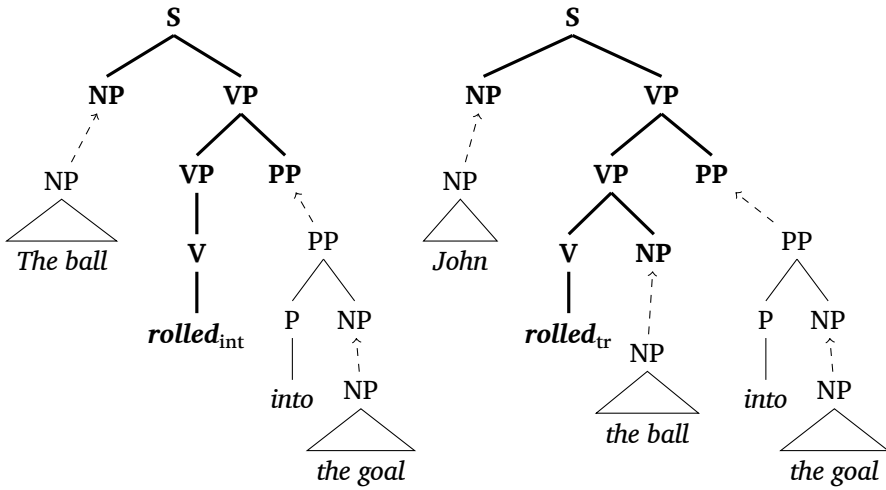


Figure 1: LTAG derivation for (1a) and (1b) with intransitive versus transitive *rolled*

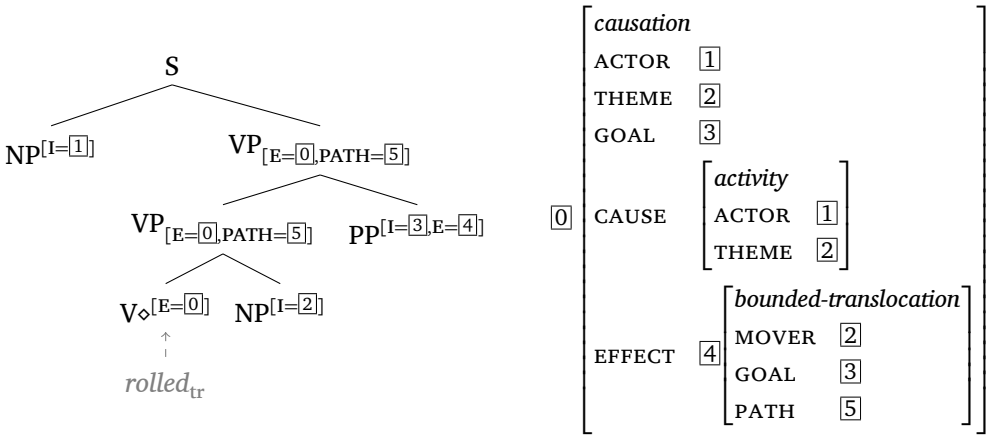


Figure 2: Tree template and frame-semantic representation of the transitive motion construction from Figure 1, taken from Kallmeyer and Osswald (2013, Fig. 26)

complex node labels will be explained presently. The right side of Figure 2 shows the event semantic contribution of the tree template in the format of a typed feature structure, here represented as an attribute value matrix (AVM). Typed feature structures are a common representation format of *frames* (see Petersen 2007), which, according to

Fillmore (1982), Barsalou (1992) and others, are considered a proper representation of mental concepts.<sup>6</sup> As can be seen from the example, features describe semantic participants and components (AGENT, THEME, ...), while feature structures correspond to conceptual objects, restricted by the type (*causation, activity, ...*) that they are associated with. The boxed numbers, finally, are *base labels* which serve to mark inequalities and correspondences in the syntax-frame interface. Furthermore, they guide the unification of (subparts of) frames, as can be seen in the next section.

Because tree templates, just as elementary trees, span an extended domain of locality,<sup>7</sup> the linking of positions within the tree template to positions within the frame-semantic representation can be achieved rather directly. In Figure 2 it is indicated by co-occurring boxed numbers. For example, the subject NP-leaf is linked with the ACTOR role(s) of the frame, eventually causing the unification of the ACTOR role and the frame of the substituting NP-tree. Note that the nodes of tree templates carry (non-recursive, non-typed) feature structures, which include, among others, interface features such as I(NDIVIDUAL) and E(VENT).<sup>8</sup> Following the terminology in Kallmeyer and Osswald (2013), we call couples of tree template and frame-semantic representation an *elementary construction*.

The composition of frame representations is moreover guided by a globally defined type hierarchy, which determines (i) the unifiability of types and the resulting type, and (ii) the set of appropriate features

---

<sup>6</sup>Note that FrameNet (Fillmore 2007), despite being declared as an implementation of Fillmore's frame semantics, deals with flat lexical frame representations that are generally less expressive (Osswald and Van Valin 2014).

<sup>7</sup>The extended domain of locality (EDL) is one of the central properties of the TAG formalism (cf. Joshi *et al.* 1990). It amounts to the capability of arguments to immediately attach to the elementary tree of their governor; see again Figure 1. In connection with TAG, EDL presupposes the availability of the adjunction operation (i.e. the rewriting of inner nodes), in order to account for discontinuity effects such as long distance dependencies.

<sup>8</sup>The approach to let the syntax-semantics interface rely on the unification of interface features can already be found in, e.g., Stone and Doran (1997), Frank and van Genabith (2001), Gardent and Kallmeyer (2003). The proposal for linking nodes of an elementary tree with positions in some semantic representation, and thus to derive syntax and semantics in parallel, dates back at least to Shieber and Schabes (1990).

(and their value types) of a type, the *appropriateness conditions*. Regarding event types, Kallmeyer and Osswald (2013) work with the partial type hierarchy in Figure 3. It has to be read top-down, with

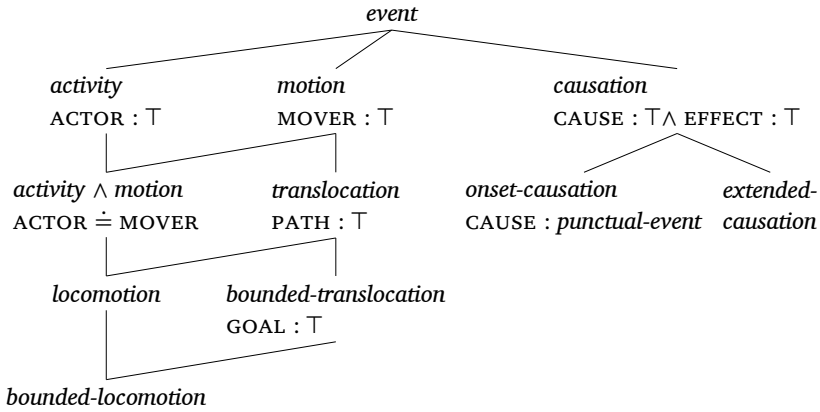


Figure 3: Partial type hierarchy for event types proposed in Kallmeyer and Osswald (2013, Fig. 16).

the more general types dominating the more specific types. Note that the type hierarchy may also contain anonymous types, e.g. *activity ^ motion*, which are usually not available in other frameworks. Roughly speaking, anonymous types make it possible to assign appropriateness conditions to a conjunction of types rather than just to a single type. The details will be explained in Section 4.

In this work we are neither concerned with the unification of frame representations following syntactic composition, nor with the preceding process of lexical insertion that triggers the unification of lexical and constructional frame components (see Kallmeyer and Osswald 2013, Fig. 13), but rather with the metagrammatical framework of XMG. Metagrammars are a tool to describe static elementary constructions such as in Figure 2, consisting of a tree template and a fixed typed feature structure. It might therefore seem unnecessary to employ all aspects of typed feature structures in metagrammars, particularly unification. However, this assumption is not warranted. Unification of types and feature structures is also found in the metagrammatical domain once the factorization of elementary constructions of the kind in Figure 2 is taken into account. This will be shown in the next section.

## FACTORIZATION OF TREE TEMPLATES AND FRAMES

Richly structured grammatical objects like those in Figure 2 make necessary some kind of metagrammatical factorization, once a large coverage grammar gets compiled and maintained (Xia *et al.* 2010). Metagrammatical factorization is a process to define recurring sub-components of grammatical objects, which can then be combined in at least two ways: in a transformation-based fashion, known as the metarule approach (Becker 1994, 2000; Prolo 2002), or in a purely constraint-based, monotonic fashion as is the case in XMG (following Candito 1996). In addition to the benefit in terms of grammar engineering, however, Kallmeyer and Osswald (2012a,b, 2013) claim that metagrammar factorization can be also used to reflect constructional analyses in the spirit of Construction Grammar (Kay 2002; Goldberg 2006). By this perspective, both the lexical material and the “constructions” used contribute meaning.

Taking these two aspects into account, Kallmeyer and Osswald (2013) propose to factorize the tree template and the frame in Figure 2 along the lines of Figure 4.<sup>9</sup> Boxes stand for the resulting factors or classes (i.e. classes in the sense of XMG), consisting of descriptions of a tree and a frame fragment. The inclusion relation between boxes is to be understood as a representation of inheritance or instantiation, so that the class of the comprising box inherits from, or instantiates, the class of the included box. Double edges (indicating identity constraints over nodes or base labels), dashed edges (non-strict dominance),  $<^*$  (non-strict precedence), and  $\vee$  (disjunction) are elements of the description language. Figure 4 then illustrates that the tree-frame couple in Figure 2 is a model of the class `n0Vn1pp(dir)`, which combines the classes `n0Vn1` and `DirPrepObj`. Combining two classes essentially means that all associated information is unified, from which a minimal model is resolved (see Section 5). Note that Figure 4 shows only a part of the proposed factorization. For example, the class `n0Vn1`, also taken from Kallmeyer and Osswald (2013, Fig. 4), results from combining three other classes (`Subj`, `VSpine`, `DirObj`), as shown in Figure 5.<sup>10</sup> Furthermore note that the class `n0Vn1pp(dir)` bears a con-

---

<sup>9</sup>The boxes-and-pipes notation in Figures 4 and 5 is of our invention.

<sup>10</sup>Kallmeyer and Osswald (2013) conjecture that class `n0V` has exactly one

Implementing semantic frames with XMG

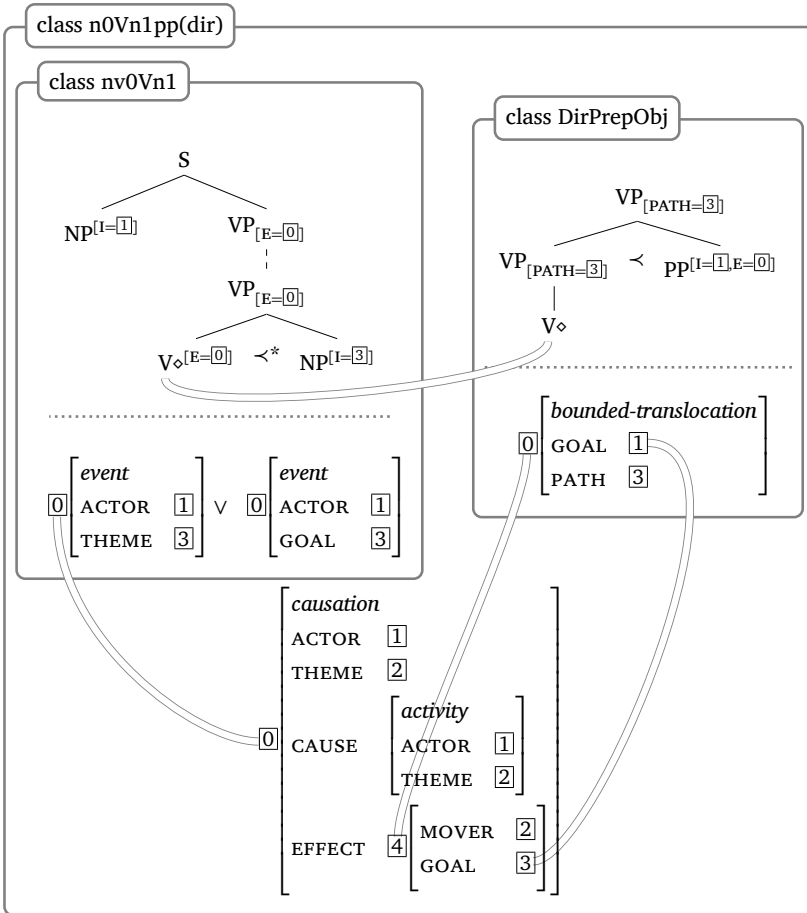
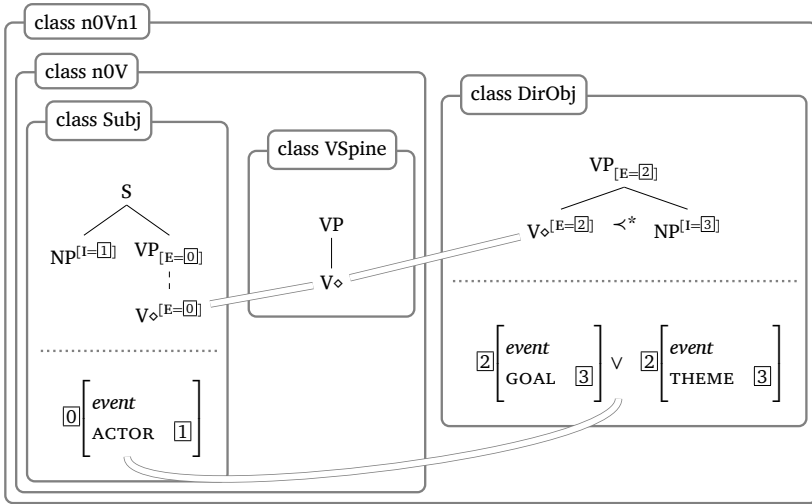


Figure 4: Metagrammatical factorization of the elementary construction from Figure 2. Boxes stand for the resulting factors or classes. Double edges indicate identity constraints, dashed edges indicate non-strict dominance,  $<^*$  is non-strict precedence, and  $\vee$  is disjunction.

minimal model, which only holds, however, if minimal models are said to consist of the smallest set of nodes that satisfy the description. In XMG, by contrast, minimal models just consist of a set of nodes mentioned in the description, and nothing more. Hence, from point of view of XMG,  $n0V$  has two minimal models, one with one VP node, and one with two VP nodes. In order to resolve only the former one, one could additionally apply polarization, or “colours” (Crabbé and Duchier 2005), onto the nodes. Fortunately, the question how many minimal models there are is irrelevant for the point made here and throughout the article.

Figure 5:  
Metagrammatical  
factorization for  
the transitive  
elementary  
construction.



structional facet: it only contributes to the frame representation, but no tree descriptions.

Now the question is whether the combination of frame representations, following the combination of two classes, should be considered a unification of typed feature structures. It is not hard to see that this is justified. A particularly good example is found in the combination of the frame representations of *n0Vn1* and *n0Vn1pp(dir)* in Figure 4. Even though they have different types, namely *event* and *causation*, the resulting type is supposed to be *causation*. According to the type hierarchy in Figure 3, *causation* is a subtype of *event*. Hence this is in line with regular type unification, but exceeds a plain union of feature structures. Therefore the unification of typed feature structures should be already supported at the metagrammatical level. The same holds for the verification of appropriateness conditions, of course. Note that, as can be seen from Figure 4, the flexibility of frame unification is fostered by base labels when unifying the frame of *DirPrepObj* with a subpart of the frame of *n0Vn1pp(dir)*.

The graphic representation of the metagrammatical factorization in Figure 4 and Figure 5 remains at a rather informal level and the question arises: how could this be translated into XMG code? Since the original XMG did not give a satisfying answer due to reasons of usability and completeness (which we will cover in Section 5) we will



develop and justify a new <frame>-dimension in Section 6. But first we will review the underlying notions: frames as typed feature structure with base labels, type hierarchies and unification.

#### 4 BASE-LABELLED TYPED FEATURE STRUCTURES: FORMAL DEFINITIONS

In what follows we largely, but not exclusively, adhere to the definitions in Kallmeyer and Osswald (2013), while streamlining them according to our terminology and taste. What is not needed from Kallmeyer and Osswald (2013), however, are (non-functional) relations, because they do not appear in the frame representations of elementary constructions, other than, e.g., in the frame representations of the anchor lexicon (Kallmeyer and Osswald 2013, Fig. 13).<sup>11</sup> So we will ignore the notion of (non-functional) relations, and instead dwell on type inference and the minimal and maximal models of feature structure constraints. It is this choice between minimal and maximal models, as well as the general availability of anonymous types, which makes the presented extension to XMG particularly flexible at describing a type system. And it is this flexibility, among other things, which sets our extension to XMG apart from current frameworks for HPSG.

Let us start with the building blocks of typed feature structures, which are settled in the *signature*:

**Definition 1 (Signature)** *A signature is a tuple  $\langle A, T, B \rangle$  with a finite set of attributes (or features)  $A$ , a finite set of elementary types  $T$ , and an infinitely countable set of base labels  $B$ .*

The set of elementary types is accompanied by special types  $\top$  and  $\perp$  that are useful in feature structure descriptions and feature structure constraints (see below).  $\top$  is the most general type, unifiable with every other type, whereas  $\perp$  is unifiable with none of them. Within Boolean expressions,  $\top$  corresponds to ‘true’ and  $\perp$  to ‘false’. *Base labels* are commonly boxed natural numbers, thus  $B = \{\boxed{0}, \boxed{1}, \boxed{2}, \dots\}$ .

Based on a given signature, typed feature structures are defined as follows:

---

<sup>11</sup> Kallmeyer and Osswald (2013) make use of the part-of relation when dealing with directional prepositions such as *to*, *into*, and *along*.

**Definition 2 (Base-labelled typed feature structure)** *Given a signature  $\langle A, T, B \rangle$ , a base-labelled typed feature structure is a tuple  $\langle V, \delta, \tau, \beta \rangle$  with*

- $V$ , a finite set of nodes,
- $\delta : (V \times A) \rightarrow V$ , a partial transition function,
- $\tau : V \rightarrow 2^T$ , a total typing function, and
- $\beta : B \rightarrow V$ , a partial base-labelling function.

We call types in  $2^T$ , i.e. elements of the powerset of  $T$ , *conjunctive types* in order to distinguish them from elementary types.

We follow Kallmeyer and Osswald (2013) in that we do not specify a type hierarchy immediately, but treat it as a model of feature structure constraints, i.e. generalized feature structure descriptions.<sup>12</sup> Therefore the following notations can be directly borrowed from Kallmeyer and Osswald (2013, (3)), omitting only those parts of their definition that deal with relations. Note that we extend  $\delta$  by feature paths in the usual way.<sup>13</sup>

**Definition 3 (Unlabelled feature structure description)** *Let  $\langle V, \delta, \tau, \beta \rangle$  be a feature structure over the signature  $\langle A, T, B \rangle$  with  $v, w \in V$ ,  $t \in 2^T$  and  $p, q \in A^+$ , then the satisfaction relation  $\models$  between nodes and feature structure descriptions is defined as follows:*

- $v \models t$                     *iff*  $t \in \tau(v)$
- $v \models p : t$                 *iff*  $\delta(v, p) \models t$
- $v \models p \doteq q$               *iff*  $\delta(v, p) = \delta(v, q)$
- $\langle v, w \rangle \models p \triangleq q$     *iff*  $\delta(v, p) = \delta(w, q)$

As such, unlabelled feature structure descriptions apply to nodes of a feature structure. For example, the root node of the causation frame in Figure 4 satisfies the following conjunction of descriptions:

$$(2) \quad \textit{causation} \wedge \textit{ACTOR} \doteq \textit{CAUSE ACTOR} \wedge \textit{THEME} \doteq \textit{CAUSE THEME} \wedge \\ \textit{THEME} \doteq \textit{EFFECT MOVER} \wedge \textit{CAUSE} : \textit{activity} \wedge \textit{EFFECT GOAL} : \textit{T}$$

By adding base labels, we can explicitly assign descriptions to nodes within a feature structure. Therefore, in contrast to unlabelled ones,

---

<sup>12</sup>However, XMG also allows one to specify a type signature explicitly; see Section 6.

<sup>13</sup> $\delta(v, p \ a)$  with  $p \in A^+$  and  $a \in A$  is a shorthand for  $\delta(\delta(v, p), a)$ .

the labelled feature structure descriptions are satisfied by feature structures as a whole:<sup>14</sup>

**Definition 4 (Labelled feature structure description)** *Let  $F = \langle V, \delta, \tau, \beta \rangle$  be a feature structure over the signature  $\langle A, T, B \rangle$  with  $p, q \in A^+$  and  $l, k \in B$ , and let  $\phi$  be an unlabelled feature structure description. The satisfaction relation  $\models$  between feature structures and labelled feature structure descriptions is defined in the following way:*

- $F \models l \phi$                       iff  $\beta(l) \models \phi$
- $F \models l p \triangleq k q$             iff  $\langle \beta(l), \beta(k) \rangle \models p \triangleq q$

The unlabelled descriptions in (2) can be straightforwardly labelled in order to be satisfied by the causation frame as a whole:

$$(3) \quad \boxed{\text{causation}} \wedge \boxed{\text{ACTOR}} \dot{=} \text{CAUSE ACTOR} \wedge \boxed{\text{THEME}} \dot{=} \text{CAUSE THEME} \wedge \boxed{\text{THEME}} \dot{=} \text{EFFECT MOVER} \wedge \boxed{\text{CAUSE : activity}} \wedge \boxed{\text{EFFECT GOAL : T}}$$

Note that  $\boxed{\text{ACTOR}} \dot{=} \text{CAUSE ACTOR}$  is equivalent with  $\boxed{\text{ACTOR}} \triangleq \boxed{\text{CAUSE ACTOR}}$  (see Kallmeyer and Osswald 2013, fn. 14). We adopt the convention to write  $l$  instead of  $l \epsilon$  for some label  $l$ . This allows us to write  $\boxed{1} \triangleq \boxed{2}$  for expressing the value identity of labels  $\boxed{1}$  and  $\boxed{2}$ .

Concerning the subsumption relation on feature structures, we can transfer the fairly standard definition from Kallmeyer and Osswald (2013), which they extend by base labels.

**Definition 5 (Subsumption,  $\sqsubseteq$ )** *Given feature structures  $F_1 = \langle V_1, \delta_1, \tau_1, \beta_1 \rangle$  and  $F_2 = \langle V_2, \delta_2, \tau_2, \beta_2 \rangle$  over the signature  $\langle A, T, B \rangle$ ,  $F_1$  subsumes  $F_2$ , if there is a function  $h : V_1 \rightarrow V_2$  so that*

- if  $\delta_1(v, a)$  is defined for  $v \in V_1$  and  $a \in A$ , then  $\delta_2(h(v), a) = h(\delta_1(v, a))$ ;
- for every  $v \in V_1$ ,  $\tau_1(v) \subseteq \tau_2(h(v))$ ;
- if  $\beta_1(l)$  is defined for  $l \in B$ , then  $\beta_2(l) = h(\beta_1(l))$ .

As usual, the definition of unification builds on subsumption:

---

<sup>14</sup>From the general shape of labelled feature structure descriptions it follows that they can only describe feature structures where every node is reachable from some labelled node via a (potentially empty) attribute path. In fact, this is why these labels are called base labels.

**Definition 6 (Unification,  $\sqcup$ )** Let  $F_1, F_2, F_3$  be feature structures.  $F_3$  is the result of the unification of  $F_1$  and  $F_2$ , iff  $F_3$  is the least specific feature structure such that  $F_1 \sqsubseteq F_3$  and  $F_2 \sqsubseteq F_3$ .

The specificity of feature structures is determined by the number of nodes, the specificity of the assigned types and the size of the base labelling function. Note that, in cases of unification such as in Figure 4, feature structures (and corresponding interface variables) are relabelled beforehand, so that they come with disjoint sets of labels. The identity of certain labels can then be imposed through additional identity statements.

Feature structure constraints consist of universally quantified, unlabelled features structure descriptions, i.e. descriptions that hold for every node in the feature structure. Kallmeyer and Osswald (2013) restrict them to Horn clauses for reasons of tractability.<sup>15</sup>

**Definition 7 (Feature structure constraint)** Given unlabelled feature structure descriptions  $\phi_1, \dots, \phi_n, \psi$ , a feature structure constraint is  $\phi_1 \wedge \dots \wedge \phi_n \preceq \psi$ , which is equivalent to the universally quantified implication  $\forall(\phi_1 \wedge \dots \wedge \phi_n \rightarrow \psi)$  where quantification is over the nodes of the described feature structure.

The following list shows different kinds of feature structure constraints (mostly taken from Kallmeyer and Osswald 2013, (12)):<sup>16</sup>

- (4) a. *activity*  $\preceq$  *event*
- b. *causation*  $\preceq$   $\neg$ *activity*  $\iff$  *causation*  $\wedge$  *activity*  $\preceq$   $\perp$
- c. *locomotion*  $\preceq$  *activity*  $\wedge$  *translocation*  
 $\iff$  (*locomotion*  $\preceq$  *activity*)  $\wedge$  (*locomotion*  $\preceq$  *translocation*)
- d. *activity*  $\preceq$  ACTOR:  $\top$
- e. AGENT:  $\top \preceq$  AGENT  $\doteq$  ACTOR

The first three feature structure constraints in (4a)–(4c) consist of descriptions over types, which is why we call them *type constraints*. The simplest type constraint in (4a) relates two elementary types and cor-

<sup>15</sup>Horn clauses are disjunctive clauses with at most one non-negative literal, all others being negative. Hence they are of the form  $\neg\phi_1 \vee \dots \vee \neg\phi_n \vee \psi$ , which has the implicational equivalent  $\phi_1 \wedge \dots \wedge \phi_n \rightarrow \psi$ .

<sup>16</sup>Note that  $\iff$  in (4) is not part of the description language but indicates the equivalence of descriptions.

responds to an ISA-relation, namely *activity* being also an *event*. An ISNOTA-relation can be expressed by type constraints of the kind in (4b). Note that the use of negation in the consequent is equivalent to a clause with a conjunctive antecedent and a ‘false’ ( $\perp$ ) consequent. Not only can the antecedent be complex, but also the consequent, even though this is reducible to a conjunction of simpler implications, as shown in (4c). The last two examples in (4d) and (4e) contain attribute-value terms. Constraint (4d) represents an *appropriateness condition*, namely a condition on the type *activity* concerning the value type of its attribute ACTOR. Finally, (4e) adds a constraint about the token identity of the values of the attributes AGENT and ACTOR (if AGENT has a value). Constraints of this sort, where every conjunct consists of a feature-value description or a path equation, can be characterized as *feature-value constraints*.<sup>17</sup>

A crucial decision concerns the model of feature structure constraints, as it can be seen as maximal or minimal. The difference is made visible in Figure 6 with type hierarchies based on  $\preceq$ .

$$T = \{a, b, c\}$$

$$b \preceq a$$

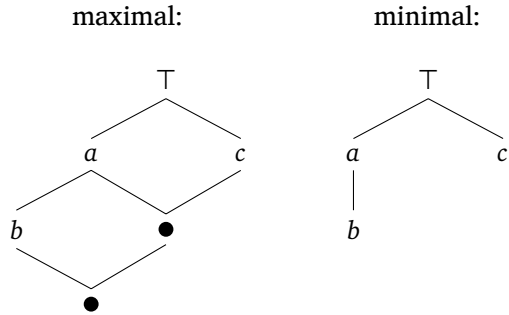


Figure 6: Examples of a maximal and a minimal model of type constraint

Roughly speaking, a maximal model of type constraints allows for *anonymous types* (indicated with dots in Figure 6) as long as they are not explicitly excluded. With minimal models it is the other way

<sup>17</sup> Other sorts of feature structure constraints, for example those in (i), are not discussed in Kallmeyer and Osswald (2013):

- (i) a.  $p : t_1 \preceq t_2$
- b.  $p : t_1 \wedge t_2 \preceq t_3$

We therefore concentrate on feature structure constraints in the shape of type constraints, appropriateness conditions and feature-value constraints.

around: anonymous types are forbidden as long as they are not explicitly allowed. In both cases they could be introduced (or excluded) by constraints such as in (5a), where the antecedent consists of a conjunction of elementary types. In Figure 6,  $a \wedge c$  and  $a \wedge b \wedge c$  are anonymous types. It is shown in (5b) that these anonymous types can bear feature-value constraints as well.<sup>18</sup>

- (5) a.  $a \wedge c \preceq \top$   
 b.  $action \wedge motion \preceq \text{ACTOR} \doteq \text{MOVER}$   
 (Kallmeyer and Osswald 2013, (12e))

The preceding remarks were concerned with elementary types. However, as we learned in Definition 2, the typing function  $\tau$  assigns sets of elementary types, the conjunctive types. They are included for good reason, since they help to treat elementary types and anonymous types in a uniform way, and eventually to facilitate the definition and implementation of unification.<sup>19</sup> In the following, we treat conjunctive types as a model of type constraints, and, on this basis, explicate what it means to be minimal or maximal.

**Definition 8 (Model of type constraints)** *Given type constraints  $TC$  over a signature  $\langle A, T, B \rangle$ , a set  $\hat{T}$  of conjunctive types over  $T$  is a model of  $TC$ , if  $\hat{T}$  satisfies the type constraints from  $TC$  in the following way:*

- $\hat{T} \models t_1 \wedge \dots \wedge t_n \preceq \perp$  iff  $\{t_1, \dots, t_n\} \not\subseteq \hat{t}$  for all  $\hat{t} \in \hat{T}$ ;
- $\hat{T} \models t_1 \wedge \dots \wedge t_n \preceq t_m$  iff if  $\{t_1, \dots, t_n\} \subseteq \hat{t}$ , then  $\{t_m\} \subseteq \hat{t}$ .

The model  $\hat{T}$  of given type constraints is said to be *maximal*, if  $\hat{T}$  is the largest set of conjunctive types that satisfies them. On the other

---

<sup>18</sup>Note that anonymous types are generally ruled out in major implementation tools for HPSG such as LKB (Copestake and Flickinger 2000; Copestake 2002) and TRALE (Götz *et al.* 1997; Carpenter *et al.* 2003), from which it follows that models from type constraints are always minimal.

<sup>19</sup>Note however that Carpenter (1992, 23–25) defines conjunctive types differently as he adds the condition that  $p$ , but not  $q$ , is included whenever the relation  $p \preceq q$  is considered. Hence, following his conception, there is no co-occurrence of elementary types and their elementary subtypes within a conjunctive type. On the other hand, our definition of conjunctive type rather coincides with the notion of “conjunctive concepts” in Carpenter and Pollard (1991), or with the notions of “entity types” and “generic entities” in Osswald (2003, 24f).

hand,  $\hat{T}$  is said to be the *minimal* model, if  $\hat{T}$  includes just those conjunctive types of the maximal model that correspond (i) to elementary types, and (ii) to conjunctive types that make up the left side (i.e. the condition) of a type constraint.

The type hierarchies from Figure 6 for elementary types are repeated in Figure 7 for conjunctive types. They are now based on  $\subseteq$  rather than  $\preceq$ . We say that conjunctive type  $\hat{t}_1$  is the subtype of  $\hat{t}_2$ , if  $\hat{t}_1 \supseteq \hat{t}_2$ . The definition of the most general common subtype is equally simple.

$$T = \{a, b, c\}$$

$$b \preceq a$$

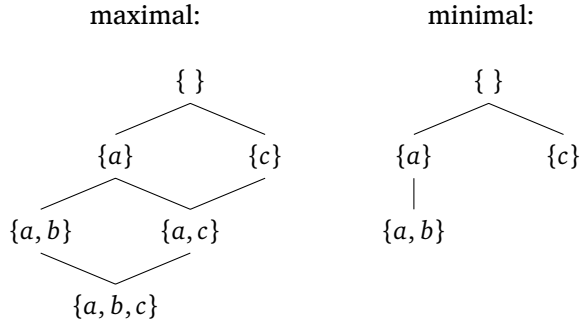


Figure 7: Example of maximal and minimal models of type constraints over conjunctive types

**Definition 9 (Most general common subtype)** *The most general common subtype of two conjunctive types  $\hat{t}_1, \hat{t}_2 \in \hat{T}$  is the smallest set  $\hat{t}_3 \in \hat{T}$  such that  $\hat{t}_1 \cup \hat{t}_2 \subseteq \hat{t}_3$ .*

It is easy to see that, whenever type constraints have the shape of Horn clauses, the maximal model over conjunctive types  $\hat{T}$  forms a meet semi-lattice (a bounded complete partially ordered set)  $\langle \hat{T}, \subseteq \rangle$ . Hence, there is a most general common subtype for any two unifiable types. This does not necessarily hold for minimal models that are confined to elementary and certain anonymous types.<sup>20</sup> In order to obtain uniqueness of the most general common subtype for any two unifiable types, it can be necessary to add further anonymous types to  $\hat{T}$  (the Dedekind-McNeille completions). These completions are computed by the compiler that will be described in Section 6.3.2.

<sup>20</sup>For example, say  $T = \{a, b, c, d\}$ ,  $TC = \{c \preceq a \wedge b, d \preceq a \wedge b\}$ , then  $\hat{T} = \{\{\}, \{a\}, \{b\}, \{a, b, c\}, \{a, b, d\}\}$  satisfies  $TC$  and only contains the conjunctive types that correspond to elementary types in  $T$ . However,  $\{a\}$  and  $\{b\}$  have more than one most general common subtype in  $\hat{T}$ , namely  $\{a, b, c\}$  and  $\{a, b, d\}$ .

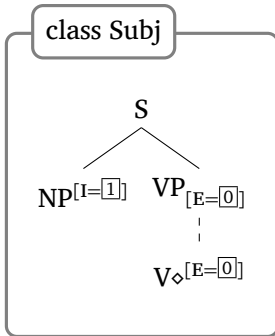
A conjunctive type  $\hat{t}$  corresponds to an elementary type  $t$ , if  $\hat{t}$  is the smallest set that contains  $t$  according to the model. Otherwise, if there is no  $t$  to which  $\hat{t}$  corresponds,  $\hat{t}$  is an anonymous type. Similarly,  $\hat{t}$  corresponds to some conjunctive type  $\hat{t}'$ , if  $\hat{t}$  is the smallest set of a model that contains  $\hat{t}'$ . Moreover, it can be useful to also express subtype relations among elementary types  $t_1, t_2 \in T$ . We say that  $t_1$  is the subtype of  $t_2$ , if for  $\hat{t}_1, \hat{t}_2 \in \hat{T}$  that correspond to  $t_1$  and  $t_2$  it holds that  $\hat{t}_1$  is a subtype of  $\hat{t}_2$ .

Before proceeding to the next section, two further aspects of the feature logic used here and in Kallmeyer and Osswald (2013) should be mentioned. Firstly, it cannot account for reverse type constraints, which refer to dominating nodes (Rainer Osswald, personal communication, July 23, 2014). A case in point is the set of relational concepts such as the *mother* example from Petersen and Osswald (2014, Fig. 11.5), where *mother* is said to constrain the existence of a dominating node, connected with a MOTHER edge. Hence, if *mother* was treated as a type (a subtype of, e.g., *person*), this constraint could not be expressed, at least not as a part of its appropriateness conditions. Secondly, what seems to be missing so far from the definition in Petersen (2007) is the notion of the central node of a frame, i.e. the node that tells us what the frame “represents” or “refers to”. We think, however, that the notion of central nodes is reflected, and generalized, in the present formalization by the notion of base labels. By reappearing in the interface features on the syntactic side, they serve to connect frame nodes with linguistic entities, and nothing else seems to be expressed by central nodes.

## 5 A BRIEF INTRODUCTION TO XMG

XMG (eXtensible MetaGrammar, Crabbé *et al.* 2013) stands both for metagrammatical descriptions and the compiler for these descriptions. Such descriptions are organized into classes that can be reused (i.e. “imported” or instantiated) by other classes. Borrowing from object oriented programming, classes are encapsulated, which means that each class can handle the scopes of their variables explicitly, by declaring variables and choosing which ones to make accessible for (i.e. to “export to”) other instantiating classes. The namespace of a class is





```

class Subj
...
<syn>{
 node ?S [cat=s];
 node ?SUBJ [cat=np,
 top=[i=?1]];
 node ?VP [cat=vp, bot=[e=?0]];
 node ?V (mark=anchor)
 [cat=v, top=[e=?0]];
 ?S->?SUBJ; ?S->?VP; ?VP->?*?V;
 ?SUBJ>>?VP
}
...

```

Figure 8:  
The <syn>-  
dimension of  
class Subj

then composed of the declared variables and all the variables exported by the imported classes.

*Dimensions* are the crucial elements of a class. They can be equipped with specific description languages and are compiled independently, thereby enabling the grammar writer to treat the levels of linguistic information separately. The standard dimensions are <syn> for the syntax, and <sem> for the semantics.<sup>21</sup>

The <syn>-dimension allows one to describe TAG tree templates (or fragments thereof). An example is shown in Figure 8 for the <syn>-dimension of class Subj from Figure 4. It includes two sorts of statements, namely those like ‘node ?S [cat=s]’ that instantiate nodes of the trees, and those like ‘?S->?SUBJ’ which determine the relative position of two nodes in the trees by referring to dominance and linear precedence.<sup>22</sup> Note that variable names are prefixed with a question mark (?). The <sem>-dimension, on the other hand, includes descriptions of a different language, for which a different compiler is used. Since this could be a candidate for hosting frame descriptions, we will have a look at <sem> more closely below. Different as they may be, one crucial commonality of all the dimensions pertains to the joint access

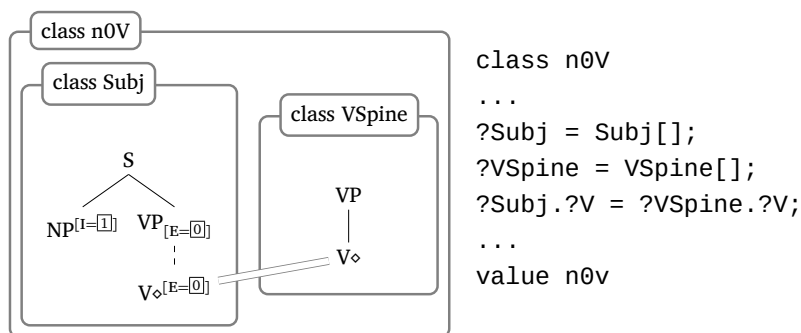
<sup>21</sup> Crabbé *et al.* (2013, 601) also mention the fairly technical dimension <dyn>, more commonly called *interface*, which helps to express the coreference of variables from different dimensions. Recently Duchier *et al.* (2012) have introduced a dimension for morphology; see also Lichte *et al.* (2013).

<sup>22</sup> There is also available a notational alternative with bracket structure.

to local variables declared in the same class. These shared variables constitute a direct interface between otherwise separated dimensions.

The combination of classes takes place outside the <syn>- and <sem>-dimensions. Figure 9 shows an example where the two classes Subj and VSpine are reused by the class n0V. First Subj and VSpine are instantiated and assigned a variable, then the encapsulated, yet exported, variables from Subj and VSpine can be accessed via the dot operator (e.g. to impose identity).

Figure 9:  
Example of  
the combination  
of classes



When the metagrammar is compiled, first a set of descriptions for each class under evaluation (triggered by value statements such as in Figure 9) is accumulated, and then the accumulated descriptions are resolved to yield minimal models. In the case of <syn>, the solver computes tree templates as minimal models, which is to say that only those nodes that are mentioned in the description are included. The final result can be explored with a viewer, or exported as an XML file in order to use it for parsing (e.g. with the TuLiPA parser, Kallmeyer *et al.* 2008).

### Frame descriptions in the <sem>-dimension?

As mentioned before, the <sem>-dimension is designed to contain underspecified, flat formulae of predicate logic (borrowing from Bos 1996). In fact, it is rather straightforward to reformulate frame descriptions in first-order predicate logic (Kallmeyer and Osswald 2013, Sec. 3.3.3). Concerning the signature, attributes can be represented with two-place predicates, while types can be seen as one-place predicates. The functionality of attributes is imposed by the following axiom for all  $a \in A$ , given some signature  $\langle A, T \rangle$ :

- (6)  $\forall x \forall y \forall z (a(x, y) \wedge a(x, z) \rightarrow y = z)$   
 (Kallmeyer and Osswald 2013, (6a))

The reformulation of feature structure descriptions and feature structure constraints is equally unproblematic:

- (7) a.  $p : t$   $\lambda x \exists y (p(x, y) \wedge t(y))$   
 b.  $p \doteq q$   $\lambda x \exists y (p(x, y) \wedge q(x, y))$   
 c.  $a p$   $\lambda x \lambda z \exists y (a(x, y) \wedge q(y, z))$   
 (Kallmeyer and Osswald 2013, (7))

- (8) a.  $t_1 \leq t_2$   $\forall x (t_1(x) \rightarrow t_2(x))$   
 b.  $t_1 \leq p : t_2$   $\forall x \exists y (t_1(x) \rightarrow p(x, y) \wedge t_2(y))$   
 c.  $t \leq p \doteq q$   $\forall x \exists y (t(x) \rightarrow p(x, y) \wedge q(x, y))$

Following this approach, a frame representation such as  $\text{ACTOR}[\text{?0, ?1}]$  would be translated into the two-place predicate  $actor(?0, ?1)$ , using regular XMG variables. A more detailed example based on the class  $nOVn1pp(\text{dir})$  is shown in Figure 10.

While the reformulation of frame descriptions via two-place predicates is straightforward, it is far less obvious how to account for feature structure constraints and the axiom of functionality. As far as type constraints and the type hierarchy are concerned, there seems to be a chance to simulate them with sets of type predicates. This approach is pursued in Figure 10. The construction of those *type simulating sets* (TSSs), as we call them, could proceed as follows: given a signature  $\langle A, T \rangle$  and a type hierarchy  $\mathcal{T}$  such as the one in Figure 10, we say that  $t \in T$  is simulated by the minimal set of predicates  $P_t(?X)$  for some variable  $?X$ , if  $P_t(?X)$  is assembled in the following way: for every  $t' \in T$ , if  $t'$  reflexively and transitively dominates  $t$  in  $\mathcal{T}$ , then  $t'(?X) \in P_t(?X)$ ; else if  $t$  and  $t'$  have no common subtype, then  $\sim t'(?X) \in P_t(?X)$ , where ‘ $\sim$ ’ stands for negation. To give an example,  $P_{\text{locomotion}}(?X)$  for the type *locomotion* in the type hierarchy of Figure 10 would be the set  $\{\text{activity}(?X), \text{motion}(?X), \sim \text{causation}(?X), \text{locomotion}(?X)\}$ . It is easily seen that the size of some  $P_t(?X)$  crucially depends on the position of  $t$  in  $\mathcal{T}$ , and on the size of  $T$ . Note that TSSs do not fully match conjunctive types, due to the insertion of negated type predicates.

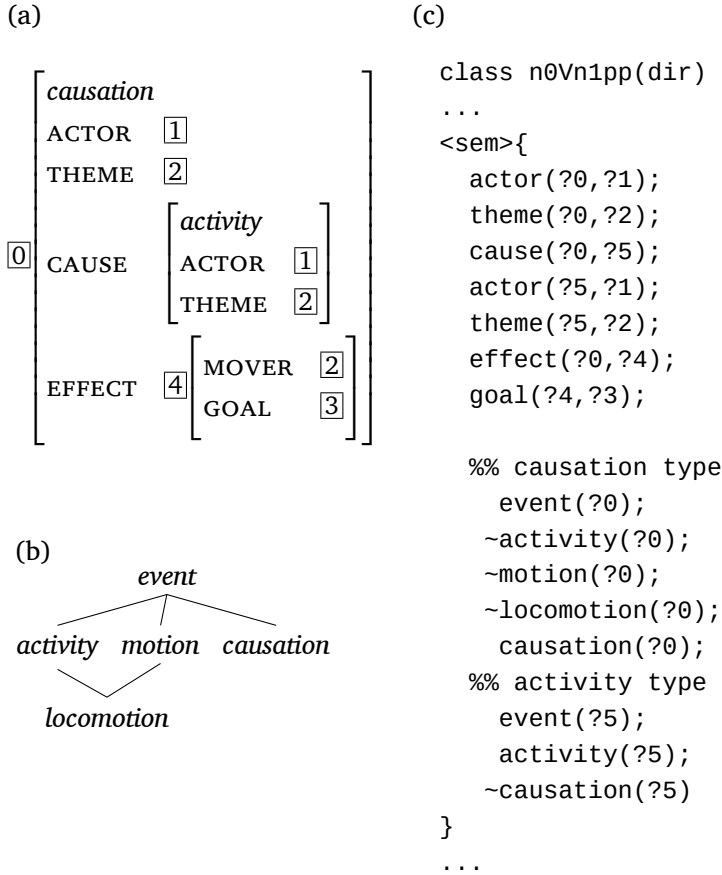


Figure 10: The feature structure of *n0Vn1pp(dir)* (repeated from Figure 4), the global type hierarchy (partially repeated from Figure 2), and its reformulation inside the *<sem>*-dimension

One basic problem of this approach is that so far XMG does not interpret the predicates of the *<sem>*-dimension, but merely accumulates them for later use. Hence XMG allows for, e.g., the coexistence of predicates *theme(x<sub>1</sub>, x<sub>2</sub>)* and *theme(x<sub>1</sub>, x<sub>3</sub>)* with *x<sub>2</sub> ≠ x<sub>3</sub>*, which conflicts with the required functionality of feature predicates. But even if XMG was enhanced to verify the functionality of predicates, at least three disadvantages would remain: (i) TSSs have to be provided by the grammar writer, (ii) they have to be included in the XMG descriptions as a whole, and (iii) unifying sister types with a common subtype will yield a TSS that does not immediately reveal the elementary type of

the common subtype. The latter disadvantage might be more of an aesthetic kind, but the first and the second one clearly have an impact on usability. Modifying the type hierarchy in the context of a large grammar would make necessary a meta-metagrammar, that would automatically recompute the TSSs and adapt the parts of the XMG descriptions, where TSSs were used. Rather than considerably modifying and extending the solver of the <sem>-dimension, let alone the available description language, we present a novel <frame>-dimension in the next section, which is closely adjusted to the peculiarities of frame representations and frame composition.

## 6 THE IMPLEMENTATION OF TYPED FEATURE STRUCTURE DESCRIPTIONS AND CONSTRAINTS

Implementation entails two operations: specification and compilation. We will deal with the first one in Section 6.2, when introducing specification languages for feature structure descriptions and constraints. The compilation of frame models based on these specifications is then covered in Section 6.3. First, however, the technical prerequisites for extending the XMG compiler are to be outlined.

### 6.1 *Architecture and extensibility of XMG*

The XMG project started in 2003 with the goal of providing a means to write large scale tree-based grammars, i.e. Tree Adjoining Grammars and Interaction Grammars (Perrier 2000). Originally, the compiler was written in Oz/Mozart, a language which is not maintained anymore. For this reason, and in order to build a compiler more in line with the project's ambitions (regarding modularity and extensibility), it was necessary to restart the implementation from scratch. The new implementation started in 2010 and is sometimes called XMG-NG or XMG2.<sup>23</sup> The compiler is now written in YAP (Yet Another Prolog) with bindings to Gecode for solving constraints. The extensibility is provided by automatic code generation using Python.

With this new version, an XMG compiler can be built from a combination of elementary compiler units. These elementary units are

---

<sup>23</sup> See <https://sourcesup.renater.fr/xmg/>.

called bricks, and correspond to the set of compiling steps of a meta-grammatical language. A brick can correspond to a description language (e.g. the TAG description language of the <syn>-dimension), to a subpart thereof (e.g. the feature structure language used by the <syn>-dimension), or to a solver (e.g. the tree solver used by the same dimension). Every part of the compiler is a brick, even the control language allowing one to express conjunction and disjunction, and a compiler is built by picking bricks and plugging them together. In this process, the parser for a new meta-grammatical language, and all the other compiling steps, can be automatically assembled according to the way the bricks are plugged together, to generate a whole compiler for this language. The idea is that every brick holds a fragment of compiler, dedicated to a fragment of language (described by a set of context-free rules). Whilst combining these language fragments to build a full language, the compiler fragments are also contributed to a full compiler.

Hence, when extending XMG by components that are able to handle typed feature structures, we can take advantage of the compiler's modularity to add another module, dedicated to this new task. Such a module, for example a new dimension, can be equipped with a dedicated specification language and compiler.

## 6.2 *Specification languages*

The design of the specification languages that we present in the following subsections is guided by certain goals, ideas and examples. Firstly, we try to adhere to the notation in the definitions in Section 4 as closely as possible. Secondly, we try to remain consistent with the coding style that already exists in XMG, though being largely unrestricted in principle from a technical point of view. Thirdly, and most importantly, we aim at specification languages that are inherently consistent, lightweight, transparent, and at the same time flexible. In doing so, we share certain elements from specification languages proposed in other work, but, as far as we know, our combination of these elements has not been presented elsewhere.

A complete code example based on the type hierarchy from Section 2 and the factorization from Section 3 is presented in the Appendix.

### 6.2.1 Specification of the signature

For the specification of the signature, that is to say attributes, types and base labels, the global fields `frame_types` and `frame_attributes` are available:

```
frame_types = {event, activity, motion, causation, ...}
frame_attributes = {actor, theme, goal, ...}
```

Base labels correspond to XMG variables and do not need to be declared globally.

### 6.2.2 Specification language for feature structure descriptions

Feature structure descriptions are specified within the `<frame>`-dimension of a class, which comes with a dedicated specification language and compiler. The mode of operation of the compiler is detailed below in Section 6.3.1. We make use of the following description language, which is basically a simple bracket notation:<sup>24</sup>

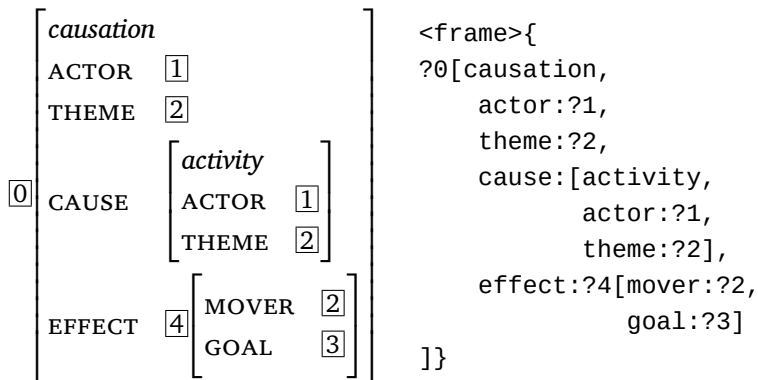
```
<frame>{Descriptions; Descriptions; ...}
Descriptions ::= var? '[' Description (',' Description)* ']'
Description ::= type | PathEquation | AVPair
PathEquation ::= attr+ '=' var? attr+ (':' Value)?
AVPair ::= attr+ ':' Value
Value ::= var | type | Descriptions
```

Unsurprisingly, `type` and `attr` stand for a type and an attribute from the signature, while `var` is an XMG variable. Borrowing the notation from Definition 7, `attr:type` is an attribute-value pair. Figure 11 then shows the description language in action, while mimicking the AVM representation of the frame. Note that the order of descriptions within a pair of brackets is generally unrestricted, as well as the number of type expressions (in order to fully support conjunctive types). The specification language furthermore follows the definitions in Section 4 in that it allows for the abbreviated specification of paths, namely as a sequence of attributes separated by whitespaces. Hence, `[cause:[actor:?1]]` and `[cause actor:?1]` have the same meaning. Path equations are similarly constructed using the equality sym-

---

<sup>24</sup>In the following we use a simplified Backus-Naur form to define the syntax of specification languages. Disjunction (`|`), optionality (`?`) and the Kleene operators (`*`, `+`) are encoded as usual.

Figure 11:  
Specification  
of the frame  
component of  
n0Vn1pp(dir)



bol =. For example, the meaning of [actor:?1,cause actor:?1] could also be expressed with [actor=cause actor:?1].<sup>25</sup>

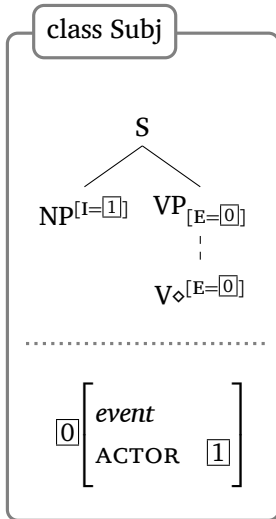
The use of XMG variables not only enables one to avoid path equations, and eventually to mimic AVM representations, as we just demonstrated, but they may also serve to link semantic components with positions in the syntactic tree, as can be seen from Figure 12, where the ACTOR role and the NP-slot of the subject are linked in this way. It shows that XMG variables may play the role of base labels in labelled feature structure descriptions (see Section 4).

Finally, it might be instructive to compare the specification language for feature structure descriptions proposed here with alternatives that are used elsewhere, particularly in grammar development tools for HPSG. The ALE/TRALE system (Götz *et al.* 1997; Carpenter *et al.* 2003), for example, is rather similar in this respect. The only major difference is found in the path specification, where attributes are separated with the colon, which happens to be also the attribute-value separator. This overloading might be disadvantageous – and in fact path equations then entail a different way of specifying paths.<sup>26</sup>

<sup>25</sup>Note that, in the <syn>-dimension, the symbol = has a different meaning for historical reasons; there it acts as a attribute-value separator.

<sup>26</sup>Another case of overloading can be observed in PATR-II (Shieber 1984), where the operator for path equations and attribute-value pairs coincides. As a consequence, paths are enclosed by angled brackets to distinguish them from regular (atomic) values. Nevertheless, the notation of feature structures in Shieber (1984) bears similarity to our specification language. Indeed, one could say that we basically merge, and extend, the path and feature structure representations from PATR-II. See also the PC-PATR manual (McConnel 1995).





```

class Subj
...
<syn>{
 node ?S [cat=s];
 node ?SUBJ [cat=np,
 top=[i=?1]];
 node ?VP [cat=vp, bot=[e=?0]];
 node ?V (mark=anchor)
 [cat=v, top=[e=?0]];
 ?S->?SUBJ; ?S->?VP; ?VP->?*?V;
 ?SUBJ>>?VP
}
<frame>{
 ?0[event,
 actor:?1]
}
...

```

Figure 12:  
Implementation  
of class Subj

In LKB (Copestake 2002), on the other hand, paths are encoded with dots instead of colons, while the whitespace acts as the separator in attribute-value pairs. This being merely an alphabetical variant of our proposal, there are other differences, notably the types being placed outside the feature structure brackets. Still, these differences seem rather marginal.

### 6.2.3 Specification language for feature structure constraints

We have seen in Section 4 that feature structure constraints can be – and in practice are – specified in different ways, namely either on the basis of a set of single constraint statements, or on the basis of a connected type hierarchy. Therefore one important aspect of the specification language for feature structure constraints is its versatility. Instead of dictating what direction to follow, the grammar writer should have several options at hand from which a suitable one may be chosen on a case-by-case basis.

Concerning the specification of feature structure constraints, two options are provided: a loose set of constraint statements, or a type hierarchy. The former are collected in the global field `frame_constraints`:

```

frame_constraints = {Constraint,Constraint,...}
Constraint ::=
 %% type constraint
 type+ '->' type+ |
 %% appropriateness condition
 type+ '->' Descriptions+ |
 %% feature-value constraint
 ('[' (AVPair|PathEquation) (',' AVPair|',' PathEquation)* '']'+
 '->' Descriptions+
 Descriptions ::= '[' Description (',' Description)* ']'
 Description ::= type | PathEquation | AVPair
 PathEquation ::= attr+ '=' attr+ (':' Value)?
 AVPair ::= attr+ ':' Value
 Value ::= type | Descriptions

```

The specification language for constraints largely integrates the specification language for (unlabelled) descriptions, while it adds  $\rightarrow$  as a symbol for generalized implication ( $\preceq$  in Definition 7). Similarly, a distinction is made between type constraints, appropriateness conditions and feature-value constraints, to which we confine ourselves in the following. An example is provided in Figure 13. Note that the antecedent and the consequent of  $\rightarrow$  may consist of more than one description separated by whitespaces, in which case the descriptions form a Cartesian product in the following way:  $t_1 t_2 \rightarrow t_3 t_4$  iff  $t_1 \rightarrow t_3$ ,  $t_1 \rightarrow t_4$ ,  $t_2 \rightarrow t_3$ ,  $t_2 \rightarrow t_4$ . Furthermore it is possible to reverse  $\rightarrow$ , hence to use  $t_2 \leftarrow t_1$  instead of  $t_1 \rightarrow t_2$ .

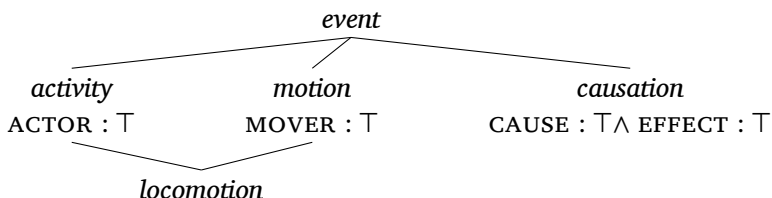
The other option is to specify feature structure constraints in the shape of a connected type hierarchy. For this the field `frame_type_hierarchy` is available and should be used in the following way:

```

frame_type_hierarchy = {Hierarchy,Hierarchy,...}
Hierarchy ::= '[' type (',' (Description|Hierarchy))* ']'
Descriptions ::= '[' Description (',' Description)* ']'
Description ::= type | PathEquation | AVPair
PathEquation ::= attr+ '=' attr+ (':' Value)?
AVPair ::= attr+ ':' Value
Value ::= type | Descriptions

```

An example of this way of specifying a type hierarchy was already included in Figure 13. Similarly to `frame_constraints`, only unlabelled feature structure descriptions are admitted, but unlike `frame_constraints`, the implication symbol  $\rightarrow$  is missing. Instead squared



```

frame_constraints = {
 activity -> event, activity -> [actor:],
 motion -> event, motion -> [mover:],
 causation -> event, causation -> [cause:+,effect:],
 locomotion -> activity motion}

frame_type_hierarchy = {
 [event, [activity, actor:+, [locomotion]],
 [motion, mover:+, [locomotion]],
 [causation, cause:+, effect:++]}

```

Figure 13: Examples of the specification of feature structure constraints. The type hierarchy is a fraction of the type hierarchy shown above in Figure 3 from Kallmeyer and Osswald (2013)

brackets receive a second interpretation where they correspond to types. In other words, the set of feature structure constraints enclosed by a pair of brackets constitute the type constraints and appropriateness conditions for exactly one type. If there are several type expressions, then they form a conjoined type. And if another pair of type-denoting brackets is embedded, then this is a subtype of the embedding type. This squared bracket notation is introduced in the Hierarchy part of the syntax definition of `frame_type_hierarchy`, and it is also the only one that is used in the example in Figure 13.

Note here that `frame_type_hierarchy` can only express a proper subset of feature structure constraints, since constraints with feature structures in their antecedents are excluded from this representational format in general. Fortunately, constraints like these may be specified in parallel in the `frame_constraints` field.

Finally, there is the possibility to set a flag to trigger either the maximal or the minimal model of feature structure constraints:

```
use hierarchy (maximal|minimal) with dims (frame)
```

At the time of writing, minimal models are compiled by default.

### 6.3 *Compilation*

Within XMG2, the compilation of feature structure descriptions and constraints leads to representations which are used in the Prolog component of the system. Therefore we will be mainly concerned with Prolog data structures in the following. Note that, other than with tree descriptions in the <syn>-dimension, no underspecification is involved, and therefore the compiler for feature structure descriptions and constraints employs no constraint solving in the proper sense.

#### 6.3.1 *Compilation of feature structure descriptions*

Typed feature structures are decomposed into two parts during compilation: types are compiled separately from feature structures. For the latter, we can reuse the XMG module (or brick) for untyped feature structures, which is already applied to feature structures in the <syn>-dimension. The module falls back on *attributed variables* and *association lists*, which are predefined data structures in Prolog. Attributed variables behave like common variables, the only difference being that they come with encapsulated attributes that can be accessed with dedicated commands only, and for which the standard unification algorithm can be customized. They are basically used here in order to replace the standard list (or term) unification by type unification and set union. Association lists, on the other hand, consist of key-value pairs with unique keys, thereby supporting the functionality of features within feature structures.

When a typed feature structure is declared, an attributed variable with two attributes is created: one attribute for the type and one for the feature-value pairs. Types are represented by bit vectors, basically Prolog lists over  $\{0, 1\}$ , that get unified by means of element-wise Boolean operations (see the next section). A set of feature-value pairs, on the other hand, is represented by an association list, whose values can be attributed variables again and thus induce recursion, i.e. feature structures of arbitrary depth. When two declared feature structures are unified, the compiler unifies their attributed variables in two different ways, namely with type unification on their type attribute and with set union on their association lists.

The check on the well-formedness of the generated structures is performed dynamically during their creation. This includes checking the appropriateness conditions (i.e. for invalid features or specific types of feature values) and computing type unifications. Contrary to the <syn>-dimension, the <frame>-dimension does not come with a description solver.

When the descriptions in the frame dimension are compiled, a number of instantiations and unifications of attributed variables is executed. It is important to note that the unification has to be explicitly specified by means of variable equations. Hence the compilation result may consist of several unconnected feature structures, as the compiler does not search for a minimal connected model that satisfies the processed feature structure descriptions. Furthermore note that, as unification is deterministic, there is at most one model for each accumulation.

### 6.3.2 Compilation of feature structure constraints

We are dealing with conjunctive types in the sense of Definition 8. Following a widespread approach (Aït-Kaci *et al.* 1989; Penn 1999; Kilbury *et al.* 2006; Skala *et al.* 2010), conjunctive types are internally represented by bit vectors (or “bit strings”), more precisely by Prolog lists over  $\{0, 1\}$ . The length of these lists is at least the number of elementary types in the signature. Every position in a bit vector stands for the membership of an elementary type in the conjunctive type, which means that a bit vector composed only of zeros is the most generic type (the empty set) and a bit vector composed only of ones is the conjunction of all elementary types.

We will present two methods to determine the set of valid bit vectors for a set of type constraints. The first one is a brute-force method that basically applies top-down filtering. The second one is based on subsumption matrices (Aït-Kaci *et al.* 1989; Penn 1999) and turns out to be more efficient. In the following let  $T = \{a, b, c, d\}$  be the set of elementary types and  $\#$  a bijective positioning function for bit vectors with  $\# = \{(a, 1), (b, 2), (c, 3), (d, 4)\}$ .

The top-down filtering on bit vector representations proceeds in the following way:

1. Generate the set of *virtual* (conjunctive) types, namely the bit vectors that represent the powerset of the set of elementary types:

$\{[1, 0, 0, 0], [1, 1, 0, 0], \dots\}$ .

2. Translate type constraints into bit vector patterns of *nonvalid* types:
 
$$\begin{aligned} a \leq b &\rightsquigarrow [1, 0, -, -] \\ a \wedge c \leq \perp &\rightsquigarrow [1, -, 1, -] \\ a \wedge b \leq d &\rightsquigarrow [1, 1, -, 0] \end{aligned}$$
3. Maximal model: Filter the set of bit vectors (representing the virtual types) based on the bit vector patterns in order to identify the valid types. Consequently, if no constraint is expressed, the set of valid types is the powerset of elementary types.

To compute the minimal model it needs an extra step:

- 3'. Minimal model: Translate elementary types and conjunctive types (on the left side of type constraints) into bit vector patterns of *declared* types:
 
$$\begin{aligned} a &\rightsquigarrow [1, -, -, -] \\ a \wedge b \leq d &\rightsquigarrow [1, 1, -, -] \end{aligned}$$

Then, for each declared type, determine the set of compatible bit vectors from the maximal model and select the bit vector with the fewest 1s. If there is more than one such bit vector, compute the bitwise AND over all these bit vectors and add the resulting bit vector to the set of declared types.

4. Assign bit vector patterns to elementary types:  $a \rightarrow [1, -, -, -]$ ,  $b \rightarrow [-, 1, -, -]$ , ...
5. Based on the set of valid bit vectors, unification of two types proceeds as list unification, after which the set of valid types is filtered with the resulting bit vector pattern. Finally the matching bit vector with the fewest 1s gets selected.

It is not hard to see that this method gets intractable for larger sets of elementary types, since the set of virtual types grows exponentially, namely with  $2^n$  where  $n$  is the number of elementary types.

Fortunately, methods based on subsumption matrices tend to be much more space-efficient, although they still come with at least quadratic growth in terms of the number of elementary types. In the following, we adapt the procedure of Ait-Kaci *et al.* (1989) (see also Penn 1999), and only add to it anonymous types and maximal models:

1. Generate a boolean matrix where rows and columns correspond to elementary types and anonymous types that are subject to type

constraints. An element  $a_{ij}$  in the  $i$ th row and the  $j$ th column has value 1 iff  $t_i \preceq t_j$  is a valid type constraint.<sup>27</sup> Otherwise it has value 0. For example, given type constraints  $d \preceq c$  and  $a \wedge b \preceq d$ , the following preliminary matrix is generated:

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ a \wedge b \end{array} \begin{array}{ccccc} a & b & c & d & a \wedge b \\ \left( \begin{array}{ccccc} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{array} \right) \end{array}$$

2. Multiply the matrix by itself until a fixpoint is reached.<sup>28</sup> This ensures that transitive subsumption relations are taken into account. When applied to the preliminary matrix above, we receive the following matrix, where  $c$  furthermore subsumes  $a \wedge b$ :

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ a \wedge b \end{array} \begin{array}{ccccc} a & b & c & d & a \wedge b \\ \left( \begin{array}{ccccc} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{array} \right) \end{array}$$

3. Each type is assigned two vectors: a subsumption vector taken from the row, and an ISA vector taken from the column (which was also used in the previous filtering approach). Hence the bit vector representation for type  $a$  consists of the subsumption vector  $[1, 0, 0, 0, 1]$  and the ISA vector  $[1, 0, 0, 0, 0]$ . The difference of the minimal and maximal model emerges in the use of these vector pairs.
4. Minimal model: During unification the subsumption vectors are combined with bitwise AND in order to determine the most general common subtype (Ait-Kaci *et al.* 1989). To give an example, the unification of the subsumption vectors assigned to  $a$  and  $d$  results in the subsumption vector of  $a \wedge b$ :

---

<sup>27</sup>We tacitly assume that tautologies (e.g.  $t_i \preceq t_i$  and  $t_i \wedge t_j \preceq t_i$ ) are taken into account.

<sup>28</sup>According to Ait-Kaci *et al.* (1989), this takes at most  $\log_2 n$  iterations.

$$\begin{array}{r} a \quad [1, 0, 0, 0, 1] \\ d \quad [0, 0, 0, 1, 1] \\ \hline a \wedge b \quad [0, 0, 0, 0, 1] \end{array}$$

Otherwise, if the resulting bit vector was not found among the bit vectors of the matrix, the unification does not necessarily fail. It only fails if the resulting bit vector only consists of 0s.

- 4'. Maximal model: During unification, the ISA vectors are combined with bitwise OR, while the subsumption vectors are ignored. Hence the unification of  $a$  and  $d$  yields an ISA vector not found in the matrix:

$$\begin{array}{r} a \quad [1, 0, 0, 0, 0] \\ d \quad [0, 0, 1, 1, 0] \\ \hline (a \wedge d) \quad [1, 0, 1, 1, 0] \end{array}$$

In order to account for constraints on anonymous types, further unification steps can be necessary. For example, upon unifying the ISA vectors of  $a$  and  $b$ , the ISA vector of  $a \wedge b$  has to be added as well:

$$\begin{array}{r} a \quad [1, 0, 0, 0, 0] \\ b \quad [0, 1, 0, 0, 0] \\ \hline [1, 1, 0, 0, 0] \\ a \wedge b \quad [1, 1, 1, 1, 1] \\ \hline [1, 1, 1, 1, 1] \end{array}$$

Types that are explicitly ruled out in the type constraints are accounted for by means of extra filtering. For example, a type constraint such as  $a \wedge d \leq \perp$  would give rise to a filter based on the pattern  $[1, -, -, 1, -]$ , which would be applied to the resulting ISA vector after unification.

For each one of the computed valid types, i.e. the assigned ISA vectors, XMG also computes the set of appropriateness conditions. Once again, this step uses the technique of vector pattern matching: when a valid type matches a pattern corresponding to a appropriateness condition, the appropriateness condition is added to the list of appropriateness conditions for this type. These lists can be used in other precompilation steps to check for cyclicity in the feature structure constraints, and for their incompatibility.



At the end of the compilation process, the bit vectors can be easily mapped back to conjunctive types and the corresponding elementary type, if it exists.

## 7 DISCUSSION AND CONCLUSION

In this article, we presented recent efforts to extend the grammar engineering framework XMG in order to deal with frame representations in the format of typed feature structures. Because metagrammatical factorization involves the composition of feature structures and types along given feature structure constraints, the full power of unification on typed feature structures is needed in XMG. We showed that the simulation of typed feature structures within the <sem>-dimension comes with severe disadvantages concerning the implementation of types and type unification. Therefore a new toolkit was developed, including a novel <frame>-dimension, which is adjusted to the peculiarities of typed feature structures and type unification, and which should eventually reduce the burden for the grammar writer. The article explained the main components of this toolkit: the specification language, for which a comprehensive code example is included in the appendix, and the compilation procedure, which uses bit vector encodings of types.

While the focus was on the theoretic foundation of the proposed extension and its proof of concept, aspects of computational complexity and the possibilities for optimization (e.g. regarding the size of bit vectors) were largely set aside. Undoubtedly, there is room for future improvements of the compiler. It has to be stressed, however, that compilation with XMG is not as time-critical as parsing, because XMG compilation is part of the preprocessing (Kallmeyer and Osswald 2013, 56). Nevertheless it would be interesting to see how compilation time scales as a function of the size and structure of the type hierarchy, and whether this is relevant given theoretically justified conceptual type systems. As no complex examples of the latter kind are known to us, this remains to be seen.<sup>29</sup> We are aware, of course, that the issue of complexity will become more critical once these resources are used for parsing. Note that, while the presented extensions to XMG are fully

---

<sup>29</sup> Just for comparison, Skala and Penn (2011) count some 3412 elementary types in the English Resource Grammar (ERG, Copestake and Flickinger 2000).

operational in a recent prototype, a compatible lexical component as well as a parser have yet to be implemented.

Regardless of complexity issues, it could be useful to separate the sources and compilation procedures for global feature structure constraints on the one hand, and local feature structure descriptions within the <frame>-dimension on the other hand. The reason is that at some point the feature structure constraints should stabilize, given that they represent cognitive concepts, and therefore constant recompilation triggered by changes in other parts of the metagrammar would be superfluous. At the current stage of research, however, as the development of frame representations is still ongoing, we think that metagrammars are the right place to implement and to experiment with feature structure constraints.

Finally it remains to be stressed that the combination of the <frame>-dimension with the <syn>-dimension is by no means privileged. The <frame>-dimension can also be used to implement standalone frames, or to implement recent frame-based accounts to morphological decomposition (e.g. Zinova and Kallmeyer 2012), thereby considerably widening the scope of XMG.

### APPENDIX: COMPLETE CODE EXAMPLE

The following code example covers the implementation of the type hierarchy in Figure 3 and the factorization of the prepositional object construction in Figures 4 and 5.

```

%%
% HEADER:
%%
type MARK = {subst, nadj, foot, anchor, coanchor, flex, lex}
type CAT = {np, v, vp, s, pp}
type VAR !
property mark : MARK
feature cat : CAT
feature top : VAR
feature bot : VAR
feature i : VAR

```

*Implementing semantic frames with XMG*

```
feature e : VAR
feature path : VAR

frame_types = {event,activity,motion,causation,translocation,
 onset-causation,extended-causation,locomotion,bounded-
 translocation,bounded-locomotion}
frame_attributes = {actor,theme,goal,mover,path,cause,effect}
frame_constraints = { activity -> event,
 activity -> actor:+,
 motion -> event,
 motion -> mover:+,
 causation -> event,
 causation -> cause:+,
 causation -> effect:+,
 [activity,motion] -> actor=mover,
 translocation -> motion,
 translocation -> path:+,
 bounded-translocation -> translocation,
 bounded-translocation -> goal:+,
 locomotion -> activity translocation,
 bounded-locomotion -> locomotion bounded-translocation
}

%%
%% TREE FRAGMENTS:
%%
class VSpine
export ?V
declare ?VP ?V ?X0
{<syn>{
 node ?VP [cat=vp] {
 node ?V (mark=anchor)[cat=v]
 } } }
%%
class Subj
export ?V ?X0
declare ?S ?NP ?VP ?V ?X0 ?X1
{ <syn>{
 node ?S [cat = s];
 node ?NP (mark=subst)[cat=np,top=[i=?X1]];
 node ?VP [cat=vp,bot=[e=?X0]];
 node ?V (mark=anchor)[cat=v,top=[e=?X0]];
 ?S -> ?NP; ?S -> ?VP; ?VP ->* ?V; ?NP >> ?VP
};
};
```

```
<frame> {
 ?X0[event,
 actor:?X1]
} }
%%
class DirObj
export ?V ?X0
declare ?VP ?NP ?V ?X0 ?X2
{ <syn>{
 node ?VP [cat = vp,bot=[e=?X0]]{
 node ?V [cat=v,top=[e=?X0]]
 ,, ,node ?NP (mark=subst)[cat=np,top=[i=?X2]]
 } };
 <frame>{
 ?X0[event,goal:?X2]
 |
 ?X0[event,theme:?X2]
 } }
%%
class DirPrepObj
export ?V ?X0
declare ?VP1 ?VP2 ?PP ?V ?X0 ?X1 ?X2 ?X3
{ <syn>{
 node ?VP1 [cat = vp,bot=[path=?X3]]{
 node ?VP2 [cat = vp,bot=[path=?X3]]{
 node ?V (mark=anchor)[cat=v]}
 node ?PP (mark=subst)[cat=pp,top=[i=?X1,e=?X0]]
 } };
 <frame>{
 ?X0[bounded-translocation,
 goal:?X1,
 path:?X3]
 } }
%%
% TREE TEMPLATES:
%%
class n0V
export ?V ?X0
declare ?V ?SSubj ?Spine ?X0
{
 ?SSubj = Subj[];
 ?Spine = VSpine[];
 ?SSubj.?V = ?V;
}
```

*Implementing semantic frames with XMG*

```
?Spine.?V = ?V;
?SSubj.?X0 =?X0
}
%%
class n0Vn1
import n0V[]
declare ?Obj
{
 ?Obj = DirObj[];
 ?Obj.?V = ?V
}
%%
class n0Vn1pp-dir
import n0Vn1[]
declare ?PPObj ?X1 ?X2 ?X3 ?X4
{
 ?PPObj = DirPrepObj[];
 ?PPObj.?V = ?V;
 ?PPObj.?X0 = ?X4;
 <frame>{
 ?X0[causation,
 actor:?X1,
 theme:?X2,
 cause:[activity,
 actor:?X1,
 theme:?X2],
 effect:?X4[
 mover:?X2,
 goal:?X3]
]
} }
%%
% EVALUATION:
%%
value n0V
value n0Vn1
value n0Vn1pp-dir
```

## REFERENCES

- Anne ABEILLÉ and Owen RAMBOW (2000a), Tree Adjoining Grammar: An overview, in Abeillé and Rambow (2000b), pp. 1–68.
- Anne ABEILLÉ and Owen RAMBOW, editors (2000b), *Tree Adjoining Grammars: Formalisms, linguistic analyses and processing*, number 107 in CSLI Lecture Notes, CSLI Publications, Stanford, CA.
- Hassan AÏT-KACI, Robert BOYER, Patrick LINCOLN, and Roger NASR (1989), Efficient implementation of lattice operations, *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 11(1):115–146.
- Lawrence BARSALOU (1992), Frames, concepts, and conceptual fields, in Adrienne LEHRER and Eva Feder KITTEY, editors, *Frames, fields, and contrasts: New essays in semantic and lexical organization*, pp. 21–74, Lawrence Erlbaum Associates, Hillsdale, NJ.
- Tilman BECKER (1994), *HyTAG: A new type of Tree Adjoining Grammars for hybrid syntactic representations of free word order languages*, Ph.D. thesis, Universität des Saarlandes,  
<http://www.dfki.de/~becker/becker.diss.ps.gz>.
- Tilman BECKER (2000), Patterns in metarules for TAG, in Abeillé and Rambow (2000b), pp. 331–342.
- Johan BOS (1996), Predicate logic unplugged, in Paul DEKKER and Martin STOKHOF, editors, *Proceedings of the tenth Amsterdam Colloquium*, pp. 133–143, Amsterdam, Netherlands.
- Marie-Hélène CANDITO (1996), A principle-based hierarchical representation of LTAGs, in *Proceedings of the 16th International Conference on Computational Linguistics (COLING 96)*, Copenhagen, Denmark,  
<http://aclweb.org/anthology-new/C/C96/C96-1034.pdf>.
- Bob CARPENTER (1992), *The logic of typed feature structures with applications to unification grammars, logic programs and constraint resolution*, number 32 in Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, UK.
- Bob CARPENTER, Gerald PENN, and Mohammad HAJI-ABDOLHOSSEINI (2003), *The Attribute Logic Engine user's guide with TRALE extensions*, version 4.0 beta edition.
- Bob CARPENTER and Carl POLLARD (1991), Inclusion, disjointness and choice: The logic of linguistic classification, in *Proceedings of the 29th annual meeting of the Association for Computational Linguistics*, pp. 9–16, Berkeley, CA,  
<http://acl.ldc.upenn.edu/P/P91/P91-1002.pdf>.
- Ann COPESTAKE (2002), *Implementing typed feature structure grammars*, CSLI Publications, Stanford, CA.

### *Implementing semantic frames with XMG*

Ann A. COPESTAKE and Dan FLICKINGER (2000), An open source grammar development environment and broad-coverage English grammar using HPSG, in *Proceedings of the second Conference on Language Resources and Evaluation (LREC 2000)*, Athens, Greece.

Benoit CRABBÉ and Denys DUCHIER (2005), Metagrammar redux, in Henning CHRISTIANSEN, Peter Rossen SKADHAUGE, and Jørgen VILLADSEN, editors, *Constraint solving and language processing*, number 3438 in Lecture Notes in Computer Science, pp. 32–47, Springer, Berlin, Germany.

Benoit CRABBÉ, Denys DUCHIER, Claire GARDENT, Joseph LE ROUX, and Yannick PARMENTIER (2013), XMG: eXtensible MetaGrammar, *Computational Linguistics*, 39(3):1–66,  
<http://hal.archives-ouvertes.fr/hal-00768224/en/>.

Denys DUCHIER, Brunelle Magnana EKOUKOU, Yannick PARMENTIER, Simon PETITJEAN, and Emmanuel SCHANG (2012), Describing morphologically rich languages using metagrammars: A look at verbs in Ikota, in *Workshop on Language Technology for Normalisation of Less-Resourced Languages (SALTMIL 8 - AfLaT 2012)*, pp. 55–59, Istanbul, Turkey, <http://www.tshwanedje.com/publications/SALTMiL8-AfLaT2012.pdf#page=67>.

Charles J. FILLMORE (1982), Frame Semantics, in The Linguistic Society of Korea, editor, *Linguistics in the morning calm*, pp. 111–137, Hanshin Publishing, Seoul, South Korea.

Charles J. FILLMORE (2007), Valency issues in FrameNet, in Thomas HERBST and Katrin GÖTZ-VOTTELER, editors, *Valency: Theoretical, descriptive and cognitive issues*, pp. 129–162, Mouton de Gruyter, Berlin, Germany.

Dan FLICKINGER (2000), On building a more efficient grammar by exploiting types, *Natural Language Engineering*, 6(1):15–28.

Anette FRANK and Josef VAN GENABITH (2001), GlueTag. Linear Logic based Semantics for LTAG – and what it teaches us about LFG and LTAG, in Miriam BUTT and Tracy Holloway KING, editors, *Proceedings of the LFG01 conference*, CSLI Publications, Hong Kong.

Claire GARDENT and Laura KALLMEYER (2003), Semantic construction in Feature-Based TAG, in *Proceedings of the 10th meeting of the European Chapter of the Association for Computational Linguistics*, pp. 123–130.

Adele GOLDBERG (2006), *Constructions at work. The nature of generalizations in language*, Oxford University Press, Oxford, UK.

Thilo GÖTZ, Detmar MEURERS, and Dale GERDEMANN (1997), *The ConTroll manual*, Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen, Germany,  
<http://www.sfs.uni-tuebingen.de/control1/control1-manual.ps.gz>,  
draft of 17. September 1997 for ConTroll v.1.0b and XTroll v.5.0b.

Aravind K. JOSHI and Yves SCHABES (1997), Tree-Adjoining Grammars, in Grzegorz ROZENBERG and Arto SALOMAA, editors, *Handbook of Formal Languages*, volume 3, pp. 69–124, Springer, Berlin, Germany.

Aravind K. JOSHI, K.Vijay SHANKER, and David WEIR (1990), The convergence of mildly context-sensitive grammar formalisms, Technical Report MS-CIS-90-01, Department of Computer and Information Science, University of Pennsylvania, [http://repository.upenn.edu/cis\\_reports/539/](http://repository.upenn.edu/cis_reports/539/).

Laura KALLMEYER, Timm LICHTTE, Wolfgang MAIER, Yannick PARMENTIER, Johannes DELLERT, and Kilian EVANG (2008), TuLiPA: Towards a multi-formalism parsing environment for grammar engineering, in *Proceedings of the workshop on Grammar Engineering Across Frameworks (GEAF '08)*, pp. 1–8, Manchester, UK.

Laura KALLMEYER and Rainer OSSWALD (2012a), An analysis of directed motion expressions with Lexicalized Tree Adjoining Grammars and frame semantics, in Luke ONG and Ruy DE QUEIROZ, editors, *Proceedings of WoLLIC*, number 7456 in Lecture Notes in Computer Science (LNCS), pp. 34–55, Springer, Berlin, Germany.

Laura KALLMEYER and Rainer OSSWALD (2012b), A frame-based semantics of the dative alternation in Lexicalized Tree Adjoining Grammars, in Christopher PIÑÓN, editor, *Empirical Issues in Syntax and Semantics 9*, pp. 167–184, Paris, France.

Laura KALLMEYER and Rainer OSSWALD (2013), Syntax-driven semantic frame composition in Lexicalized Tree Adjoining Grammar, *Journal of Language Modelling*, 1:267–330.

Paul KAY (2002), An informal sketch of a formal architecture for Construction Grammar, *Grammars*, 5:1–19.

James KILBURY, Wiebke PETERSEN, and Christof RUMPF (2006), Inheritance-based models of the lexicon, in Dieter WUNDERLICH, editor, *Advances in the theory of the lexicon*, number 13 in Interface Explorations, pp. 429–478, De Gruyter, Berlin, Germany.

Timm LICHTTE, Alexander DIEZ, and Simon PETITJEAN (2013), Coupling trees, words and frames through XMG, in *Proceedings of the ESSLLI 2013 workshop on high-level methodologies for grammar engineering*, Düsseldorf, Germany.

Robert MALOUF, John CARROLL, and Ann COPESTAKE. (2000), Efficient feature structure operations without compilation, *Natural Language Engineering*, 6(1):29–46.

Stephen MCCONNELL (1995), *PC-PATR reference manual*, <http://www.sil.org/pcpatr/manual/pcpatr.html>, version 0.97a9.

Rainer OSSWALD (2003), *A logic of classification with applications to linguistic theory*, Dissertation, FernUniversität Hagen.



Rainer OSSWALD and Robert D. VAN VALIN, Jr. (2014), FrameNet, frame structure, and the syntax-semantics interface, in Thomas GAMERSCHLAG, Doris GERLAND, Rainer OSSWALD, and Wiebke PETERSEN, editors, *Frames and concept types*, number 94 in Studies in Linguistics and Philosophy, pp. 125–156, Springer, Cham, Switzerland.

Gerald PENN (1999), An optimized Prolog encoding of typed feature structures, Arbeitspapiere des SFB 340 138, University of Tübingen.

Guy PERRIER (2000), Interaction Grammars, in *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pp. 600–606, Saarbrücken, Germany.

Wiebke PETERSEN (2007), Representation of concepts as frames, *The Baltic International Yearbook of Cognition, Logic and Communication*, 2:151–170.

Wiebke PETERSEN and Tanja OSSWALD (2014), Concept composition in frames: Focusing on genitive constructions, in Thomas GAMERSCHLAG, Doris GERLAND, Rainer OSSWALD, and Wiebke PETERSEN, editors, *Frames and concept types*, number 94 in Studies in Linguistics and Philosophy, pp. 243–266, Springer, Cham, Switzerland.

Carlos A. PROLO (2002), Generating the XTAG English grammar using metarules, in *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pp. 814–820, Taipei, Taiwan.

Stuart M. SHIEBER (1984), The design of a computer language for linguistic information, in *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting of the Association for Computational Linguistics*, pp. 362–366, Stanford, CA, <http://www.aclweb.org/anthology/P84-1075>.

Stuart M. SHIEBER and Yves SCHABES (1990), Synchronous Tree-Adjoining Grammars, in *Proceedings of the 13th International Conference on Computational Linguistics (COLING 1990)*, pp. 253–258, Helsinki, Finland.

Matthew SKALA, Victoria KRAKOVNA, János KRAMÁR, and Gerald PENN (2010), A generalized-zero-preserving method for compact encoding of concept lattices, in *Proceedings of the 48th annual meeting of the Association for Computational Linguistics*, pp. 1512–1521, Uppsala, Sweden.

Matthew SKALA and Gerald PENN (2011), Approximate bit vectors for fast unification, in Makoto KANAZAWA, András KORNAI, Marcus KRACHT, and Hiroyuki SEKI, editors, *The mathematics of language*, number 6878 in Lecture Notes in Computer Science, pp. 158–173, Springer, Berlin, Germany.

Matthew STONE and Christine DORAN (1997), Sentence planning as description using Tree Adjoining Grammar, in *Proceedings of the eighth conference of the European Chapter of the Association for Computational Linguistics (EACL'97)*, pp. 198–205.

*Timm Lichte, Simon Petitjean*

Fei XIA, Martha PALMER, and K. VIJAY-SHANKER (2010), Developing Tree-Adjoining Grammars with lexical descriptions, in Srinivas BANGALORE and Aravind K. JOSHI, editors, *Using complex lexical descriptions in natural language processing*, pp. 73–110, MIT Press, Cambridge, UK.

Yulia ZINOVA and Laura KALLMEYER (2012), A frame-based semantics of locative alternation in LTAG, in *Proceedings of the 11th international workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pp. 28–36, Paris, France, <http://www.aclweb.org/anthology-new/W/W12/W12-4604>.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*

<http://creativecommons.org/licenses/by/3.0/>



# A type-logical treebank for French

*Richard Moot*

CNRS (LaBRD), Bordeaux University

## ABSTRACT

This paper describes the TLGbank, a treebank developed in the framework of (multimodal) type-logical grammar. Using the French Treebank as a starting point, a combination of automated and manual techniques are applied to obtain type-logical derivations (parses) corresponding to the phrases of the French Treebank. The TLGbank has been developed with applications to wide-coverage semantics in mind. This means that the TLGbank has richer structure than the original French Treebank, especially where it concerns semantically relevant information such as passives, coordination, extraction and gapping.

*Keywords:*  
*type-logical  
grammar,  
categorial  
grammar,  
semi-automatic  
grammar  
extraction*

1

## INTRODUCTION

Categorial grammars have interesting theoretical advantages, most notably their very clean syntax-semantics interface. In the last decade, research in Combinatory Categorial Grammar has shown that this is not merely a *theoretical* advantage, but that, with the appropriate resources and tools – an annotated treebank, the CCGbank (Hockenmaier and Steedman 2007), a very efficient parser (Clark and Curran 2004) and a semantic lexicon (Bos *et al.* 2004) – we can use categorial grammars for wide-coverage, deep semantic analysis. Applications of the resulting wide-coverage semantics include natural-language question-answering (Bos *et al.* 2007) and recognising textual entailments (Bos and Markert 2005).

The development of the CCGbank, which has allowed parameter optimization for the wide-coverage parser and provided a framework

(in types and in derivations) for the semantic applications, has been a key element for these applications.

Categorial grammars in the logical tradition initiated by Lambek (1958) (Moortgat 2011; Morrill 2011; Moot and Retoré 2012) have stayed somewhat behind in terms of their application to large-scale linguistic data. The goal of the current paper is to describe the TLGbank, a semi-automatically extracted treebank containing type-logical proofs, created with the explicit goal of making similar wide-coverage parsing and semantics possible in the type-logical context.

The work described in this paper extends and refines a much earlier version of the TLGbank (Moot 2010b). Lefeuvre *et al.* (2012) and Moot (2012) discuss some initial applications of the treebank to wide-coverage semantics.

## 2

## TYPE-LOGICAL GRAMMAR

This section is a very short introduction to (multimodal) type-logical grammars. For more detailed introductions, see Oehrle (2011), Moortgat (2011, Section 2.4) or Moot and Retoré (2012, Chapter 5).

Although the treebank is annotated using multimodal type-logical grammar, the annotation has been chosen in such a way that derivations in the treebank can easily be translated into derivations of the Displacement calculus (Morrill *et al.* 2011) or of first-order linear logic (Moot and Piazza 2001; Moot 2014). Translations to other versions of categorial grammar are conceivable, but will probably require significantly more work.

The atomic formulas are  $n$  (for nouns),  $np$  (for noun phrases),  $pp_x$  (for prepositional phrases, with  $x$  the preposition heading the phrase) and  $s_x$  for sentences, where we distinguish between several types of sentences/phrases:  $s_{main}$  for main, tensed sentence,  $s_{whq}$  for a wh-question,  $s_q$  for a sentence introduced by “que” (*that*) and further types for passives  $s_{pass}$ , infinitives  $s_{inf}$ ,<sup>1</sup> and past  $s_{ppart}$  and present  $s_{ppres}$  participles; this is inspired by the French Treebank annotation –

---

<sup>1</sup>Like prepositions,  $s_{inf}$  is further subdivided into categories for infinitive phrases headed by a preposition:  $s_{inf_a}$ ,  $s_{inf_{de}}$ ,  $s_{inf_{pour}}$ ,  $s_{inf_{par}}$ . This allows us to distinguish, for example, between “finir de” (*to finish doing something*) and “finir par” (*to end up doing something*). The infinitive headed by “pour” occurs in constructions like “trop tôt pour ...” (*too early to ...*).

$\frac{}{w \vdash A} \textit{Lex}$	$\frac{}{x \vdash A} \textit{Hyp}$
$\frac{X \vdash A/B \quad Y \vdash B}{X \circ Y \vdash A} /E$	$\frac{X \vdash B \quad Y \vdash B \setminus A}{X \circ Y \vdash A} \setminus E$
$\frac{x \vdash B \quad \vdots \quad X \circ x \vdash A}{X \vdash A/B} /I$	$\frac{x \vdash B \quad \vdots \quad x \circ X \vdash A}{X \vdash B \setminus A} \setminus I$
$\frac{X[Y] \vdash B \quad Z \vdash B \setminus_1 A}{X[Y \circ_1 Z] \vdash A} \setminus_1 E$	$\frac{x \vdash B \quad \vdots \quad X[Y \circ x] \vdash A}{X[Y] \vdash A / \diamond_1 \square_1 B} / \diamond_1 \square_1 I$

Table 1:  
Logical rules for  
multimodal categorial  
grammars

though passives are not annotated as such in this treebank – and the categorial treatments of Carpenter (1991) and Hockenmaier and Steedman (2007). The different subtypes of *s* and *pp* are implemented using first-order variables and unification, following Moot (2014) and Morrill (1994, Section 2.1).

An intransitive verb is assigned  $np \setminus s_{main}$ , indicating that it requires a noun phrase to its left in order to form an inflected sentence. Similarly, transitive verbs are assigned the formula  $(np \setminus s_{main}) / np$ , requiring a noun phrase to their right in order to form an intransitive verb. In what follows, we will often simply write *s* instead of  $s_{main}$ .

To make this article understandable to the reader not intimately familiar with modern type-logical grammars, all examples in the text use the simplified presentation of Table 1. The intrepid reader interested in the full technical details can find the complete presentation in Appendix A, with further applications in Appendix B.

We will abbreviate the lexicon rule as  $\frac{w}{A}$ . The rule for  $/E$  simply states that whenever we have shown an expression *X* to be of type  $A/B$  and we have shown an expression *Y* to be of type *B*, then the tree with *X* as its immediate subtree on the left and *Y* as its immediate subtree of the right is of type *A* (the  $\setminus E$  rule is symmetric).

An easy instantiation of the  $/E$  rule (with  $X := the$ ,  $Y := student$ ,  $A := np$ ,  $B := n$ ) would be the following.

$$\frac{the \vdash np/n \quad student \vdash n}{the \circ student \vdash np} /E$$

The two rules at the bottom row of the table require some special attention. The  $\backslash_1 E$  rule is an *infixation rule*. This rule is used for adverbs (and other VP modifiers) occurring after the verb. Like the  $\backslash E$  rule, it takes a  $B$  formula as its argument, but infixes itself to the right of any subtree  $Y$  of  $X$  ( $X[Y]$  denotes a tree  $X$  with a designated subtree  $Y$ ). This tree  $Y$  can occur at any depth in the tree  $X[Y]$ , including the root, i.e.  $Y$  can be equal to  $X[Y]$ .<sup>2</sup> An example is shown below for the VP “*impoverishes the CGT dangerously*”. The interest of this rule is that it allows a uniform type assignment for adverbs occurring post-verbally, regardless of other verb arguments.

$$\frac{appauvrit \vdash (np \backslash s) / np \quad la \circ CGT \vdash np}{appauvrit \circ (la \circ CGT) \vdash np \backslash s} /E \quad \frac{dangereusement \vdash (np \backslash s) \backslash_1 (np \backslash s)}{(appauvrit \circ_1 dangereusement) \circ (la \circ CGT) \vdash np \backslash s} /E$$

Each occurrence of the introduction rules  $/I$ ,  $\backslash I$  and  $/\diamond_1 \square_1$  uses a distinct syntactic variable  $x$  which is unique to the proof; therefore, in the case of a proof containing multiple introduction rules, the hypothesis corresponding to an introduction rule can always be uniquely determined by this variable name (we can use any naming convention to ensure this; common choices are  $x_0, x_1, \dots$  or, for shorter proofs,  $x, y, z$ ).

The  $/\diamond_1 \square_1$  rule is an *extraction rule*, extracting a  $B$  constituent from any right branch inside an  $X$  constituent.<sup>3</sup> Comparing the rule  $/\diamond_1 \square_1 I$  to the rule  $/I$ , we can see that  $/I$  is the special case of  $/\diamond_1 \square_1 I$  where the context  $X[\ ]$  is empty (i.e. where  $X[Y]$  is equal to  $Y$ ). From the point of view of semantics the two rules are the same — both correspond to abstraction over the semantic variable assigned to the  $B$  formula which is withdrawn by the rule — but the rule  $/\diamond_1 \square_1 I$  can

<sup>2</sup>For adverbs, as here,  $Y$  is typically the verb, but in principle infixation is possible anywhere (an admitted oversimplification, which can be remedied by a more sophisticated treatment of mode information).

<sup>3</sup>For readers familiar with the Displacement calculus (Morrill *et al.* 2011), the infixation construction  $A \backslash_1 B$  corresponds to  $B \downarrow A$  and the extraction construction  $A / \diamond_1 \square_1 B$  to  $\wedge(A \uparrow B)$ .



detailed treatment of the unabbreviated version of this proof, showing notably how to *derive*  $A$  from  $\diamond_1 \square_1 A$ . To summarize, formulas of the form  $\diamond_1 \square_1 A$  are special types of  $A$  formulas that can be extracted from deeply embedded positions.

### 3 THE FRENCH TREEBANK

The French Treebank (FTB, Abeillé *et al.* 2000) is a set of syntactically annotated news articles from the newspaper *Le Monde*. The FTB consists of 12,891 annotated sentences with a total of 383,227 words. The FTB has previously been used to extract phrase structure grammars (Arun and Keller 2005), dependency grammars (Candido *et al.* 2009; Guillaume and Perrier 2012), lexical-functional grammars (Schluter and van Genabith 2008), and tree adjoining grammars (Dybro-Johansen 2004).

For its annotation, the FTB uses simple, rather flat trees with some functional syntactic annotation (subject, object, infinitival argument, etc.). Consecutive multiword-expressions have been merged in the annotation and neither traces nor discontinuous dependencies have been annotated.

Consider the following sentence from the French Treebank.

- (2) À cette époque, on avait dénombré cent quarante candidats  
at that time, we had counted hundred-forty candidates  
'At that time, there were 140 candidates.'

Its FTB annotation is shown in Figure 1. We can see that verb clusters are treated as constituents (labelled VN) and that the arguments of the verb occur as sisters of this verbal cluster. For example, the object noun phrase in Figure 1 is the sister of the VN. However, as we will see in Section 4.3, we obtain a much neater analysis when we treat the object as an argument of “dénombré” (*counted*), which is the past participle of a transitive verb.

### 4 GRAMMAR EXTRACTION

Grammar extraction algorithms for categorial grammars follow a general methodology – see, for example, Buszkowski and Penn (1990),



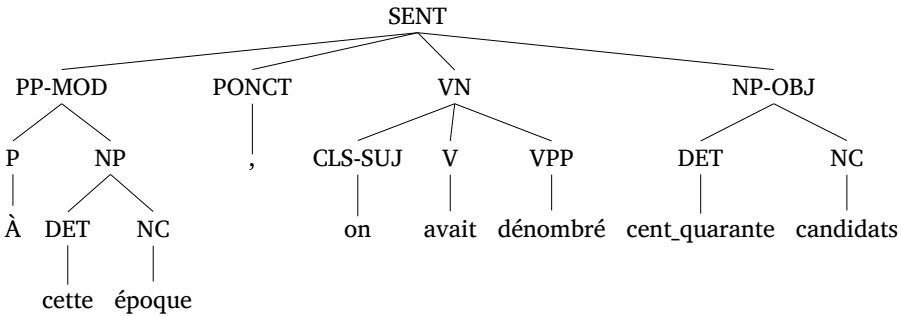


Figure 1: An example sentence from the French Treebank

Moortgat and Moot (2001), Hockenmaier and Steedman (2007) and Sandillon-Rezer (2013), shown as item 2 below – with some additional rules to deal with the quirks of the format of the input treebank. A high-level description of the grammar extraction algorithm used to convert the FTB into the TLGbank is given below.

1. split multiword expressions,
2. binarize the tree, keeping track of the distinction between modifiers and arguments; arguments are assigned formulas based on their syntactic label (e.g. *np* for a noun phrase argument, *np*\s<sub>inf</sub> for an infinitival argument, etc.)
3. reattach verb cluster arguments,
4. rearrange coordinations,
5. insert traces in the appropriate places and assign the appropriate formulas to relative pronouns and clitics.<sup>5</sup>

All steps are done by a single Prolog tree transformation, then verified and corrected manually (either by writing an ad hoc tree transformation script or by manually editing the output, then verifying that the result remains a valid derivation). Since the FTB annotation makes

---

<sup>5</sup>Subject clitics are treated as normal *np* subjects. Object clitics, such as the object clitic “l” in “Marie l’aime” (Marie him-clitic loves, *Marie loves him*) are assigned the formula  $(np \setminus s) / ((np \setminus s) / \diamond_1 \square_1 np)$  following the analysis of Moot and Retoré (2006). By assigning these higher-order formulas to the clitics, we can assign a normal transitive verb formula to “aime” (*loves*). Only the reflexive clitic “se” and the clitic “y” in the construction “il y a” (*there is/are*) are treated as arguments of the verb (with formulas *cl<sub>se</sub>* and *cl<sub>y</sub>*, respectively).

the distinction between modifiers and arguments only for certain categories (sentences, infinitive phrases, present participle phrases, but not past participle phrases or noun phrases), this information is not explicitly annotated for many major categories (the extraction script treats these cases as modifiers for noun phrases and as arguments for other categories, such as past participle phrases). In addition, all forms of the verb “être” (*to be*) with a past participle as argument have been manually changed to passive whenever this was a passive construction.<sup>6</sup>

In Step 4, which harmonizes the annotation of coordinations, many simple coordinations are treated correctly by the extraction script. Special care has been taken of the punctuation symbols, which in many cases are manually given a coordination-like formula assignment, and of gapping, which must be treated manually as well (the treatment of gapping is presented in detail in Appendix B.4).

Finally, relative pronouns are treated by the extraction script as arguments of the immediately following verb, which is correct in many cases but needs to be manually verified for all occurrences.

In sum, after a pass of the extraction script, many constructions are manually verified and corrected. To give an indication of the amount of manual cleanup done: simply running the Prolog script on the treebank results in a lexicon with 5,240 distinct formulas assigned to the words of the lexicon (Moot 2010b) (note that this is without a distinction between passives and past participles), but after cleanup there are 1,101.

From Section 4.1 to Section 4.5, we will treat each of the stages of the extraction algorithm in turn.

#### 4.1 *Splitting multiword expressions*

The French Treebank treats many multiword expressions as single nodes in the annotation. For example, the expression “dépôt de bilan” (*voluntary liquidation*) occurs as “dépôt\_de\_bilan”; similarly, as shown in Figure 1, numbers such as “cent quarante” (140) are analysed as

---

<sup>6</sup> Not all occurrences of passives are accompanied by a form of “to be”: adjectival uses of passive (e.g. in English “books written by Stephen King”) are treated automatically, whereas extraposed passive phrases, such as “Elaborated with the greatest discretion, this project...”, are handled during the manual correction of coordination/punctuation.

a single word. Though very good solutions exist to detect these automatically in a separate preprocessing step (see, for example, Constant *et al.* 2011), we have decided to split all these into their separate words in order not to have to depend on additional components.

Fortunately, the French Treebank also annotates the internal structure for many of these complex lexical lemmas, so we can find that “*dépôt de bilan*” has the internal structure [noun, preposition, noun] and use this to automatically annotate the expression according to the basic case discussed below, so this step requires little human intervention.

#### 4.2 *The basic case*

The heart of the algorithm binarizes the trees from the French Treebank and separates the daughters of a node into functors/heads, arguments, and modifiers. This step is done automatically, using a version of the classic “head percolation” table (Magerman 1994) similar to the ones used for other categorial grammar extraction algorithms (Hockenmaier and Steedman 2007; Moortgat and Moot 2001; Moot 2010a).

The automated part of the extraction algorithm recursively descends each node and successively performs each of the different transformations described here, as well as the refinements described in Sections 4.3 to 4.5. Thus, even though these cases are described separately for ease of exposition, they apply together at each node.

For example, the following sentence

- (3) le score correspondait à peine au tiers de l’ objectif  
the score corresponded barely to a third of the goal  
mensuel  
monthly  
‘the score barely corresponded to a third of the monthly goal’

has the French Treebank annotation shown in Figure 2. In the figure, the multiword expression “à peine” (*hardly*) has already been separated into its component words in the previous step of the algorithm.

The binarization step first selects the head of the constituent (the head percolation table first tries to find a verbal group VN as the head of a sentence SENT) and then combines it first with the sisters to its right, then with the sisters to its left, as shown in the figure below.

Figure 2:  
Initial French  
Treebank tree.

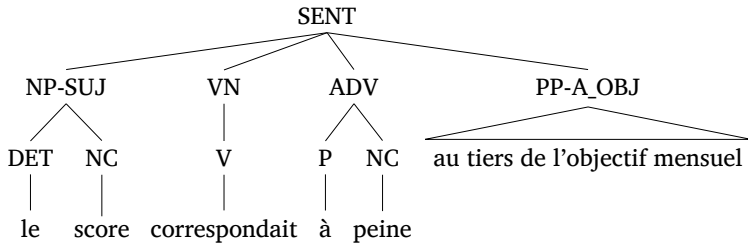
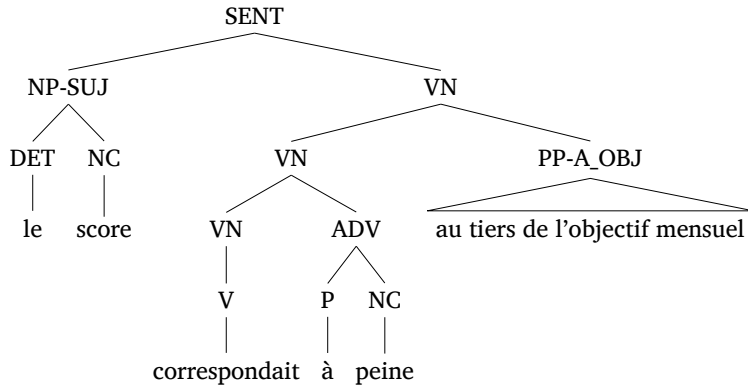


Figure 3:  
The tree of  
Figure 2 after  
binarization.



The label of the newly created nodes remains the same; VN in this case. The resulting tree, shown in Figure 3 has only unary and binary branches.

Next, a similar table of defaults decides for each binary branch if the pair of nodes concerned are a functor and its argument or a modifier and a category it modifies. So in the current example ADV is treated as a modifier whereas PP-A\_OBJ is treated as an argument. A functor and argument are given the formulas  $F/A$  and  $A$ , if the argument occurs on the right, or  $A$  and  $A\F$  if the argument occurs on the left, where  $F$  is the formula assigned to the parent node and  $A$  is the formula corresponding to the syntactic label of the argument node (this is again performed by looking up the values in a table, which indicates for example, that NP corresponds to  $np$  and PP-A\_OBJ corresponds to  $pp_a$ ). For modifiers, the modifier is assigned  $F/F$  if it occurs on the left and  $F\F$  if it occurs on the right, where  $F$  is the formula assigned to the parent node; the sister node of the modifier will therefore be assigned the same formula  $F$  as the parent node.

A type-logical treebank for French

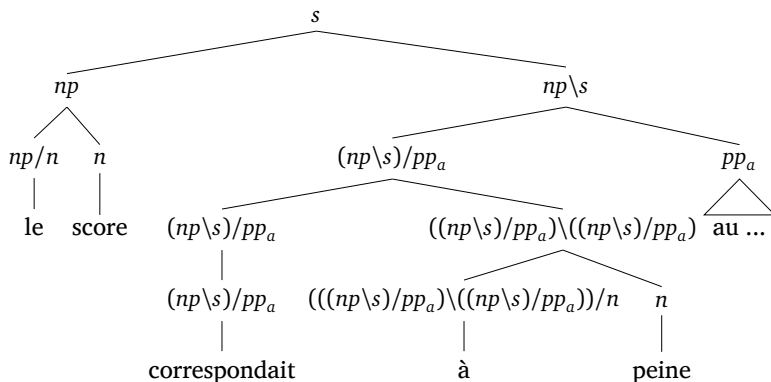


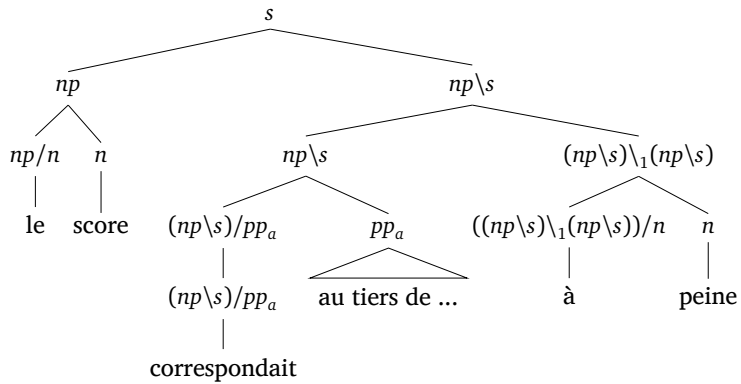
Figure 4: First derivation corresponding to Figure 3, using only elimination rules.

This translates the binarized tree of Figure 3 into the tree shown in Figure 4. This tree gives a full description of a derivation using only the elimination rules  $/E$  and  $\backslash E$ : suppressing the unary modes, we can label each pair of sisters uniquely with one of these rules by looking only at their formulas, either  $F/A$  and  $A$  or  $A$  and  $A\backslash F$ ; the distinction between modifiers and other functors is no longer relevant now, modifiers are simply those formulas where  $F = A$ .

So far, the extraction algorithm has followed the classic categorial grammar extraction methodology of Buszkowski and Penn (1990) and Moortgat and Moot (2001). However, the tree above gives a rather complicated formula to the modifier “à peine” (*hardly*). Moreover, this formula would change with the formula assigned to the verb it modifies – requiring a different formula for transitive verbs, intransitive verbs, auxiliary verbs, etc. – resulting in unnecessary duplication of lexical entries for all adverbs. As we have seen in Section 2 with the adverb “dangereusement” (*dangerously*), we can choose an infixation solution and treat all adverbs as VP modifiers as shown in Figure 5.

From this tree, we can again obtain a complete derivation, this time using the  $/E$ ,  $\backslash E$  and  $\backslash_1 E$  rules of Table 1, though we now need the word order of the original sentence to determine the position of the adverb. The  $\backslash_1 E$  rule essentially plays the role of the crossing composition rules used for similar situations in the CCGbank (Hockenmaier and Steedman 2007). This simplification is performed automatically whenever a complex verb-modifier formula would be assigned to an adverb.

Figure 5:  
A version of  
the derivation  
of Figure 4  
using a simpler  
lexical entry  
for the adverb  
“à peine”



## 4.3

## Verb clusters

As discussed in Section 3, verb clusters (which include clitics and some adverbs) and the arguments of verbs are sisters in the FTB annotation trees. While this wasn't a problem for the simple cases treated in the previous section, this becomes problematic in the case of a complex verbal group. Figure 6 shows an example corresponding to sentence (4) (Figure 1 back on page 235 requires a similar treatment).

- (4) Ils ont déjà pu constater que (...)  
they have already been able to note that

In a categorial setting, we obtain a much simpler analysis if the VN arguments are arguments of the embedded verbs instead: in the current case, we'd like the infinitival group to be the argument of the past participle “pu” (past participle of the verb “pouvoir”, *can*). At the bottom of Figure 6 we see the rightward branching structure which results from the corpus transformation. Note also how the adverb “déjà” (*already*) is assigned the VP-modifier formula  $(np\s_x)/(np\s_x)$  which is parametric for the type of sentence (in essence, this is a formula with an implicit first-order quantifier ranging over the different sentence types, see Moot 2014 or Moortgat 2011, Section 2.7; in the figure,  $x$  is instantiated to *ppart*).

The extraction script automatically rebrackets the verb clusters as indicated above and treats any arguments of the verb cluster as arguments of the final verb in the cluster. This step requires very few manual corrections.

4.4 Coordination and punctuation symbols

The sentences below illustrate some of the problems with coordination which we will discuss in this section.

- (5) Elles reprennent et amplifient des programmes existants  
 they resume and amplify programs existing  
 ou en cours d' adaptation  
 or currently being adapted
- (6) Les lieux où les deux derniers morts ont été  
 the places where the two last deaths have been  
 recensés, lundi 30 décembre, La Yougoslavie et La  
 reported, Monday 30 December, Yugoslavia and  
 Colombie, (...)  
 Colombia,

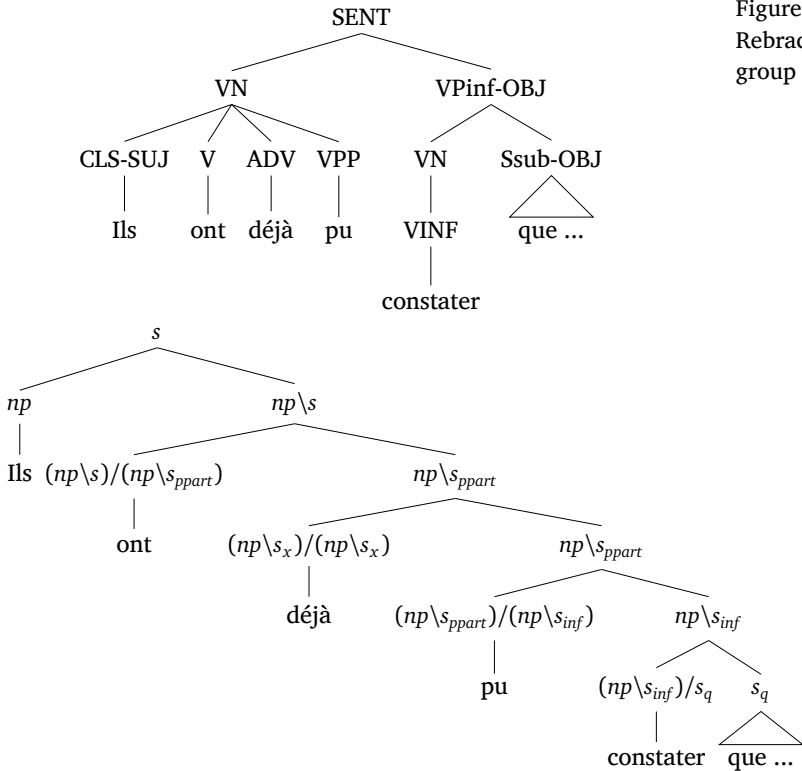


Figure 6:  
 Rebracketing a verbal  
 group and its arguments

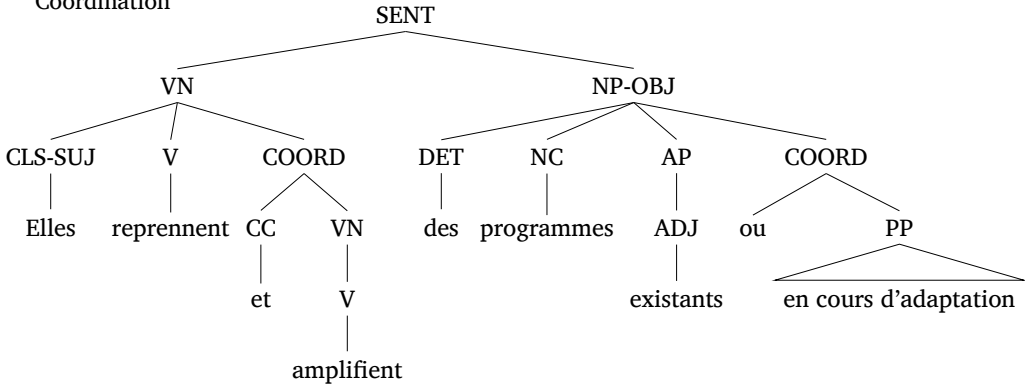
Figure 7:  
Coordination

Figure 7 shows the FTB syntactic structure of sentence (5). In categorial grammars, conjunctions like “ou” (*or*) are generally assigned instances of the formula  $(X \setminus X) / X$  (for a contextually appropriate choice of the formula  $X$ ). The first conjunction is of the two transitive verbs (instantiating  $X$  with the formula  $(np \setminus s_{main}) / np$ ) that share both the subject and the object. For the second coordination it is the adjective and the prepositional phrase which are conjoined (though this is not so clear from the annotation only, where it seems to be an unlike coordination between an  $np$  and a  $pp$ ). As is standard in categorial grammars, we assign both the adjective and the PP the formula  $n \setminus n$  (this is the standard assignment for a PP modifying a noun), turning this seemingly unlike coordination into a trivial instance of the general coordination scheme.

The (somewhat simplified) FTB annotation of sentence (6) of Figure 8 shows another problem: appositives, which are treated by assigning a coordination-like formula to the punctuation symbol preceding them (a similar solution is used for parentheticals and for most extrapositions).<sup>7</sup> An additional complication in this example is that we have

<sup>7</sup>Not all extrapositions can be analysed as coordinations this way. In the example below

- (i) A celà s'ajoute une considération générale : (...)  
to that adds-itself a general consideration

“A celà” is assigned  $s / (s / \diamond_1 \square_1 pp_a)$  allowing it to function as a long-distance  $pp$  argument to “s’ajoute”, as we have seen for the  $s / \diamond_1 \square_1 pp_{de}$  argument of “dont” in Section 2.



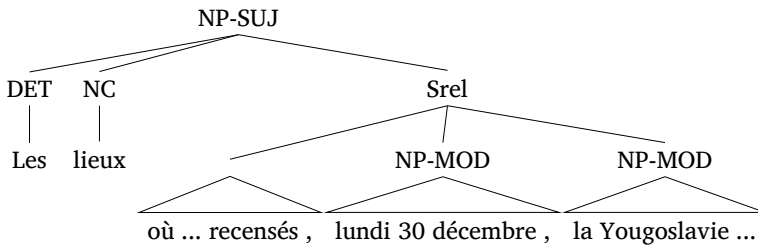


Figure 8:  
Appositives

to distinguish between the NP-MOD temporal adverb, which modifies the verb “recensés” (*reported*), and the NP-MOD for the appositive, which conjoins to “Les lieux” (*the places*) with the NP containing “la Yougoslavie” (*Yugoslavia*).

As the example shows, these cases are difficult to infer from the information provided by the FTB annotation alone, and therefore must be annotated manually; in total a bit more than 20% of the punctuation symbols – over ten thousand punctuation symbols – are assigned coordination-like categories. This complicated treatment of punctuation is not necessary for standard phrase structure parsers but given that in a categorial grammar analysis we want coordination-like punctuation to behave *semantically* like coordination, some special treatment of coordination is necessary.

More complex forms of coordination, such as right-node raising and gapping, require a more sophisticated treatment, which is discussed in Appendix B.

#### 4.5 Traces and long-distance dependencies

As an example of a simple long-distance dependency in the corpus, consider the example below.

- (7) Premier handicap auquel il convenait de s’attaquer:  
 first handicap to which it was agreed to attack:  
 l’inflation  
 the inflation

Figure 9 shows how the insertion of traces works. In the input structure on the top of the figure, “auquel” (*to which*) is assigned a preposition + pronoun POS-tag and assigned the role of a prepositional object with the preposition “à” (*to*). However, this preposition is an argument

of the verb “s’attaquer à” (*to attack*), which occurs much lower in the annotation tree. Since none of these dependencies are annotated in the French Treebank, the default automatic treatment assigns them as arguments of the next occurring verb. Even though this is a reasonable default, it still produces many errors. In the example above, it would assign the  $pp_a$  as argument of the main verb “convenait” (*to agree*), which is a possible assignment for this verb but is incorrect in the current case. As a consequence all relative pronouns, wh-pronouns, and clitics – a total of over 3,000 occurrences in the corpus – have been manually verified and, where necessary, corrected with the appropriate long-distance dependencies. At the bottom of Figure 9, the manually added long-distance dependency is shown (for reasons of horizontal space, the subproof of “de s’attaquer  $pp_a$ ” has been stretched, as indicated by the dots).

5

ANALYSIS

Categorial grammars, much like lexicalized tree adjoining grammars and other strongly lexicalized formalisms, use very construction-specific lexical entries. This means, for example, that when a verb can be used both as transitive and intransitive, it will have (at least) two distinct lexical entries. For extracted grammars, this generally means a very high level of lexical ambiguity.

Using the most detailed extraction parameters, the final lexicon uses 1,101 distinct formulas, though only 800 of these occur more than once and, 684 more than twice and 570 at least five times. The lion’s share of these rare formulas are assigned to frequently occurring words, such as “et” (*and*) and verbs, appearing in unusual syntactic constructions.

Using a slightly less detailed extraction (which, for example, distinguishes only  $pp_{de}$ ,  $pp_a$  and  $pp_{par}$  and uses simply  $pp$  for prepositional phrases headed by other prepositions) there are 761 different formulas used in the lexicon (of which only 684 occur more than once, 546 occur more than twice and 471 occur at least five times).

Even in this second lexicon, many frequent words have a great number of lexical assignments. The conjunction “et” (*and*) has 86 different lexical formulas, the comma “,” (which, as we have seen, often functions much like a conjunction) is assigned 72 distinct formulas,

the adverb “plus” (*more*) has 44 formulas (in part because of possible combinations with “que”, *than*), the prepositions “pour” (*for/to*), “en” (*in/while*) and “de” (*of/from*) have 43, 42 and 40 formulas respectively, and the verb “est” (*is*) has 39 formulas.

Although this kind of lexical ambiguity may seem like an important problem when using the extracted lexicon for parsing, well-known techniques such as *supertagging* (Bangalore and Joshi 2011), which assign the contextually most likely set of formulas (supertags) to each word, can be used to reduce the lexical ambiguity to an acceptable level. To give an idea of how effective this strategy is in the current context and with the reduced lexicon of 761 formulas: using the supertagger of Clark and Curran (2004) and assigning only the most

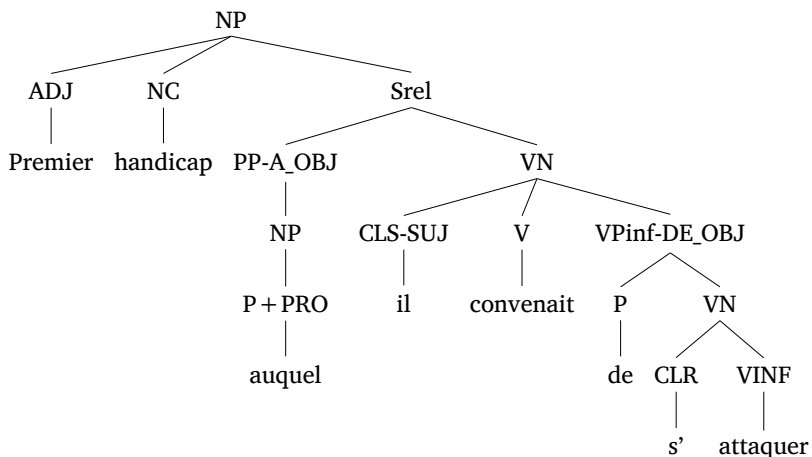


Figure 9:  
Adding traces to  
the output

$$\begin{array}{c}
 \frac{\text{attaquer}}{[Lex]} \quad \frac{[Hyp]_1}{p_0 \vdash pp_a} \quad [E]}{\frac{[Lex]}{cl_r \setminus (np \setminus s_i) / pp_a} \quad [E]} \quad \frac{s'}{[Lex]} \\
 \frac{de}{[Lex]} \quad \frac{[Lex]}{np \setminus s_{di} / (np \setminus s_i)} \quad \frac{[E]}{s' \circ (a \circ p_0) \vdash np \setminus s_i} \quad [E]}{\frac{[E]}{de \circ (s' \circ (a \circ p_0)) \vdash np \setminus s_i}} \\
 \vdots \\
 \frac{il}{np} \quad \frac{[Lex]}{(np \setminus s) / (np \setminus s_{di})} \quad \frac{[E]}{de \circ (s' \circ (a \circ p_0)) \vdash np \setminus s_i} \quad [E]}{\frac{[E]}{c \circ (de \circ (s' \circ (a \circ p_0))) \vdash np \setminus s}} \\
 \frac{auquel}{[Lex]} \quad \frac{[I]_1}{(n \setminus n) / (s / \diamond_1 \square_1 pp_a)} \quad \frac{[E]}{il \circ (c \circ (de \circ (s' \circ a))) \vdash s / \diamond_1 \square_1 pp_a}}{\frac{[E]}{auquel \circ (il \circ (c \circ (de \circ (s' \circ a)))) \vdash n \setminus n}}
 \end{array}$$

likely formula to each word, 90.6% of the words are assigned the correct formula. When assigning each word all formulas with probability greater than 1% of the most likely supertag (for an average of 2.3 formulas per word), the supertagger assigns the correct formula to 98.4% of all words (for the FTB section of the TLGbank, using ten-fold cross-validation).

Supertagging does not solve the problem of data sparseness: for the supertagger, formulas which are seen only once or twice in the training data are not fundamentally different from formulas which do not occur at all. However, since these are exceptional cases, this has little effect on the coverage of the parser: Clark and Curran (2007) use only categories occurring at least 10 times for their parser based on the CCGbank and still obtain 99.58% coverage on unseen sentences.

We will discuss the performance of the supertagger in more detail, especially on sentences *outside* of the French Treebank, while discussing bootstrapping in Section 7.1.

## 6 COMPARISON WITH THE CCGBANK

Apart from the obvious theoretical differences between CCG and type-logical grammars and the different treatment of certain linguistic phenomena – such as extraction – that this implies, it is worth spending some time on some of the less obvious differences between the two treebanks.

Whereas the CCGbank uses a certain number of rules besides the standard combinatory schemata – notably for extraposition and coordination,<sup>8</sup> but also to transform passives  $np \setminus s_{pass}$  into adjectives  $n \setminus n$  and (bare) nouns  $n$  into noun phrases  $np$  – the TLGbank uses no non-logical rules. As a result, the lexicon of the type-logical treebank does more of the work. The lexicon is bigger and consequently, the tasks of the supertagger and the parser are more difficult in comparison with the CCG supertagger (Clark and Curran 2007). The supertagger’s precision is similar – 98.4% correct in both cases – though the number

---

<sup>8</sup>To give an idea of the form of these rules, there is an extraposition rule transforming “ $np$ ” (that is, a noun phrase followed by a comma) into a sentence modifier  $s/s$  and a set of rules transforming constructions like “ $X$  and  $X$ ” (that is, the word “and” occurring between two expression of the same category  $X$ ) to  $X$ , see Section 2.5.5 of Hockenmaier and Steedman (2005) for more details.

of lexical formulas per word is higher – 2.3 for the TLGbank versus 1.7 for the CCGbank. The number of lexical formulas per word is an important factor for parsing speed.

If we want to reduce the size of the lexicon in a way similar to the CCGbank, there are two basic options:

1. the first option is to allow non-logical rules of the same style as those used for the CCGbank,
2. the second option, more in line with the general spirit of type-logical grammars, is to exploit the derivability relation and to replace the analysis of passives by a formula  $F$  such that  $F \vdash n \setminus n$  (see Section 4.4.2 of Morrill 2011 for a particularly nice solution).

Since reducing the lexical ambiguity increases parsing speed but adding rules (as in option 1) or complicating the formulas (as in option 2) will reduce it, a careful evaluation of the benefits should be made. We leave to future research the transformation of the TLGbank in these two ways.

## 7

## TOOLS AND RESOURCES

To facilitate annotation, correction, and parsing, several tools have been developed, using a combination of Prolog and TclTk. In addition, several well-known tools have been used for the exploitation of the corpus: the Stanford Tregex tool (Levy and Andrew 2006) for browsing and querying the French Treebank (as well as some of its transformations), Leff (Sagot 2010) for lemmatizing and related tasks, the C&C tools (Clark and Curran 2004) for training POS-tag and supertag models using the annotated corpus, and a chart parser strongly inspired by Shieber *et al.* (1995) for parsing with the resulting grammar.

Figure 10 shows a screenshot of the interface to the supertagger and parser. This “horizontal” interface allows the user to type in sentences and see the resulting semantic output from the parser. The darker-shaded percentage of the block to the left of the formula gives a visual indication of the probability assigned to the formula (the exact numbers can be seen by moving the mouse over the corresponding area). Apart from some configuration options, this interface is not interactive.

Figure 10:  
Screenshot of  
the supertagger  
interface

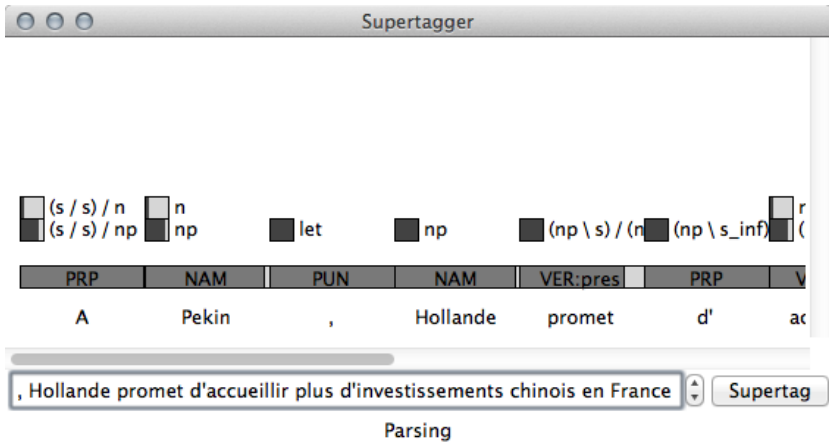


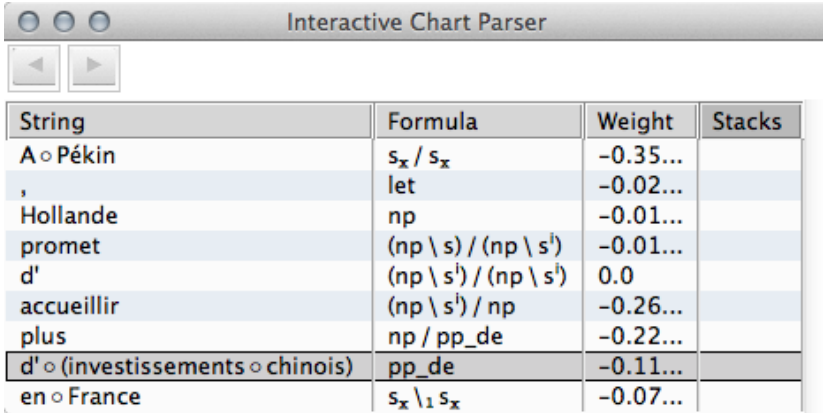
Figure 11 shows a screenshot of the “vertical” interface to the parser and supertagger. This is an interactive interface, allowing the user to select (or type in) the desired formula – to help prevent errors, the current frequency of the chosen formula for the current word is displayed after a manual choice of the formula – as well as allowing the user to select the parser rule applications by clicking on one of the premises for a rule (an additional dialog pops up if the rule choice is ambiguous, which happens infrequently). The weight column shows the log-probability of the item.<sup>9</sup>

### 7.1 Bootstrapping

Given that the French Treebank is somewhat small compared to other treebanks and given that the conversion of the FTB to the type-logical treebank was rather labour-intensive, it makes sense to look at more efficient ways of increasing the size of the treebank. The tools described in the previous section, interfacing with the supertagger and the parser for the core corpus are useful in this respect.

Currently, slightly over 1,600 additional sentences have been annotated (for a total annotated corpus of 14,539 sentences and 421,348

<sup>9</sup>The current implementation of the parser is not statistical in the sense that the rule applications do not have a probability assigned to them (the supertags do, so the parser outputs the first parse found for the most probable combination of supertags which allows a parse). However, the source code has the required hooks to add a probability model for the rule applications, whereas the required probabilities can be estimated from the treebank itself.



String	Formula	Weight	Stacks
A ◦ Pékin	$s_x / s_x$	-0.35...	
,	let	-0.02...	
Hollande	np	-0.01...	
promet	$(np \setminus s) / (np \setminus s^i)$	-0.01...	
d'	$(np \setminus s^i) / (np \setminus s^i)$	0.0	
accueillir	$(np \setminus s^i) / np$	-0.26...	
plus	np / pp_de	-0.22...	
d' ◦ (investissements ◦ chinois)	pp_de	-0.11...	
en ◦ France	$s_x \setminus_1 s_x$	-0.07...	

Figure 11: Screenshot of the interactive parser

words). Most of these sentences come from the Sequoia treebank (Candito and Seddah 2012) and the French Timebank (Bittar 2010). The observed accuracy of the supertagger for these sentences from the *L'Est Républicain* newspaper is slightly lower than the results reported in Section 5: in 88.1% of cases, the best supertag is correct, and in 97.6% of cases the correct supertag has probability greater than 1% of the best supertag (as compared to 90.6% and 98.4% respectively for the cross-validated results). Part of this difference might be attributable to stylistic differences between the two newspapers (initial experiments with annotating unseen sentences from *Le Monde* seem to confirm this) but it may also be the case that cross-validation gives a somewhat optimistic picture of actual performance on unseen data from other sources (the different training and test sets not being completely independent).

Table 2 shows the accuracy of the Part-of-Speech tagger and of the supertagger for the different sub-corpora. The columns POS and Super list the accuracy of the Part-of-Speech tagger and of the supertagger respectively for the different corpora. Performance degrades gracefully for the different newspaper corpora (the French treebank and more modern articles in *Le Monde* being presumably the most similar, whereas the articles in *L'Est Républicain* from Sequoia and the French Timebank have a slightly reduced supertagger performance) but it shows a somewhat more important reduction for the literary corpus of travelogues in the Pyrenees of Itipy (Lefeuvre *et al.* 2012; Moot 2012).

Table 2:  
Supertagger and  
Part-of-Speech tagger  
performance on the  
different sections  
of the corpus

Corpus	POS	Super	0.1	0.01	F/w
French Treebank	97.8	90.6	96.4	98.4	2.3
Le Monde 2010	97.3	89.9	95.8	97.9	2.2
L'Est Républicain	97.3	88.1	94.8	97.6	2.4
Itipy/Forbes	95.7	86.7	93.8	97.1	2.6

The 0.1 and 0.01 columns indicate the supertagger's performance when all supertags with probability greater than  $\beta$  ( $=0.1$  or  $0.01$ ) times the probability of the most likely supertag have been included. The column F/w indicates how many supertags this is per word for  $\beta = 0.01$  (for  $\beta = 0.1$  this number is around 1.4). We can see that even though the supertagger's performance for the best supertag (in the Super column) reduces steadily – from 90.6 on the main corpus to 86.7 on the Itipy corpus, a 3.9 percentage points difference – when using multiple supertags, this difference is greatly reduced (from 98.4 to 97.1, a 1.3 percentage points difference).

Even in the more difficult context of the Itipy corpus, the parser/supertagger combination (with  $\beta = 0.01$ ) finds a complete analysis for 88.6% of the sentences in this subcorpus. We expect this figure to improve when better search heuristics, such as those described by Clark and Curran (2007), are used to deal with the increased number of formulas per word. To give an indication that even the current parser implementation performs well: the only other parsing statistics I've seen for the Itipy corpus are given by Nguyen (2012), who reports that a total of 18.5% of the sentences in the Itipy corpus were successfully parsed using an off-the-shelf parser.

## 7.2

### *Availability*

All the tools and resources are available from the author under the GNU Lesser General Public License.

<http://richardmoot.github.io/TLGbank/>

An unfortunate exception to this is the main part of the Typological Treebank itself: being a derived work of the French Treebank, it is available only to those who have a license for the original treebank (contact to author for access to the private Git). The Sequoia part of the treebank and the models derived from the complete treebank are freely available, however.



## CONCLUSION

We have shown how the French Treebank has been semi-automatically transformed into a set of derivations in multimodal type-logical grammars: the TLGbank. This is an important first step in training and evaluating wide-coverage type-logical parsers and we hope to see several competitive type-logical parsers in the future.

## ACKNOWLEDGMENTS

I would like to thank Michael Moortgat and Noémie-Fleur Sandillon-Rezer for our work together on similar grammar extraction tasks. In addition, Michael Moortgat’s insights on how to design multimodal type-logical grammars have deeply influenced all aspects of the design of the current treebank.

I also thank Yannick Parmentier and Denys Duchier for organizing the ESSLI 2013 workshop in Düsseldorf where I presented this material and all workshop participants for their feedback.

I would also like to thank the anonymous referees for their many useful comments.

I, of course, take full responsibility for any remaining errors.

## APPENDIX

## A COMPLETE LOGICAL RULES

Table 3 lists the full set of rules for multimodal categorial grammars. Binary modes  $i$  range over  $\{\epsilon, 1, 2, 3, l, r\}$  – although we will continue to write  $X \circ_{\epsilon} Y$  as  $X \circ Y$  and  $A /_{\epsilon} B$  as  $A/B$ , etc. – and unary modes  $j$  range over  $\{0, 1, l, r\}$ .

A.1 *The unary connectives*

The rules for  $\diamond$  and  $\square$  may require some additional explanation for people unused to multimodal type-logical grammars. Whereas the rules for  $\bullet$ ,  $/$ , and  $\backslash$  produce binary trees labelled by indices — with the  $\bullet I$ ,  $/E$ , and  $\backslash E$  rules *constructing* trees (i.e. combining previously derived trees  $X$  and  $Y$  into a single tree  $X \circ_j Y$ ) and with the  $\bullet E$ ,  $/I$ , and  $\backslash I$  rules *removing* binary branches — the rules for  $\square$  and  $\diamond$  produce and remove unary branches. So the  $\square E$  rule states that if we have previ-

Table 3:  
Full set of logical rules  
for multimodal type-logical  
grammar

$\frac{}{w \vdash A} \text{Lex}$	$\frac{}{x \vdash A} \text{Hyp}$
$\frac{X \vdash A/iB \quad Y \vdash B}{X \circ_i Y \vdash A} /E$	$\frac{X \vdash B \quad Y \vdash B \setminus_i A}{X \circ_i Y \vdash A} \setminus E$
$\frac{x \vdash B \quad \vdots \quad X \circ_i x \vdash A}{X \vdash A/iB} /I$	$\frac{x \vdash B \quad \vdots \quad x \circ_i X \vdash A}{X \vdash B \setminus_i A} \setminus I$
$\frac{x \vdash A \quad y \vdash B \quad \vdots \quad Y \vdash A \bullet_i B \quad X[x \circ_i y] \vdash C}{X[Y] \vdash C} \bullet E$	$\frac{X \vdash A \quad Y \vdash B}{X \circ_i Y \vdash A \bullet_i B} \bullet I$
$\frac{X \vdash \Box_j A}{\langle X \rangle^j \vdash A} \Box E$	$\frac{\langle X \rangle^j \vdash A}{X \vdash \Box_j A} \Box I$
$\frac{x \vdash A \quad \vdots \quad Y \vdash \Diamond_j A \quad X[\langle x \rangle^j] \vdash C}{X[Y] \vdash C} \Diamond E$	$\frac{X \vdash A}{\langle X \rangle^j \vdash \Diamond_j A} \Diamond I$

ously derived  $X$  to be of type  $\Box_j A$ , then  $\langle X \rangle^j$  is of type  $A$ ; we remove the  $\Box_j$  connective and add a unary branch labelled by the index  $j$ . Symmetrically, the  $\Box I$  rule states that if we have derived  $\langle X \rangle^j$  (i.e. we have an initial unary branch labelled  $j$  with a daughter subtree  $X$ ) to be of type  $A$  then the tree  $X$  by itself is of type  $\Box_j A$ .

The elimination rules for the product  $\bullet$  and the diamond  $\Diamond$  may appear a bit odd: they are similar to the disjunction elimination rule in intuitionistic logic and involve an arbitrary formula  $C$ . The  $\Diamond E$  rule gives instructions on how to use a formula  $\Diamond_j A$  once we have derived it (as the subproof of the left premise of the rule) by stating that if we can use a formula  $A$  labelled with a fresh variable  $x$  to derive any tree  $X$  (of any formula  $C$ ) such that this  $x$  corresponding to  $A$  occurs as a leaf with a unary branch labelled  $j$  as its immediate parent (as indicated by the tree term  $X[\langle x \rangle^j]$ ), then we can conclude that this tree  $X$  with the unary branch  $j$  and leaf  $x$  replaced by  $Y$  (the tree corresponding to  $\Diamond_j A$ ) is also a tree

Table 4:  
Structural rules

**Infixation**

$$\frac{V[(X \circ Y) \circ_1 Z] \vdash C}{V[X \circ (Y \circ_1 Z)] \vdash C} MA \qquad \frac{V[(X \circ Y) \circ_1 Z] \vdash C}{V[(X \circ_1 Z) \circ Y] \vdash C} MC$$

**Extraction**

$$\frac{V[X \circ (Y \circ \langle Z \rangle^1)] \vdash C}{V[(X \circ Y) \circ \langle Z \rangle^1] \vdash C} MA\Diamond_1 \qquad \frac{V[(X \circ \langle Z \rangle^1) \circ Y] \vdash C}{V[(X \circ Y) \circ \langle Z \rangle^1] \vdash C} MC\Diamond_1$$

**Left-node raising/right-node raising**

$$\frac{V[(\langle X \rangle^0 \circ Y) \circ Z] \vdash C}{V[\langle X \rangle^0 \circ (Y \circ Z)] \vdash C} MA_l\Diamond_0 \qquad \frac{V[X \circ (Y \circ \langle Z \rangle^0)] \vdash C}{V[(X \circ Y) \circ \langle Z \rangle^0] \vdash C} MA_r\Diamond_0$$

**In situ binding**

$$\frac{V[X \circ \langle Y \rangle^2] \vdash C}{V[(X)^r \circ_2 Y] \vdash C} I_{2r} \qquad \frac{V[\langle X \rangle^2 \circ Y] \vdash C}{V[\langle Y \rangle^l \circ_2 X] \vdash C} I_{2l}$$

$$\frac{V[X \circ (Y \circ_2 Z)] \vdash C}{V[(X \circ_r Y) \circ_2 Z] \vdash C} MA_{2r} \qquad \frac{V[(X \circ_2 Z) \circ Y] \vdash C}{V[(X \circ_l Y) \circ_2 Z] \vdash C} MC_{2l}$$

**Quoted speech**

$$\frac{V[(X \circ_3 Y) \circ Z] \vdash C}{V[X \circ_3 (Y \circ Z)] \vdash C} MA_3 \qquad \frac{V[Y \circ (X \circ_3 Z)] \vdash C}{V[X \circ_3 (Y \circ Z)] \vdash C} MC_3$$

of type  $C$ . In other words,  $X[\langle x \rangle^j]$  becomes  $X[Y]$  as indicated in the rule.

As an example, we show that if a tree  $Y$  is of type  $\Diamond_j \Box_j A$  then this tree is also of type  $A$  (for all formulas  $A$  and unary indices  $j$ ), as already alluded to in Section 2.

$$\frac{\frac{Y \vdash \Diamond_j \Box_j A \quad \frac{x \vdash \Box_j A}{\langle x \rangle^j \vdash A} \Box E}{Y \vdash A} \Diamond E}{Y \vdash A} \Diamond E$$

If  $Y$  is of type  $\Diamond_j \Box_j A$ , then we start the subproof on the right using the hypothesis  $x$  of type  $\Box_j A$ . Then we apply the elimination rule for  $\Box$  to produce the tree  $\langle x \rangle^j$  of type  $A$ . But now, we are immediately in the right configuration to apply the  $\Diamond E$  rule (it is the special case

where the context  $X[\ ]$  is empty) and this allows us to replace  $\langle x \rangle^j$  by  $Y$ , thereby proving that  $Y$  is of type  $A$  as required.

## A.2 *The structural rules*

Although these patterns of derivability are interesting and can be used to give accounts of case and other forms of subtyping (Bernardi and Moot 2003), our interest here lies in the fact that they give access to structural rules which can rearrange our derived trees in controlled ways. The structural rules are listed in Table 4. The double line for the in situ binding rules indicate that these rules can be applied in both directions: top-to-bottom and bottom-to-top.

Even though this looks like a rather large list, these are principally instantiations of the well-known universal rule schemata of mixed associativity and mixed commutativity (see Moortgat 2011 and Moot and Retoré 2012 for commentary, and Vermaat 2005 for arguments that these structural rules are truly universal).

For the grammar engineer, the structural rules give us great flexibility and modularity when designing our grammars (although it could be argued that there is *too* much flexibility to this). However, the account given for different linguistic phenomena follows the conventional wisdom of categorial grammars and, as discussed in the next subsection, our annotation choices have been designed to be compatible with other modern type-logical grammars. So there has been a conscious choice not to create the smallest possible lexicon (at the cost of additional structural rules) but to keep the set of structural rules to the current set of instantiations of well-known schemata.

The abbreviated proof from Section 2, is repeated below.

$$\frac{\frac{\text{appauvrit} \vdash (np \setminus s) / np \quad la \circ CGT \vdash np}{\text{appauvrit} \circ (la \circ CGT) \vdash np \setminus s} /E \quad \text{dangereusement} \vdash (np \setminus s) \setminus_1 (np \setminus s)}{(\text{appauvrit} \circ_1 \text{dangereusement}) \circ (la \circ CGT) \vdash np \setminus s} /E$$

Using the structural rules of Table 4, this proof looks as follows.

$$\frac{\frac{\text{appauvrit} \vdash (np \setminus s) / np \quad la \circ CGT \vdash np}{\text{appauvrit} \circ (la \circ CGT) \vdash np \setminus s} /E \quad \text{dangereusement} \vdash (np \setminus s) \setminus_1 (np \setminus s)}{(\text{appauvrit} \circ (la \circ CGT)) \circ_1 \text{dangereusement} \vdash np \setminus s} /E \quad MC$$







de ... et les quasi-fonds propres de ...” (abbreviated as  $e$  in the proof below) as being of type  $np \bullet \diamond_0 \square_0 pp$  using an application of the  $/E$  rule followed by an application of the  $\backslash E$  rule. We can then combine this  $np \bullet \diamond_0 \square_0 pp$  constituent with the verb “augmenter” (abbreviated as  $a$ ) as follows.

$$\frac{\frac{\frac{a}{((np \backslash s)/pp)/np} \text{Lex} \quad \frac{x \vdash np}{x \vdash np} \text{Hyp} \quad \frac{z \vdash \square_0 pp}{z \vdash \square_0 pp} \text{Hyp}}{\frac{a \circ x \vdash (np \backslash s)/pp}{\langle z \rangle^0 \vdash pp} \square E} \quad /E}{\frac{\frac{y \vdash \diamond_0 \square_0 pp}{y \vdash \diamond_0 \square_0 pp} \text{Hyp} \quad \frac{(a \circ x) \circ \langle z \rangle^0 \vdash np \backslash s}{a \circ (x \circ \langle z \rangle^0) \vdash np \backslash s} \text{MA}_r \diamond_0}{a \circ (x \circ y) \vdash np \backslash s} \diamond E} \quad \bullet E} e \vdash np \bullet \diamond_0 \square_0 pp \quad a \circ e \vdash np \backslash s$$

The  $\diamond_0 \square_0 pp$  formula allows us to use the right-node raising rule of Section B.1. The proof would be slightly simpler if we assigned the word “augmenter” the formula  $(np \backslash s)/(np \bullet pp)$  instead (such an analysis can also be found on page 19 of Morrill 2011). However, since we have already found independent motivation for the right-node raising rules, we have chosen to give the verb the more classical analysis of  $((np \backslash s)/pp)/np$ .

#### B.4

#### Gapping

The extraction/inflection rules are used for the analysis of gapping, as shown in sentence (12) below, where the transitive verb “atteindre” is absent from the second clause.

- (12) Le salaire horaire atteint dorénavant 34,06 francs et  
the wages per hour reach from now on 34.06 francs and  
le SMIC mensuel brut [tv] 5756,14 francs.  
the gross minimum monthly wage [tv] 5756.14 francs.  
‘Hourly wages now reach 34.06 francs and the monthly minimum wage 5756.14 francs.’

We use the multimodal approach first proposed by Hendriks (1995) and then advanced by Moortgat (1996). Schematically, the formulas for gapping are of the following form

$$((s/_2 \square_2 X) \backslash_i (s/_2 X)) / (s / \diamond_1 \square_1 X)$$

with  $X$  being a formula for a verb, for example  $X = (np \backslash s)/np$  for a



transitive verb.<sup>10</sup> This formula indicates that first a sentence missing a transitive verb to its right is selected (this is the extraction scheme we have seen before, though no longer restricted to right branches), then a sentence missing a transitive verb to its left, but keeping track of the *position* of this missing transitive verb in the sentence – this is implemented using the *l* and *r* modes which indicate whether the extracted verb is on the left or on the right of the current node. Finally, we *insert* a transitive verb at the position of this missing transitive verb on the left.

Even though this may seem like a rather roundabout way of achieving the desired sentence – first moving the transitive verb out, then moving it back into its original place – it has the important advantage of allowing us to get the semantics right; we know the verb from the first sentence and can therefore use it in the semantics, whereas a simpler type such as  $(s \setminus s) / (s / \diamond_1 \square_1 X)$  would not allow us to obtain the correct semantics.

In addition, abstracting away from the mode information and the unary connectives, the current analysis is an instantiation of the universal coordination formula  $(Y \setminus Y) / Y$  when we choose  $Y = s / X$ , giving  $((s / X) \setminus (s / X)) / (s / X)$ .

The extraction part of the gapping proof proceeds as shown below; *s* abbreviates “le salaire horaire” and *f* abbreviates “34,06 francs”.

$$\frac{\frac{\frac{z \vdash \square_2((np \setminus s) / np)}{\langle z \rangle^2 \vdash (np \setminus s) / np} \text{Hyp} \quad \frac{f \vdash np}{f \vdash np} \text{□E}}{s \vdash np \quad \langle z \rangle^2 \circ f \vdash np \setminus s} \text{/E}}{\frac{s \circ (\langle z \rangle^2 \circ f) \vdash s}{s \circ (\langle f \rangle^l \circ_2 z) \vdash s} \text{I}_{2l} \quad \frac{s \circ (\langle f \rangle^l \circ_2 z) \vdash s}{(s \circ_r \langle f \rangle^l) \circ_2 z \vdash s} \text{MA}_{2r}}{s \circ_r \langle f \rangle^l \vdash s /_2 \square_2((np \setminus s) / np)} \text{/I}_1} \text{□E}$$

We move the hypothetical  $\square_2((np \setminus s) / np)$  out, but keep track of where

<sup>10</sup> More precisely, the instantiation of the schema we need is

$$((s /_2 \square_2((np \setminus s) / np)) \setminus_1 (s /_2((np \setminus s) / \diamond_0 \square_0 np))) / (s / \diamond_1 \square_1((np \setminus s) / np))$$

with the  $\diamond_0 \square_0 np$  permitting right-node raising (associativity) as we have seen it in Section B.1.

we have used it: from the bottom, we started left of  $f$  (leaving  $l$  as a unary branch there), then right ( $r$ ).

Consequently, to get back from the top, we first go right ( $r$ ) and finally left ( $l$ ), ending up between  $s$  and  $f$  as required: we can then insert “atteint” of type  $(np \setminus s) / np$ , removing the trail of  $l$  and  $r$  during the process, as follows.<sup>11</sup>

$$\frac{(s \circ_r \langle f \rangle^l) \circ_l (et \dots) \vdash s /_2 ((np \setminus s) / np) \quad a \vdash (np \setminus s) / np}{\frac{\frac{\frac{((s \circ_r \langle f \rangle^l) \circ_l (et \dots)) \circ_2 a \vdash s}{((s \circ_r \langle f \rangle^l) \circ_2 a) \circ (et \dots) \vdash s} \quad MC_{2l}^{-1}}{(s \circ (\langle f \rangle^l \circ_2 a)) \circ (et \dots) \vdash s} \quad MA_{2r}^{-1}}{s \circ (\langle a \rangle^2 \circ f) \circ (et \dots) \vdash s} \quad I_{2l}^{-1}} \quad /E$$

B.5

*Quoted speech*

We need some special rules to treat past-perfect quoted speech, as shown in sentence (14) below. The parenthesized sentence is argument of the past participle “ajouté” and, in addition, this argument is discontinuous.

- (13) [<sub>s</sub> L’indice composite (...) a baissé de 0,3% en  
 [<sub>s</sub> the index composite has descended 0.3% in  
 novembre ], a annoncé mardi 31 décembre le  
 November ], has announced Tuesday 31 December the  
 département du commerce.  
 Department of Commerce.  
 ‘The composite index fell 0.3% in November, announced the  
 Department of Commerce on Tuesday December 31st.’
- (14) [<sub>sl</sub> Les conservateurs], a ajouté le premier ministre ...,  
 [<sub>sl</sub> the Conservatives], has added the Prime Minister,  
 [<sub>sr</sub> “ne sont pas des opportunistes qui virevoltent d’une  
 [<sub>sr</sub> “ are not opportunists who flip-flop from one  
 politique à l’autre ]  
 policy to another ]

The solution is essentially to analyse the entire verb group missing the  $s$  argument “a ajouté  $np$ ” as  $s_{main} \setminus_3 s_{main}$ , the structural rules the

<sup>11</sup>The  $-1$  as superscript to the rule names, e.g. in  $I_{2l}^{-1}$ , indicates that we apply the structural rules from *in situ binding* section of Table 4 in the “inverse” sense, i.e. bottom-up.

allow this entire group to move to the required position in the final string.

To illustrate this basic idea, we show how the structural rules for quoted speech allow us to derive “a ajouté np” (for some np) as  $s_{main} \backslash_3 s_{main}$ .

$$\begin{array}{c}
 \frac{\frac{a}{(s/np)/(np \backslash_3 s_{ppart})} \text{Lex}}{a \circ (x \circ_3 \text{ajouté}) \vdash s/np} \text{Lex} \quad \frac{\frac{\frac{x \vdash s}{\text{Hyp}} \quad \frac{\text{ajouté}}{s \backslash_3 (np \backslash_3 s_{ppart})} \text{Lex}}{x \circ_3 \text{ajouté} \vdash np \backslash_3 s_{ppart}} \backslash E}{np \vdash np} /E \\
 \hline
 \frac{\frac{(a \circ (x \circ_3 \text{ajouté})) \circ np \vdash s}{(x \circ_3 (a \circ \text{ajouté})) \circ np \vdash s} MC_3}{x \circ_3 ((a \circ \text{ajouté}) \circ np) \vdash s} MA_3}{(a \circ \text{ajouté}) \circ np \vdash s \backslash_3 s} \backslash I
 \end{array}$$

## REFERENCES

- Anne ABEILLÉ, Lionel CLÉMENT, and Alexandra KINYON (2000), Building a treebank for French, in *Proceedings of the Second International Language Resources and Evaluation Conference*, pp. 87–94, Athens.
- Abhishek ARUN and Frank KELLER (2005), Lexicalization in crosslinguistic probabilistic parsing: the case of French, in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pp. 306–313, Ann Arbor, Michigan.
- Srinivas BANGALORE and Aravind JOSHI (2011), *Supertagging: Using Complex Lexical Descriptions in Natural Language Processing*, MIT Press, Cambridge, Massachusetts.
- Raffaella BERNARDI and Richard MOOT (2003), Generalized quantifiers in declarative and interrogative sentences, *Logic Journal of the IGPL*, 11(4):419–434.
- André BITTAR (2010), *Building a TimeBank for French: A Reference Corpus Annotated According to the ISO-TimeML Standard*, Ph.D. thesis, Université Paris Diderot.
- Johan BOS, Stephen CLARK, Mark STEEDMAN, James R. CURRAN, and Julia HOCKENMAIER (2004), Wide-coverage semantic representation from a CCG parser, in *Proceedings of the 20th International Conference on Computational Linguistics (COLING-2004)*, pp. 1240–1246, Geneva.
- Johan BOS, James R. CURRAN, and Edoardo GUZZETTI (2007), The Pronto QA system at TREC-2007: harvesting hyponyms, using nominalisation patterns, and computing answer cardinality, in E. M. VOORHEES and L. P. BUCKLAND,

editors, *The Sixteenth Text REtrieval Conference, TREC 2007*, pp. 726–732, Gaitersburg, Maryland.

Johan BOS and Katja MARKERT (2005), Recognising textual entailment with logical inference, in *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*, pp. 628–635.

Wojciech BUSZKOWSKI and Gerald PENN (1990), Categorical grammars determined from linguistic data by unification, *Studia Logica*, 49:431–454.

Marie CANDITO, Benoît CRABBÉ, Pascal DENIS, and François GUÉRIN (2009), Analyse syntaxique du français : des constituants aux dépendances, in *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, Senlis.

Marie CANDITO and Djamé SEDDAH (2012), Le corpus Sequoia : annotation syntaxique et exploitation pour l’adaptation d’analyseur par pont lexical, in *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, pp. 321–334, Grenoble.

Bob CARPENTER (1991), Categorical grammars, lexical rules and the English predicative, in Robert LEVINE, editor, *Formal Grammar: Theory and Practice*, number 2 in Vancouver Studies in Cognitive Science, pp. 168–242, University of British Columbia Press, Vancouver.

Stephen CLARK and James R. CURRAN (2004), Parsing the WSJ using CCG and log-linear models, in *Proceedings of the 42nd annual meeting of the Association for Computational Linguistics (ACL-2004)*, pp. 104–111, Barcelona.

Stephen CLARK and James R. CURRAN (2007), Wide-coverage efficient statistical parsing with CCG and log-linear models, *Computational Linguistics*, 33(4):493–552.

Mathieu CONSTANT, Isabelle TELLIER, Denys DUCHIER, Yoann DUPONT, Anthony SIGOGNE, and Sylvie BILLOT (2011), Intégrer des connaissances linguistiques dans un CRF : application à l’apprentissage d’un segmenteur-étiqueteur du français, in *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, Montpellier.

Ane DYBRO-JOHANSEN (2004), *Extraction Automatique de Grammaires à Partir d’un Corpus Français*, Master’s thesis, Université Paris 7.

Bruno GUILLAUME and Guy PERRIER (2012), Semantic annotation of the French Treebank with modular graph rewriting, in *Proceedings of the Proceedings of META-RESEARCH Workshop on Advanced Treebanking (LREC’12)*, pp. 14–21, Istanbul.

Petra HENDRIKS (1995), Ellipsis and multimodal categorial type logic, in Glyn MORRILL and Richard T. OEHRLE, editors, *Proceedings of Formal Grammar 1995*, pp. 107–122, Barcelona.

Julia HOCKENMAIER and Mark STEEDMAN (2005), CCGbank: users’s manual, Technical report, Department of Computer and Information Science, University of Pennsylvania.

Julia HOCKENMAIER and Mark STEEDMAN (2007), CCGbank, a corpus of CCG derivations and dependency structures extracted from the Penn Treebank, *Computational Linguistics*, 33(3):355–396.

Joachim LAMBEK (1958), The mathematics of sentence structure, *American Mathematical Monthly*, 65:154–170.

Anaïs LEFEUVRE, Richard MOOT, Christian RETORÉ, and Noémie-Fleur SANDILLON-REZER (2012), Traitement automatique sur corpus de récits de voyages pyrénéens : une analyse syntaxique, sémantique et temporelle, in *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, Grenoble.

Roger LEVY and Galen ANDREW (2006), Tregex and Tsurgeon: tools for querying and manipulating tree data structures, in *5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa.

David M. MAGERMAN (1994), *Natural Language Parsing as Statistical Pattern Recognition*, Ph.D. thesis, University of Pennsylvania.

Michael MOORTGAT (1996), In situ binding: a modal analysis, in Paul DEKKER and Martin STOKHOF, editors, *Proceedings 10th Amsterdam Colloquium*, pp. 539–549, ILLC, Amsterdam.

Michael MOORTGAT (2011), Categorical type logics, in Johan VAN BENTHEM and Alice TER MEULEN, editors, *Handbook of Logic and Language*, chapter 2, pp. 95–179, North-Holland Elsevier, Amsterdam.

Michael MOORTGAT and Richard MOOT (2001), CGN to Grail: extracting a type-logical lexicon from the CGN annotation, *Language and Computers*, 37(1):126–143.

Richard MOOT (2010a), Automated extraction of type-logical supertags from the Spoken Dutch Corpus, in Srinivas BANGALORE and Aravind JOSHI, editors, *Complexity of Lexical Descriptions and its Relevance to Natural Language Processing: A Supertagging Approach*, chapter 12, pp. 291–312, MIT Press, Cambridge, Massachusetts.

Richard MOOT (2010b), Semi-automated extraction of a wide-coverage type-logical grammar for French, in *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, Montreal.

Richard MOOT (2012), Wide-coverage semantics for spatio-temporal reasoning, *Traitement Automatique des Langues*, 53(2):115–142.

Richard MOOT (2014), Extended Lambek calculi and first-order linear logic, in Claudia CASADIO, Bob COECKE, Michael MOORTGAT, and Philip SCOTT, editors, *Categories and Types in Logic, Language, and Physics: Essays dedicated to Jim Lambek on the Occasion of this 90th Birthday*, number 8222 in Lecture Notes in Artificial Intelligence, pp. 297–330, Springer, Heidelberg.

Richard MOOT and Mario PIAZZA (2001), Linguistic applications of first order multiplicative linear logic, *Journal of Logic, Language and Information*, 10(2):211–232.

- Richard MOOT and Christian RETORÉ (2006), Les indices pronominaux du français dans les grammaires catégorielles, *Linguisticae Investigationes*, 29(1):137–146.
- Richard MOOT and Christian RETORÉ (2012), *The Logic of Categorical Grammars: A Deductive Account of Natural Language Syntax and Semantics*, number 6850 in Lecture Notes in Artificial Intelligence, Springer, Heidelberg.
- Glyn MORRILL (1994), *Type Logical Grammar*, Kluwer Academic Publishers, Dordrecht.
- Glyn MORRILL (2011), *Categorical Grammar: Logical Syntax, Semantics, and Processing*, Oxford University Press, Oxford.
- Glyn MORRILL, Oriol VALENTÍN, and Mario FADDA (2011), The Displacement calculus, *Journal of Logic, Language and Information*, 20(1):1–48.
- Van Tien NGUYEN (2012), *Méthode d'Extraction d'Informations Géographiques à des fins d'Enrichissement d'une Ontologie de Domaine*, Ph.D. thesis, Université de Pau et des Pays de l'Adour.
- Richard T. OEHRLE (2011), Multi-modal type-logical grammar, in Robert BORSLEY and Kersti BÖRJARS, editors, *Non-transformational Syntax: Formal and Explicit Models of Grammar*, chapter 6, pp. 225–267, Wiley-Blackwell.
- Benoît SAGOT (2010), The Lefff, a freely available and large-coverage morphological and syntactic lexicon for French, in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta.
- Noémie-Fleur SANDILLON-REZER (2013), *Apprentissage de Grammaires Catégorielles: Transducteurs d'Arbres et Clustering pour Induction de Grammaires Catégorielles*, Ph.D. thesis, Bordeaux University.
- Natalie SCHLUTER and Josef VAN GENABITH (2008), Treebank-based acquisition of LFG parsing resources for French, in *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech.
- Stuart SHIEBER, Yves SCHABES, and Fernando PEREIRA (1995), Principles and implementation of deductive parsing, *Journal of Logic Programming*, 24(1–2):3–36.
- Willemijn VERMAAT (2005), *The Logic of Variation. A Cross-Linguistic Account of wh-question Formation*, Ph.D. thesis, Utrecht Institute of Linguistics OTS, Utrecht University.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*

<http://creativecommons.org/licenses/by/3.0/>



# FRIGRAM: a French Interaction Grammar

*Guy Perrier<sup>1</sup> and Bruno Guillaume<sup>2</sup>*

<sup>1</sup>LORIA, Université de Lorraine, Nancy, France

<sup>2</sup>LORIA, Inria Nancy Grand-Est

## ABSTRACT

We present FRIGRAM, a French grammar with a large coverage, written in the formalism of Interaction Grammars. The originality of the formalism lies in its system of polarities which expresses the resource sensitivity of natural languages and which is used to guide syntactic composition. We present the principles underlying grammar design, highlight its modular architecture and show that the lexicon used is independent of the grammar formalism. We also introduce the “companion property”, and show that it helps to enforce grammar consistency.

*Keywords:*  
*formal grammar,*  
*model theoretic*  
*syntax,*  
*polarity,*  
*Interaction*  
*Grammar.*

1

## INTRODUCTION

Following the seminal work initiated on Categorical Grammars (CG) by Lambek (1958), many other grammatical formalisms were proposed to describe the syntax of natural language. Apart from CG, the most well-known ones are TAG (Joshi *et al.* 1975), LFG (Bresnan 2001) and HPSG (Pollard and Sag 1994). These formalisms have several advantages. First, they allow for a detailed encoding of linguistic knowledge. Second, they can be used to investigate formal properties of natural language or to study linguistic hypotheses by testing them on real linguistic utterances. Third, grammars written using these formalisms can be used in more complete systems where syntax modeling is required either within a parsing or a generation application.

All these formalisms use a finite set of elementary structures and some mechanisms for composing these elementary structures to produce syntactic structures for larger utterances of a natural language. A large coverage system based on these approaches necessarily requires a huge number of elementary structures hence the development and the maintenance of such systems is a challenge and requires a lot of manual work. Among existing works to develop large coverage systems, mainly for English but also for some other languages, we can cite the ParGram (Butt *et al.* 2002) project (for LFG), the DELPHIN (Oepen *et al.* 2002) project (for HPSG) and XTAG (XTAG Research Group 2001) (for TAG).

We aim to conduct similar work within the framework of Interaction Grammars (IG), a formalism first introduced in Perrier (2000) and presented in more detail in Guillaume and Perrier (2009). IG combines a flexible view of grammars as constraint systems with the use of a polarity system to control syntactic composition. This polarity system expresses the saturation state of partial syntactic structures and their ability to combine.

The present paper reports on the construction of a syntactic resource for the French language and shows that it is possible to build a wide coverage IG grammar of French which encodes fine-grained linguistic knowledge.

In this grammar (called FRIGRAM), the syntax of French sentences is described by dependency structures which contain the usual surface syntactic dependencies but also by additional dependency relations which contain the information needed to produce a deeper syntactic analysis. These additional relations are: infinitival subjects, participial subjects and pronoun antecedents that are syntactically predictable.

The main challenge is to guarantee and maintain the consistency of the grammar while aiming for large coverage. To this end, we resort to the following means:

- a modular organization of the grammar in a hierarchy of classes which captures linguistic generalizations,
- well-formedness principles imposed on the elementary structures of the grammar,
- a strict separation between grammar and lexicon with a lexicon that is independent of the particular grammatical formalism used,



- the use of the *companion property* to help checking grammar consistency.

The paper is structured as follows. We start with a brief presentation of IG. We then go on to explain the different points just mentioned. We conclude with a comparison with other French grammars and discussion about the evaluation of the grammar.

## 2 INTERACTION GRAMMARS

IG is a grammatical formalism which describes the syntax of natural language using two notions: *tree description* and *polarity*. For a complete presentation of the formalism, the reader is referred to Guillaume and Perrier (2009).

### 2.1 Parse trees

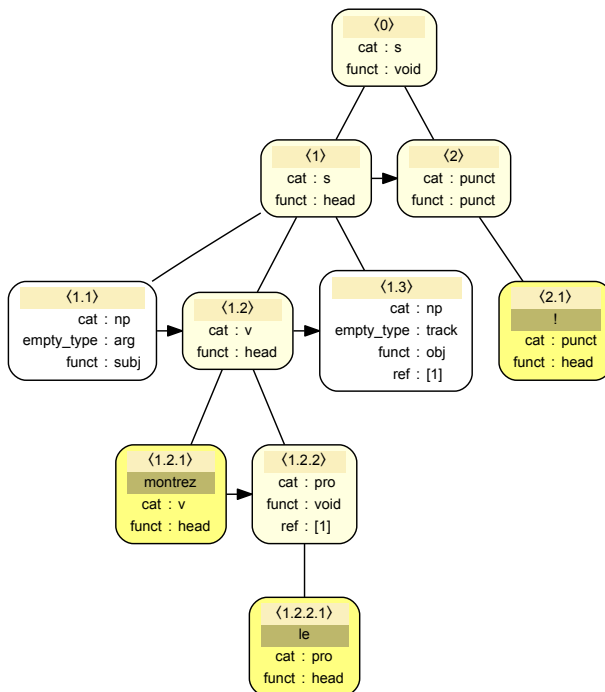
In IG, the syntax of sentences is modeled using constituency-based parse trees. An example of such a parse tree for the sentence *montrez-le!* ‘show it!’ is given in Figure 1.<sup>1</sup> Parse trees are totally ordered trees. Each node represents a constituent and carries a feature structure which encodes morpho-syntactic properties. All features in parse trees are written with the ‘:’ symbol like in [f:v] in contrast with features in tree description that are written with other symbols (see below). Leaves of parse trees can be either anchors carrying lexical units (nodes ⟨1.2.1⟩, ⟨1.2.2.1⟩, and ⟨2.1⟩) or empty nodes (⟨1.1⟩ and ⟨1.3⟩).

Empty nodes represent constituents that are not directly realized in the syntax. In FRIGRAM, empty nodes are used in several ways. The empty node ⟨1.1⟩ is an example of an argument which is not syntactically realized (feature [empty\_type:arg]). This is the case for infinitival and imperative subjects, as well as for some infinitival objects (tough movement). In Figure 1, the node ⟨1.1⟩ represents the non-expressed subject of the imperative verb *montrez*. The empty node ⟨1.3⟩ is an example of a trace of an argument which is moved from its canonical position, this trace is marked (feature [empty\_type:track]). In Figure 1, the node ⟨1.3⟩ is the trace of

---

<sup>1</sup>We suppose that the hyphen in *montrez-le!* is removed at the tokenization stage.

Figure 1:  
Parse tree  
representing  
the syntax of  
the sentence  
*montrez-le !*



the object clitic pronoun *le*;<sup>2</sup> the link between the pronoun and the extracted position is encoded by feature sharing: both constituents contain the feature [ref: [1]]. This mechanism is used in all cases of extraction, subject inversion, and cliticization of arguments.

## 2.2 Tree descriptions

To produce the parse trees described above, IG relies on a model theoretic syntax approach of natural languages (Pullum and Scholz 2001). Parse trees are defined as models of a more general notion of tree description (introduced by Rogers and Vijay-Shanker 1994). In this view, the basic objects of the grammar are not trees but properties that are used to describe them, in other words, *tree descriptions*. This approach is flexible and allows expressing elementary properties in independent ways and combining them freely. A tree description can be viewed ei-

<sup>2</sup>For consistency, all clitics are treated as moved arguments even if, as in the short sentence *montrez-le !*, the clitic *le* seems to be in a canonical place.

FRIGRAM: a French Interaction Grammar

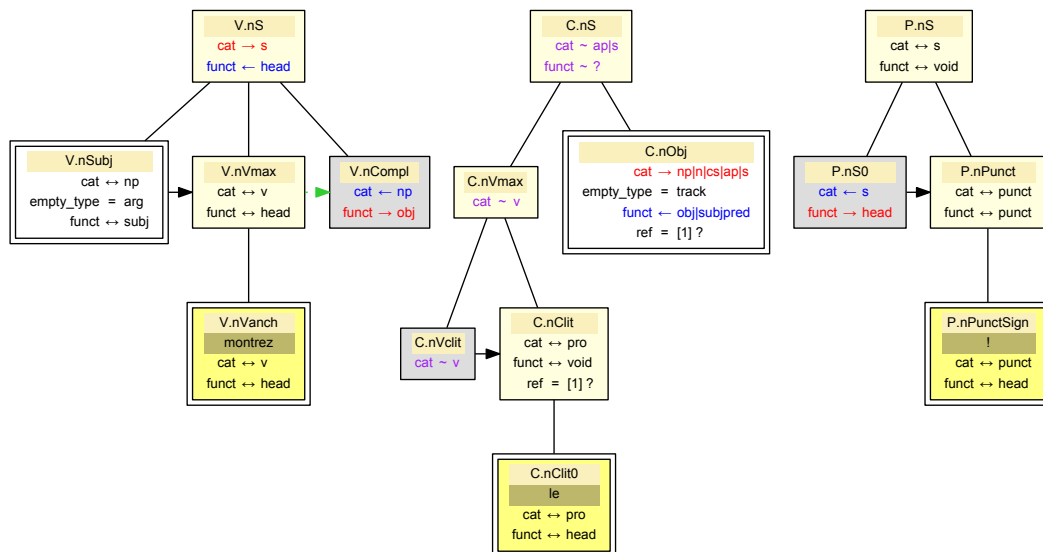


Figure 2: Polarized tree description associated with the sentence *montrez-le !* by the grammar FRIGRAM.

ther as an underspecified tree, or as the specification of a tree family, each tree being a model of this specification.

Figure 2 gives an example of a tree description<sup>3</sup> which is associated with the sentence *montrez-le !*. This description is composed of three connected components associated with the three lexical units *montrez*, *le*, and *!* occurring in that sentence.

Formally, a tree description is a finite set of nodes structured by two kinds of relations: *dominance* and *precedence*. Dominance relations can be immediate or underspecified. In the example, there are only immediate dominance relations represented with solid lines. Precedence relations can also be immediate or underspecified. They are represented with arrows; these arrows are solid and black (for immediate precedence) or dashed and green (for underspecified precedence).

Nodes are labeled with features describing their morpho-syntactic properties whereby a feature value is either an atom (like in a parse tree) or a disjunction of atoms. When a feature value is the disjunction of all elements of a domain, this value is denoted with "?".

<sup>3</sup>Note that the figures are simplified and only display a part of the full feature structures.

A co-indexation mechanism between feature values is also available at the description tree level (a common index  $[n]$  is put before their values). For instance, the *ref* feature of node *C.nObj* shares its value with the *ref* feature of node *C.nClit* meaning that both constituents refer to the same semantic entity.

### 2.3 Polarities

Polarities are used in tree descriptions to describe the saturation state of incomplete syntactic trees. The set of features is partitioned into two subsets: the set of *resource sensitive* features and the set of *neutral* features. Neutral features are written as  $[f = v]$ . For instance, agreement properties are expressed with neutral features.

Polarities are attached to resource sensitive features that label description nodes. Given a feature name  $f$  and a feature value  $v$  (which may be either an atomic value or a disjunction of atomic values), the four kinds of polarized features and their meanings are:

- a *positive* feature  $[f \rightarrow v]$  expresses an available resource which must be consumed;
- a *negative* feature  $[f \leftarrow v]$  expresses an expected resource which must be provided; it is the dual of a positive feature; one negative feature must match exactly one corresponding positive feature to be saturated and conversely;
- a *saturated* feature  $[f \leftrightarrow v]$  expresses a linguistic property that needs no combination to be saturated;
- a *virtual* feature  $[f \sim v]$  expresses a linguistic property that needs to be realized by combining with either a saturated feature or a pair of a negative and positive features.

In Figure 2, node *V.nCompl* carries a negative  $[cat \leftarrow np]$  and positive feature  $[funct \rightarrow obj]$ , which represents the expected object noun phrase for the transitive verb *montrez*.

The virtual features in the three nodes *C.nVclit*, *C.nVmax*, and *C.nS* of the second connected component in Figure 2 represent the syntactic context required by the clitic pronoun *le*, namely, a verb *C.nVclit* occurring immediately before the pronoun to build the node *C.nVmax* with it.

The descriptions labeled with polarized features are called *polarized tree descriptions (PTDs)* in the rest of the article.

An interaction grammar is defined by a finite set of PTDs, named Elementary PTDs (EPTDs) and generates a tree language. A parse tree (as defined in Section 2.1) belongs to the language if it is a model of a finite list of EPTDs in the sense given in Guillaume and Perrier (2009). Each node of the list of EPTDs is mapped to a node of the tree model through an interpretation function. When two nodes of EPTDs have the same image by the interpretation function, we say that they are *merged*. We also say that the features labeling these nodes are *merged*. Models can be saturated and/or minimal:

- A tree model is *saturated* if every positive feature  $[f \rightarrow v]$  is merged with a dual feature  $[f \leftarrow v]$  (and vice versa) and if every virtual feature is merged with either a saturated feature or a pair of a positive and a negative feature. Merging a positive (resp. negative) polarity with a saturated polarity or with another positive (resp. negative) polarity is forbidden. There is no saturation constraint on neutral features.
- A tree model is *minimal* if a minimum of information is added to the input EPTDs (no node, immediate dominance relation or feature that does not exist in the initial descriptions is added).

Parsing a sentence with a grammar  $G$  consists first of all in selecting an appropriate list of EPTDs from  $G$ . This selection step is easier if  $G$  is lexicalized. In that case, each EPTD has an anchor associated with a lexical unit of the language. This strongly reduces the search space for the EPTDs.

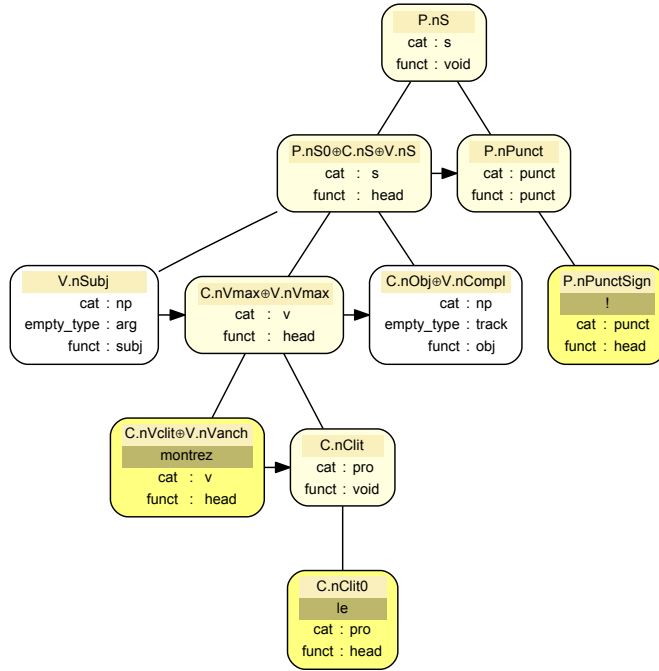
Then, the parsing process itself reduces to the resolution of a constraint system. It consists in building all models of the selected lists of EPTDs that respect the linear order of the words in the input sentence.

Figure 2 represented one of the possible selections of EPTDs from FRIGRAM for the sentence *montrez-le !*. The selection includes three EPTDs<sup>4</sup> which are considered as one single PTD with three connected components. Figure 3 shows the unique minimal and saturated model of the PTD. The parse tree of Figure 3 is the same as the one of Figure 1

---

<sup>4</sup>The three EPTDs are identified by a short name: in the order of their appearance in the sentence,  $V$  for the verb,  $C$  for the clitic pronoun,  $P$  for the punctuation. Nodes are then named with a prefixed notation like  $C.nS$ .

Figure 3:  
Tree model of  
the PTD shown  
in Figure 2  
representing  
the syntax of  
the sentence  
*montrez-le !*



but with information about the interpretation function used to build the model: the set of nodes of the PTD of Figure 2 that are interpreted in the given node of the model is given in the first line of each node. For instance, the anchor *montrez* is the interpretation of the two nodes *C.nVclit* and *V.nVanch* of the PTD of Figure 2.

It can be checked that all features in the model are saturated. For instance, the feature *cat* of the node *P.nS0/C.nS/V.nS* is saturated because the feature [*cat* → *s*] of node *V.nS* is merged with the negative feature [*cat* ← *s*] of node *P.nS0* and the virtual feature [*cat* ~ *ap* | *s*] of node *G.nS* is merged with the feature [*cat* → *s*] of node *V.nS*.

Other kinds on constraints are possible in PTDs. A node can be declared as *empty* (graphically represented with a white background, like nodes *V.nSubj* and *C.nObj* in Figure 2). It is then required that the image has an empty phonological projection, i.e., all leaves of the corresponding subtree are empty nodes. A node can be declared as *full* (graphically represented with a yellow background, like nodes *V.nS* and *C.nS* in Figure 2); it is then required that the image has a nonempty phonological projection: there is at least one anchor node

in the corresponding subtree in the model. Finally, a node can be declared to be *closed* (represented with a double rectangle, like nodes *C.nObj* and *P.PunctSign*) meaning that the set of its daughters is fixed. In the model, the node cannot have daughters that are not already present in the PTD.

### 2.5 Parsing as a process of node merging controlled by polarities

In an operational view of parsing, the building of a saturated and minimal model is performed step by step by refining the initial PTD with a merging operation between nodes, guided by one of the following constraints:

- neutralize a positive feature with a negative feature having the same name and carrying a value unifiable with the value of the positive feature;
- realize a virtual feature by combining it with a positive or saturated feature having the same name and carrying a value unifiable with the value of that virtual feature.

The constraints of the description interact with node merging to entail a partial superposition of their contexts represented by the tree fragments in which they occur. Let us illustrate this phenomenon with the parsing of the sentence *montrez-le !*, starting from PTD of Figure 2.

First, we try to saturate the virtual feature [*cat ~ v*] of the *C.nVclit* node by merging it with the *V.nVanch* node. The final target model is a tree and thus, every node has one mother node at most. As a consequence, in the PTD, merging propagates to the ancestors of the two nodes *C.nVclit* and *V.nVanch*: nodes *C.nVmax* and *V.nVmax* and again propagates to nodes *C.nS* and *V.nS*. Figure 4 shows the resulting PTD.

In a second step, nodes *V.nCompl* and *C.nObj* are merged to saturate their polarities [*cat ← np*] and [*cat → np | n | cs | ap | s*] respectively. Figure 5 shows the PTD resulting from this second merging.

In a last step, the root of the left tree and the *P.nSO* node of the right tree merge to build a unique tree in which all polarities are saturated. There remains a precedence relation which is not fully specified. Specifying it leads to the model shown in Figure 3.

Unlike in LFG or HPSG, in IG, feature structures are not recursive (feature values are atomic). This restriction is partly balanced by the

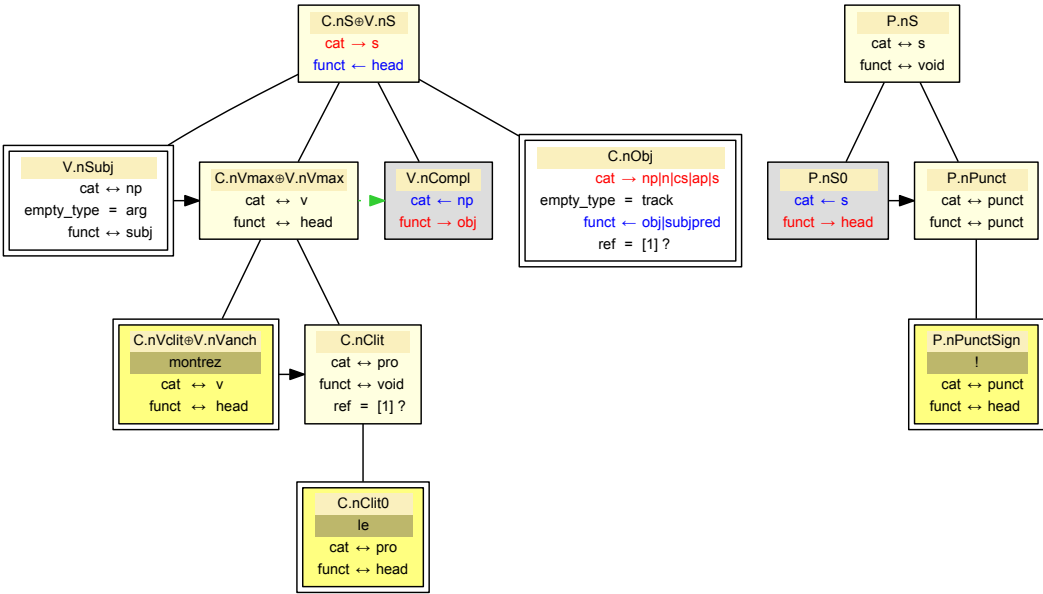


Figure 4: PTD after a first merging step in the parsing of the sentence *montrez-le !*

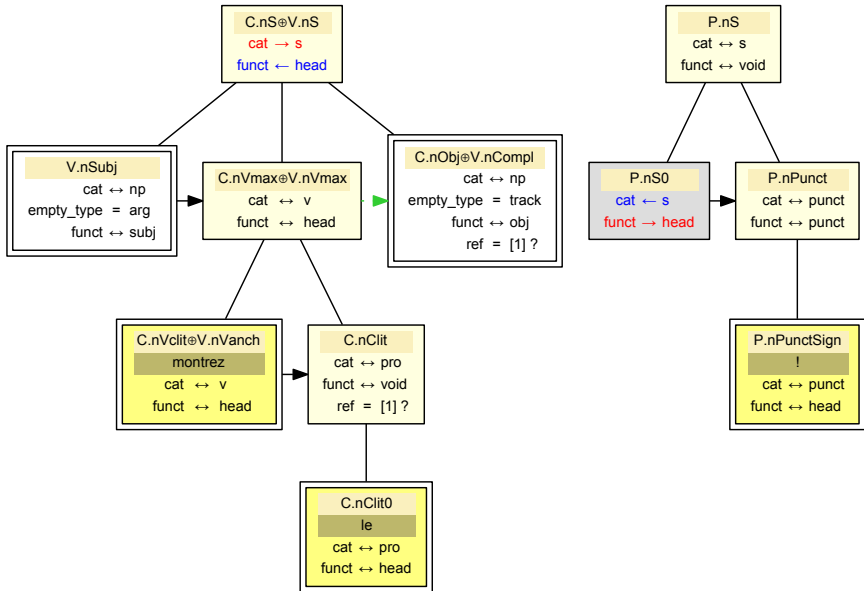


Figure 5: PTD after a second merging step in the parsing of the sentence *montrez-le !*



ability of IG to superpose tree structures. Using virtual polarities to enforce superposition, one can, for instance, impose restriction on the features of a subconstituent as is done by feature equations in LFG.

To summarize, IG combines the strong points of two families of formalisms: the flexibility of *Unification Grammars* and the saturation control of *Categorial Grammars*.

3

### THE PRINCIPLES OF THE GRAMMAR FRIGRAM

The formalism of IG is very general and does not impose any linguistic choice apart from the phrase structure tree representation of the syntax of sentences. When specifying a grammar, however, a number of linguistic decisions must be taken such as, for instance, how the notion of a constituent *head* should be defined and whether VP constituents should be used or not. In FRIGRAM parse trees, each node is a constituent linked to a leaf (either an anchor or an empty node) of its subtree called its *head*.

These choices are reflected in the EPTDs of the grammar through *principles*. In this section, we present the principles used in the design of FRIGRAM. These principles are not specific to the French language, however, and can be used or easily adapted to other languages. FRIGRAM, like other grammars with a large coverage, is a large resource and maintaining its consistency is a challenge. The principles facilitate the checking of this consistency. Principles are also heavily used in the conversion from constituent based parse trees to dependency trees discussed in the next section.

The first decision remains the *feature domain*: the choice of the set of feature names and of the possible feature values for each feature name. In FRIGRAM, *cat* is a resource sensitive feature used to encode the category of a constituent.

**Principle 1** (*cat*). *In an EPTD, every node has a cat feature.*

This feature is used to make a partition on the set of description nodes into two sets: *concrete* and *abstract* nodes.

**Definition 1.** *A node with a positive or saturated cat feature is called a concrete node. A node with a virtual or negative cat feature is called an abstract node.*

Another resource sensitive feature called *funct* is used to encode information about lexical heads and grammatical functions.

**Principle 2** (*funct*). *In an EPTD, every concrete node has a funct feature.*

For lexical head nodes, the special head value is used for feature *funct*. For other nodes, the value of the *funct* feature (*subj*, *obj*, ...) encodes the syntactic function of the constituent. There are exceptions (main sentences, moved constituents<sup>5</sup>) and for these exceptions, the *funct* feature carries the special value *void*.

**Corollary 1.** *From these first two principles, we can infer that:*

1. *each node in a parse tree has a cat feature;*
2. *each node in a parse tree is the image of exactly one concrete node of the starting PTD;*
3. *each node in a parse tree has a funct feature.*

*Proof.* Point 1 follows from the minimality of models: each model node is the image of at least one description node. Point 2 is a consequence of the polarity composition rule: a model node *N* is either the image of a node with a positive *cat* feature, a node with a negative *cat* feature, and any number of nodes with a virtual *cat* feature or the image of a node with a saturated *cat* feature and any number of nodes with a virtual *cat* feature; in the first (resp. second) case, the node with the positive (resp. saturated) *cat* feature is the only concrete node whose image is the given node *N*. Point 3 naturally follows from the previous principle and Principle 2. □

The third principle does not refer to linguistic properties but rather to a particular way of attaching linguistic phenomena to words.

**Principle 3** (strict lexicalization). *Every EPTD has exactly one anchor node. This anchor node has a saturated cat feature with an atomic feature value.*

The lexicalization of the grammar is mainly motivated by implementation aspects. Parsing with a lexicalized grammar is guided by

---

<sup>5</sup> In this case, there are traces of the moved constituents and these traces carry the *funct* feature expressing the syntactic function of the constituent.

the set of lexical units in the input sentences. In IG, as in the TAG formalism, the drawback is that each element of the final tree must be linked to one of the words of the sentence. However, in the case of IG, this is less problematic because of the superposition mechanism which makes it possible to freely split contributions to the final tree among several lexical units.

The last principle relies on the linguistic notions of head and projection which are required to define the concept of *spine*.

**Definition 2.** A spine in an EPTD is a list of nodes  $N_1, N_2, \dots, N_p$  such that:

- for all  $i$  such that  $1 < i \leq p$ , node  $N_i$  is a daughter node of  $N_{i-1}$ ;
- for all  $i$  such that  $1 < i \leq p$ , node  $N_i$  has a saturated feature *cat* and a feature [*funct*  $\leftrightarrow$  *head*];
- node  $N_1$  is a concrete node and its feature *funct* has a value different from *head*; it is called the maximal projection of all nodes belonging to the spine;
- node  $N_p$  is either an anchor or an empty leaf; in the first case, the spine is called a main spine; in the second case, it is called an empty spine; in both cases, node  $N_p$  is called the lexical head of all nodes belonging to the spine.

**Principle 4 (spine).** Every concrete node of an EPTD belongs to exactly one spine.

This principle means that every concrete node in an EPTD belongs to a continuous chain of concrete projections of a unique head leaf which may be empty.

**Corollary 2.** From the strict lexicalization principle and the spine principle we can deduce the following facts:

1. every EPTD has exactly one main spine;
2. every node  $N$  of a tree model has exactly one lexical head in this model, denoted  $\text{head}(N)$  and defined as follows: the concrete antecedent  $N_j$  of  $N$  in the initial PTD belongs to exactly one spine  $N_1, N_2, \dots, N_p$  and  $\text{head}(N)$  is the interpretation in the model of the leaf  $N_p$  ending that spine;
3. every node  $N$  in a tree model which is not a leaf has exactly one daughter node with the feature [*funct* : *head*] (recursively by fol-

lowing all nodes with feature [funct : head], we have an equivalent way of finding the lexical head  $head(N)$  of every node in the model);

4. each EPTD node with a positive feature *cat* is the maximal projection of some spine.

*Proof.* Points 1, 2, and 4 are obvious from definitions and principles. Point 3:  $N$  has at least one daughter with the feature [funct : head] because its concrete antecedent is on some spine and so it has a daughter node with [funct : head];  $N$  has at most one such daughter: by contradiction, suppose that there are two such daughters  $M$  and  $M'$ , then the concrete antecedents  $N_i$  of  $M$  and  $N'_j$  of  $M'$  are on two spines  $N_1, N_2, \dots, N_p$  and  $N'_1, N'_2, \dots, N'_q$ ; hence  $N$  is the image of  $N_{i-1}$  and  $N'_{j-1}$  which are two concrete nodes but two concrete nodes cannot be merged.  $\square$

To illustrate the concept of a spine, let us consider the EPTDs of Figure 2. The EPTD associated with the verb *montrez* has two spines: the main spine  $V.nS, V.nVmax, V.nVanch$  with its lexical head  $V.nVanch$ , and an empty spine reduced to a single node  $V.nSubj$ . In the EPTD associated with the clitic *le*, there are two spines: the main spine  $C.nClit, C.nClitO$ , and an empty spine reduced to node  $C.nObj$ .

#### 4 THE TRANSFORMATION OF PHRASE STRUCTURE SYNTAX INTO DEPENDENCY SYNTAX

The IG formalism is based on the constituency approach to syntax, as opposed to the dependency approach but the principles introduced in the previous section allow for an automatic transformation of IG constituency parses into dependency structures.

There is a large literature about this kind of transformation (see for example Choi and Palmer 2010). Some of the difficult aspects of this transformation include (i) deciding, for each node of the constituency tree, which of its daughters is the linguistic head and (ii) deciding, for each dependency edge, the grammatical function with which it should be labeled.

In previous versions of FRIGRAM, this transformation relied on the trace of the parsing construction (in IG these traces are the record of the polarities composition but this corresponds to the notion of

derivation tree in TAG). With the most recent version of FRIGRAM, it is not necessary to refer to the parsing construction and the two difficulties mentioned above were taken into account during the grammar building. Thanks to the principles described above, constituency trees contain systematic and precise information about heads of constituents and about their grammatical functions. In consequence, there is a canonical way to derive a dependency tree from the constituency tree.

The transformation is done in two steps. In the first step, all leaves of the model are items of the dependency structure and each node of the constituency tree with a feature `funct` with a value  $f$  different from `head` or `void` yields a dependency relation from the head of the mother of the node label with  $f$  to its own head. Applied to Figure 1, this gives:

- Node ⟨1.1⟩ produces a dependency relation labeled `subj` from the head of its mother ⟨1.2.1⟩ to leaf ⟨1.1⟩;
- Node ⟨1.3⟩ produces a dependency relation labeled `obj` from the head of its mother ⟨1.2.1⟩ to leaf ⟨1.3⟩;
- Node ⟨2⟩ produces a dependency relation labeled `punct` from the head of its mother ⟨1.2.1⟩ to leaf ⟨2.1⟩.

A relation `ANT` is used to keep track of the link encoded by the `ref` feature between an empty node and the extracted position.

In the second step, to produce more standard dependency structures devoid of  $\epsilon$ , empty words are removed and their incident dependencies are transferred to their full antecedent, when it exists. Figure 6 (right) shows the effect of empty node removal on the dependency structure obtained above (left).

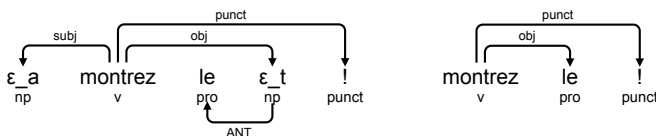


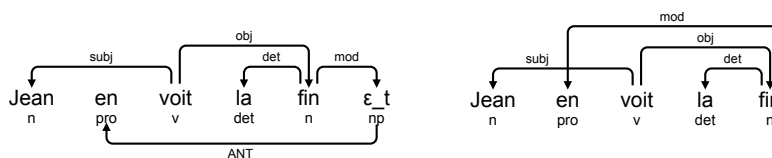
Figure 6: Dependency graphs representing the syntax of the sentence *montrez-le !*

The systematic usage of empty nodes to describe constructions where some constituents are extracted allows for a uniform conversion process. Even for some problematic cases that require non-projective dependency representation like sentence (1), a projective structure is

produced in the first step and the non-projective one is obtained with the second step (see Figure 7).

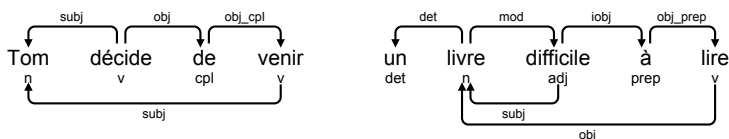
- (1) *Jean en voit la fin.*  
 Jean of it sees the end  
 ‘Jean sees the end of it.’

Figure 7:  
 Examples of  
 non-projective  
 dependency  
 structure



In our very simple example, the resulting dependency graph reduces to a tree but in more complex sentences, the dependency graph may contain nodes with several governors or even cycles. In Figure 8, two other examples are given; the first one is a DAG (*Tom décide de venir* ‘Tom decides to come’), the second one contains cycles (*un livre difficile à lire* ‘a book which is difficult to read’).

Figure 8:  
 Examples of  
 dependency  
 structures  
 produced by  
 FRIGRAM



As pointed out in Ivanova *et al.* (2012), different linguistic choice bring incompatible structures. It is the case here, and the structure given by FRIGRAM requires further transformation to match a corpus like Sequoia (Candito and Seddah 2012) for instance.

## 5 THE ARCHITECTURE OF THE GRAMMAR

A lexicalized grammar for French with a large coverage necessarily has an enormous size: the number of EPTDs in the grammar is the product of the number of entries in the lexicon of inflected words by the average of the number of ETPDs anchored by each inflected word. Now, a lot of these EPTDs are only different at the morphological and phonological level; to factorize these similarities, we use the notion of *EPTD template*. An EPTD template is an EPTD whose anchor is not attached to a particular word. A set of EPTD templates is called an *unanchored grammar*.

The lexicalized grammar is then produced as a combination of an unanchored grammar with a lexicon. Only the much smaller unanchored grammar is stored and the EPTDs are built on the fly during the parsing process. In our case, the unanchored grammar considered is called FRIGRAM.

Another interest of dissociating the lexicon from the grammar is that the lexicon may be written in a way that is totally independent of the grammatical formalism, so that it is reusable with grammars in other formalisms.

This independence of the lexicon with respect to the grammar is also used by FRMG for the formalism of TAG (Villemonte De La Clergerie 2010). It is an advantage with respect to systems in which the lexicon depends more or less strictly on the formalism used for writing the grammar:

- in LKB (Copestake and Flickinger 2000), the lexicon is totally integrated in the typed feature system of the grammar;
- in DotCCG (Baldrige *et al.* 2007), the dependency of the lexicon on the grammar is expressed through the notion of family; a lexical entry is associated with a family, which is a set of syntactic types having a linguistic unity;
- the same notion of family is used in XTAG (XTAG Research Group 2001) for TAG, but here, a family is a set of tree schemas.

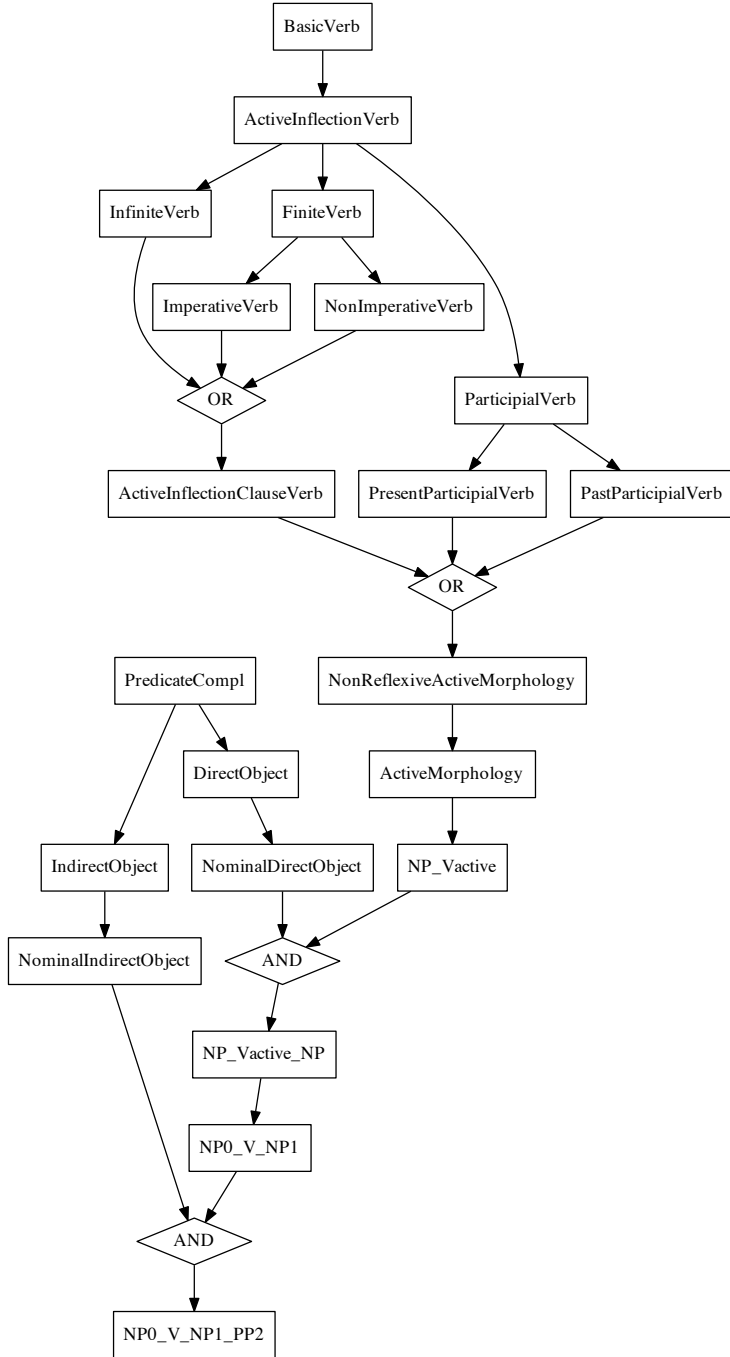
### 5.1 *The modular organisation of the grammar*

It is unthinkable to manually build a grammar with about 4,000 EPTD templates, considering each one individually. Even if it were possible, it would be intractable to maintain the consistency of such a grammar.

Now, the EPTD templates of FRIGRAM share a lot of fragments and it is possible to organize the grammar as a class hierarchy. The structuration of a grammar on the basis of a hierarchy is not new: HPSG uses a hierarchy of typed feature structures via an inheritance relation (Pollard and Sag 1994). Systems where built to help the development of HPSG-based grammars: LKB (Copestake and Flickinger 2000) or TRALE.

IG uses the more generic tool XMG (Crabbé *et al.* 2013). XMG can be used for various formalisms. It was already used for TAG (Crabbé

Figure 9:  
Hierarchy  
of classes used  
to define the  
NP0\_V\_NP1\_PP2  
class of transitive  
verbs with  
an indirect  
complement





2005) and IG. XMG provides a language to define a grammar as a set of classes. A class can be defined directly but also from other classes by means of two composition operations: *conjunction* (represented as diamond nodes labeled AND in Figure 9) and *disjunction* (represented as diamond nodes labeled OR).

Each class is structured according to several dimensions. FRIGRAM uses two dimensions: the first one is the syntactic dimension, where objects are EPTD templates, and the second one is the dimension of the interface with the lexicon, where objects are feature structures.

Defining the conjunction of two classes requires a specification of how the components are combined for each dimension: for the syntactic dimension, PTD union is performed; for the dimension of the interface with the lexicon, it is realized as unification between feature structures.

The terminal classes of the hierarchy define the EPTD templates of the grammar that are computed by the XMG compiler. Figure 9 gives the example of a terminal class, the NP0\_V\_NP1\_PP2 class of transitive verbs with an indirect complement, with the hierarchy of classes used to define it. The compiler accumulates the information given by all classes from the top to the bottom of the hierarchy, taking into account the two manners of composing classes. In this way, the compilation of the NP0\_V\_NP1\_PP2 class produces 40 EPTD templates.

For TAG, the compiler also produces tree descriptions, but these are not polarized, and after the compiler, a solver generates the elementary trees that are models of tree descriptions.

The *source grammar* is the set of all classes. In our case, the current source grammar FRIGRAM<sub>S</sub> is composed of 425 classes, including 175 terminal ones. The *object grammar* is the set of EPTD templates produced by the compilation of the terminal classes. In our case, the object grammar, FRIGRAM<sub>O</sub>, produced from FRIGRAM<sub>S</sub>, is composed of 3,890 EPTD templates.<sup>6</sup>

Of course, some general classes can be used in several different contexts. For instance, the classes related to complements of predicative structures are used as subclasses for the classes related to adjectives, nouns, and verbs requiring complements. For the sake of read-

---

<sup>6</sup>The grammar is systematically described in 280 pages of documentation (Perrier 2014).

ability, the set of classes is organized in a *module hierarchy*. Here is the list of all modules in the alphabetic order, with the number of classes by module between parentheses and a brief characterization of these classes:

- ADJECTIVE (16): adjectives,
- ADVERB (37): adverbs,
- COMPLEMENT (24): complements required by verbs, nouns, or adjectives,
- COMPLEMENTIZER (7): complementizers,<sup>7</sup>
- COORDINATION (12): coordination,
- DETERMINER (12): determiners, except interrogative determiners,
- EXTRACTGRAMWORD (18): extraction (from relative, interrogative, and cleft clauses),
- INTERROGATIVE (17): interrogative pronouns, adverbs, and determiners,
- NOUN (21): common and proper nouns,
- PREPOSITION (13): prepositions,
- PROCLITIC (26): clitic pronouns,
- PRONOUN (21): disjoint pronouns, except interrogative and relative pronouns,
- PUNCTUATION (24): punctuation marks,
- RELATIVE (11): relative pronouns,
- VERB (72): different families of verbs according to their subcategorization frame and specific verbs such as presentatives, modal and causative verbs,
- VERBIMPERSONALDIATHESES (25): different diatheses, active, passive and middle, with an impersonal subject,
- VERBKERNEL (28): classes defining the common verbal kernel of all verbs with the morphology and its interaction with the form of the subject, the syntactic function of the verb, and its voice,
- VERBPERSONALDIATHESES (40): different diatheses, active, passive, and middle, with a personal subject,

---

<sup>7</sup>The prepositions *à* and *de* introducing direct object infinitives are considered as complementizers, following Huot (1982).

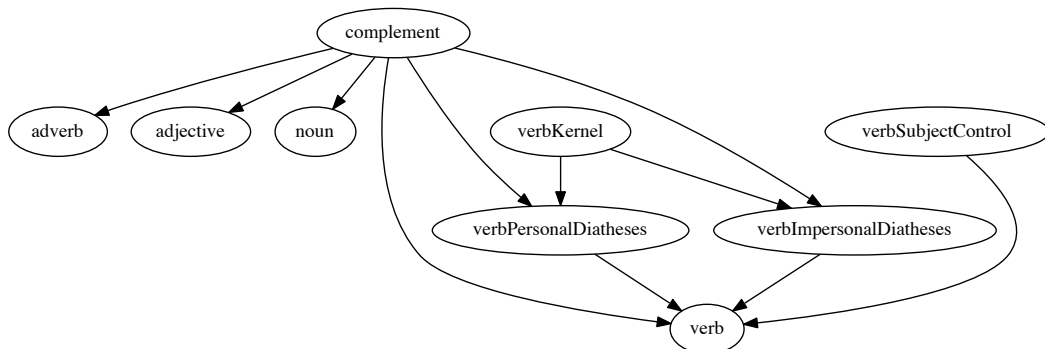


Figure 10: Hierarchy of modules grouping the classes of FRIGRAM<sub>S</sub> concerning verbs, nouns, adjectives, and adverbs

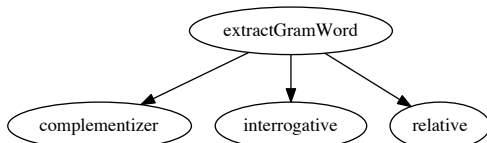


Figure 11: Hierarchy of modules grouping the classes of FRIGRAM<sub>S</sub> concerning extraction

- VERBSUBJECTCONTROL (1): control of subjects of infinitives by arguments of the verb governing the infinitive.

The number of classes by module is not proportional to the number of words the module pertains to but also depends on the number and the complexity of the phenomena anchored by these words. For instance, there are few clitic pronouns but they contribute to various syntactic constructions.

Some classes of one module are defined from classes of another module. We can represent this property with a graph where an edge means that some classes of the target module are defined from classes of the source module. Figure 10 shows these dependencies for the modules concerning verbs, nouns, and adjectives.

Figure 11 shows these dependencies for the modules modeling extraction from relative, interrogative, and cleft clauses. The modules

absent in Figure 10 and Figure 11 are isolated ones, without external dependencies.

A natural question arises: since XMG can be used for various formalisms, TAG and IG in particular, is it possible to re-use a grammar constructed in one formalism to build a new grammar in another formalism? The problem is that the organization of a grammar constructed with XMG is very close to the formalism, especially to the operation modeling syntactic composition. In TAG, this operation is adjunction, so that grammatical information tends to be anchored at verbs. In IG, this operation is merging of PTDs, which is more flexible, so that it is possible to anchor information at grammatical words.

### 5.2 *The link with a lexicon independent of the formalism*

The full grammar is produced from  $\text{FRIGRAM}_O$  and a lexicon. Each EPTD template from  $\text{FRIGRAM}_O$  is associated with a feature structure, called its *interface*, which describes a syntactic frame corresponding to lexical units able to anchor it. Lexicon entries are also described through feature structures. The set of features used in the interfaces differs from the one used in EPTDs because they do not play the same role: they do not aim at describing syntactic structures but are used for describing the morpho-syntactic properties of the words of the language anchoring the EPTDs in a way independent of the formalism.

Figure 12 shows an EPTD generated by the  $\text{NP0\_V\_NP1\_PP2}$  class with its interface above. The interface appears as a two-level feature structure. Features at the first level give the constituents of the frame associated with the EPTD. In our example, the head feature represents the verb. The *subj*, *obj*, and *iobj1* features respectively represent the subject, the direct object and the indirect object of the verb.

The second level describes the properties of each constituent of the frame. For instance, the value of the head feature is a feature structure describing some morpho-syntactic properties of the verb. Among the features present in this structure, [*impers*:maybe|never] says that the verb is either a verb that accepts a personal and impersonal construction of its subject or a verb with only a personal construction. The feature [*pronominal*:maybe|never] says that the verb may accept a reflexive object or it does not admit a reflexive clitic.

In the lexicon, the entries are pairs of an inflected word and feature structure. The feature structure has exactly the same format as

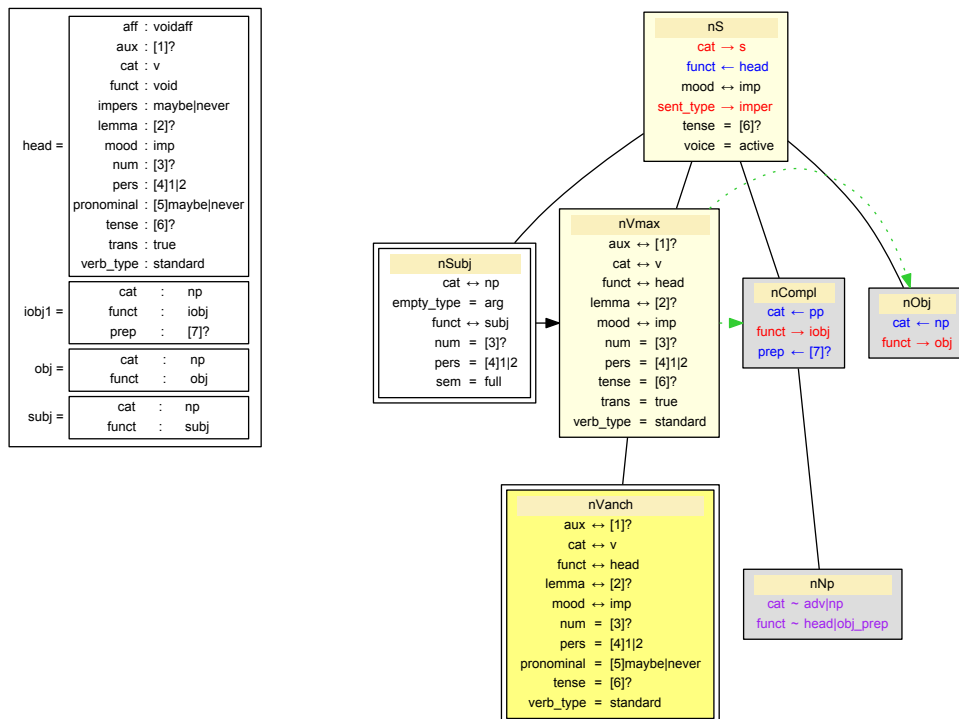


Figure 12: One of the EPTDs generated by the NP0\_V\_NP1\_PP2 class

the interfaces of the grammar. For instance, Figure 13 shows a lexical entry for the verb *montrez* used with a direct and indirect object, like in the sentence *montrez-le moi !* ‘show it to me!’.<sup>8</sup>

The anchoring of an EPTD template is performed by unifying its interface with the entries of the lexicon. At the first level, interfaces and lexical entries are viewed as closed structures: they unify if they have exactly the same set of features. At the second level, they are viewed as open structures: for a given feature of the first level, its value in the interface unifies with its value in the lexicon entry in the sense usually given to the operation of unification, which may entail the addition of second level features to the interface not present initially. For instance, the interface of the EPTD template presented in

<sup>8</sup>The question mark after the name of feature *iobj1* expresses that this feature is optional.

Figure 13:  
Entry of the verb *montrez*  
in the lexicon

head =	aff : voidaff aux : avoir cat : v impers : never lemma : «montrer» mood : implind num : pl passiv : total pers : 2 pronominal : maybe tense : pres trans : true
iobj1? =	cat : np prep : «dat»
obj =	cat : np
subj =	cat : np

Figure 12 succeeds in the unification with the lexical entry presented in Figure 13. At the first level, they have exactly the same set of features: head, iobj1, obj, and subj. At the second level, the values of the first level features unify in a standard way.

Since there is a co-indexation between features of the EPTD template and features of the interface, some feature values of the EPTD may be instantiated during anchoring. Figure 14 shows the anchored EPTD resulting from the unification between the interface of the EPTD template from Figure 12 and the lexical entry from Figure 13. As a side effect of anchoring the values of features aux, lemma, num, pers, tense, and prep have been instantiated.

The lexicon used to anchor FRIGRAM<sub>0</sub> (called FRILEX) combines morphological information extracted from ABU<sup>9</sup> and from Morphalou (Romary *et al.* 2004) with syntactic information for verbs extracted from Dicovalence (Van den Eynde and Mertens 2003). FRILEX contains 530,000 entries. To avoid size explosion, the required EPTDs of the grammar are built on the fly during parsing.

## 6 THE COMPANION PROPERTY AND THE CONSISTENCY OF THE GRAMMAR

Our ambition is to build a large coverage grammar for French syntax. Even if the hierarchical structure of the grammar makes it more compact and facilitates maintenance, the size of the grammar may be

<sup>9</sup><http://abu.cnam.fr/DICO/mots-communs.html>

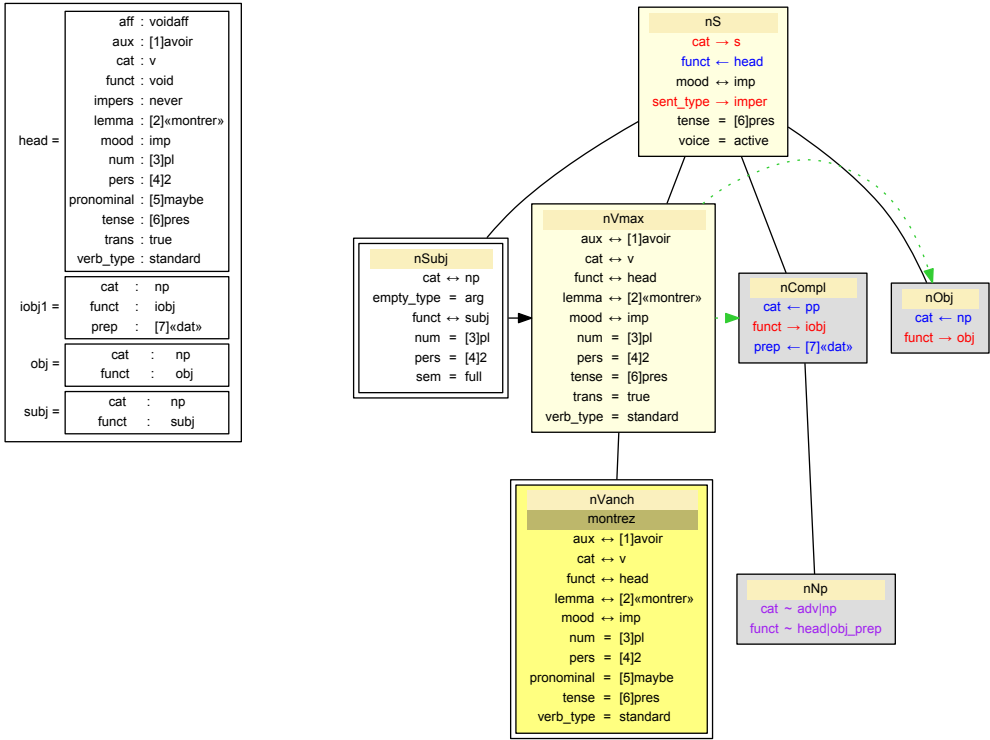


Figure 14: Anchored EPTD for the verb *montrez*

important and global consistency is difficult to maintain. The problem is that the different classes of the grammar source, even if they are not linked in the hierarchy, are generally not independent because EPTDs interact through the process of syntactic composition. For instance, in the parsing of the sentence *montrez-le !*, the EPTD anchored by the verb *montrez* interacts with the EPTD anchored by the clitic pronoun *le*, though the corresponding classes in the source grammar are totally independent. It is essential to check that this interaction works correctly.

A standard way to check global consistency of the grammar is to parse real sentences from test suites or corpora. However, the IG formalism provides another mechanism, complementary to the parsing of real sentences, to help checking the consistency of a grammar in a static way based on the EPTDs of the grammar without using

parsing. This mechanism uses the *Companion Property*. Originally, this property was introduced by Bonfante *et al.* (2009) to perform lexical disambiguation with IG.

Let us consider an interaction grammar.

**Definition 3.** *A companion of an unsaturated polarized feature in an EPTD of the grammar is another polarized feature of an EPTD such that the first feature is saturated by the second one: a merging of their respective nodes leads to a consistent PTD.*

A PTD is called *consistent* when it does not contain any unsatisfiable constraint; in other words, there exists some context in which the PTD can produce a model.

For instance, consider the EPTD associated with the verb *montrez* in Figure 2. A companion of the positive feature [funct → obj] is the negative feature [funct ← obj | subjpred] of the EPTD associated with the clitic pronoun *le*. The notion of companion can be expressed at the template level: if a template EPTD contains an unsaturated polarized feature, it requires to be saturated by some other complementary feature and we can search for companions (polarized features in a template EPTD) that are able to saturate it. Thanks to the medium size of FRIGRAM<sub>0</sub>, it is possible to exhaustively compute the set of companions of all polarized features of the EPTDs templates of grammar FRIGRAM<sub>0</sub>.

For instance, consider the EPTD template (noted  $E_0$ ) corresponding to the EPTD anchored with *montrez* in Figure 2. It contains a positive feature [funct → obj]; scanning FRIGRAM<sub>0</sub>, we find 107 companions. Moreover, for 95 companions (out of the 107), we know that they necessarily are companions coming from EPTD templates for which the anchor is on the right of the anchor of  $E_0$  after merging. The remaining 12 cases do not imply any order constraint on anchors.

As already said, companions are used for lexical disambiguation: when parsing a sentence, if an unsaturated feature of an EPTD  $E$  fails to find a companion in the EPTDs of the other lexical units of the sentence,  $E$  cannot be used in the parsing and so it can be removed. But companions can also be useful to help grammar development. The first diagnostic is when a polarized feature has an empty companion set: this means that the corresponding EPTD cannot be used in any grammatical parse; in this case this EPTD can be removed from the grammar



or there is some mistake in its definition. Other kinds of observations can be used by the expert: if a polarized feature has companions only on the left or on the right, it can be checked whether this corresponds to linguistic intuition. The full grammar contains 56,462 unsaturated polarities waiting for a companion: 21.2% have companions only on the right and 11.1% have companions only on the left.

During the FRIGRAM development, many inconsistencies were discovered using information about companion sets. In particular, some verbs requiring complements were found to have no companion because of the incompatibility between features or polarities of the expected complements and those of the EPTDs attached at the potential complements. It greatly helped to correct more or less deep defects and errors of FRIGRAM.

7

COVERAGE OF THE GRAMMAR

FRIGRAM covers a lot of syntactic phenomena: the different verb diatheses (active, passive, middle, causative, and impersonal diatheses), raising and control verbs, different types of sentences (declarative, interrogative, exclamative, imperative), extraction from interrogative, relative, cleft and dislocated clauses, noun complement clitic pronouns, quantifier pronouns, tough movement, coordination, and others. Within the limits of this article, it is not possible to describe all phenomena covered by FRIGRAM in an exhaustive way, but the reader can find a more complete information in the documentation of FRIGRAM (Perrier 2014).

In this section, we have chosen to present some of the most complex syntactic phenomena in French: causative constructions, negation, comparative, and consecutive constructions, extraction with pied-piping. Some of these have been little studied, especially comparative and consecutive constructions.

The modeling of these phenomena is constrained on the one hand by the formalism of IG and on the other hand by the rules of the French grammar. For the latter, our guide is the grammar of Riegel *et al.* (1999).

For every phenomenon, we compare the modeling in IG with the modeling in other formalisms, but most publications presenting such works are theoretical without any implementation and we know that

from a theoretical idea to its implementation in a grammar with a large coverage there is a long way which may be fraught with pitfalls (Bender 2008).

7.1 *Causative constructions*

In a causative construction, a causative verb (*faire* or *laisser*<sup>10</sup> in French) combines with an infinitive in the active voice. Here are examples illustrating this construction. For every sentence, the causative auxiliary and the complement infinitive are in bold.

- (2) *Jean le fait remplir.*  
Jean it makes fill  
'Jean makes someone fill it.'
- (3) *Jean fait se rencontrer les ingénieurs aujourd'hui.*  
Jean makes meet the engineers today  
'Jean makes the engineers meet today.'
- (4) *Jean s' est fait contrôler.*  
Jean himself has made control  
'Jean has been controlled.'
- (5) *Que Marie mange beaucoup la fait dormir.*  
That Marie eats a lot her makes sleep  
'That Marie eats a lot makes her sleep.'
- (6) *Jean fait prendre par Marie son billet de train.*  
Jean asks to take by Marie his ticket of train  
'Jean asks Marie to take his train ticket.'
- (7) *Jean fait balayer la cour à Marie.*  
Jean asks to sweep the yard to Marie  
'Jean asks Marie to sweep the yard.'

All these sentences are parsed with FRIGRAM. Sentence (2) illustrates clitic climbing in causative constructions: the clitic pronoun *le* is the direct object of *remplir* but it is attached at the causative verb *fait*.

Sentence (3) shows that clitic climbing is not performed if the clitic is a reflexive pronoun, *se* in the example that refers to the object *les ingénieurs* in the example.

However, in Sentence (4), the reflexive pronoun *se* refers to the subject of the causative verb *Jean*, while being the object of *contrôler*.

---

<sup>10</sup>The verb *laisser* can be only partially considered as a causative verb (Abeillé et al. 1997).

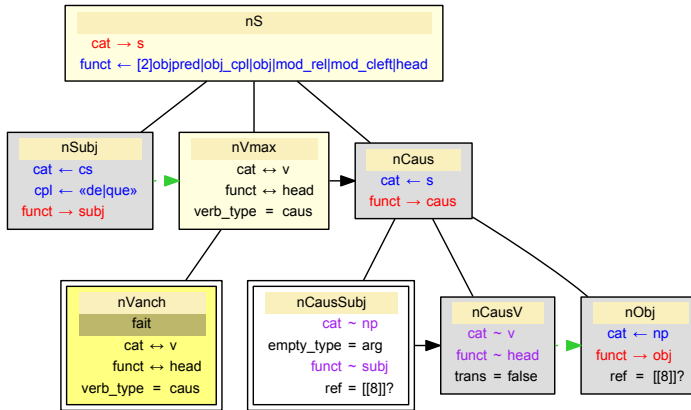


Figure 15:  
Anchored EPTD for the verb *fait* used as a causative verb with a specific object representing the subject of an intransitive caused verb

In this case, it climbs. Moreover, in this example, if the construction is a causative from a syntactic point of view, it is not the case from a semantic point of view, because *Jean* is not the causer of the action *contrôler*.

Sentence (5) shows a sentence where the causer is a complete clause: *Que Marie mange beaucoup*.

Sentences (6) and (7) illustrate the following rule: if the caused verb is transitive, its subject is expressed as an agent complement or indirect object.

In FRIGRAM, causative verbs are considered as special full verbs, as Figure 15 illustrates. The EPTD, shown in this figure, is attached at the verb *fait* used as a causative verb, with an intransitive caused verb. The caused verb is the head of an infinitive represented with the node *nCaus*. A specific complement, representing the subject of the caused verb introduced by the causative verb is considered a sub-constituent of the infinitive clause headed by the caused verb. In the EPTD, it is represented by the node *nObj* with the function *object*. A co-indexed feature *ref* indicates that it refers to the same entity as the empty subject of the infinitive represented by the node *nCausSubj*.

The question arises: why have we chosen to put this node *nObj* as a sub-constituent of *nCaus* and not of *nS*? The reason is that it is possible to insert specific complements linked to the causative construction between the caused verb and its own complements. Sentence (6) illustrates this problem: the agent complement *par Marie*, depending on the causative construction, comes between the caused verb *prendre* and its

object *son billet de train*. As a consequence, it must be put in the same constituent.

An alternative way of modeling causative constructions would be to consider causative verbs as auxiliaries, like tense or passive auxiliaries. It would require adding a specific entry in the grammar for all infinitives likely to take a causative auxiliary, which would increase the size of the grammar and lexical ambiguity in parsing. Another drawback comes from the flat representation that this entails. Let us consider Sentence (3). If we consider *fait* as a usual auxiliary of the verb *rencontrer*, node *nS* representing the whole sentence will have two daughter nodes as sub-constituents with the same function object: the own object of *rencontrer*, the clitic pronoun *se*, represented by its trace, and a specific object, *les ingénieurs*, introduced by the causative construction. Thus, if we want to attach every object at its verb, we need additional information to know which object is attached at which verb. We have the same problems for dative complements: in a flat structure, we can have two dative complements at the same level, one that is attached at the causative verb and another one that is attached at the infinitive.

Now, how to model constraints on clitic climbing, as they are expressed in the previous examples? In order to limit the ambiguity of the grammar, they are not attached at the causative verb but at the clitic pronouns. Figure 16 shows an EPTD for a reflexive clitic that is an object of the caused verb and that refers to the subject of the causative verb. Such a clitic must climb to the causative verb, as Sentence(4) illustrates. A node *nConst* represents the trace of this object at the canonical object position in the clause *nSO* headed by the caused verb, *contrôler* in our example. The clitic *se*, represented by the node *nClit*, is attached at the auxiliary *est* of the causative verb *fait*, represented by the node *nVclit*. Finally, the feature [*ref* = [[8]] ?] indicates a co-reference with the subject *Jean*. There is another lexical entry for *se* when it does not climb in a causative construction, as Sentence (3) illustrates. For non-reflexive clitic pronouns, there is only one lexical entry which forces climbing.

There are in-depth studies of French causative constructions in two other formalisms, HPSG and LFG. In all studies, the discussion is about the choice between the flat and the biclausal representation, which relates to a linguistic choice: to see a causative verb as any

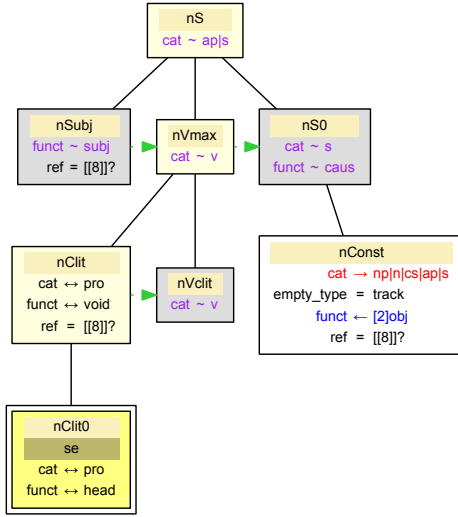


Figure 16:  
Anchored EPTD for the clitic pronoun *se* used as an object of a caused verb referring to the subject of the causative verb

control verb or to see it as constituting a complex verb with its object infinitive.

Abeillé *et al.* (1997) do not choose any of the options but dedicate a specific representation to each case: the biclausal representation is used when the causee is an object clitic and when there is no clitic climbing; the flat representation is used in other cases. The two cases are not completely disjunct, so that the two representations can be used when the caused verb is intransitive and the causee is an object clitic as in Sentence (5), whereas FRIGRAM provides a unique parse in this case. On the other hand, Abeillé *et al.* (1997) accept very rare constructions which are rejected by FRIGRAM. Here are examples illustrating these cases.

- (8) *Jean le fait remplir la citerne.*  
Jean him makes fill the cistern  
'Jean makes him fill the cistern.'
- (9) *Jean le fait lui téléphoner.*  
Jean him makes him call  
'Jean makes him call him.'

In Sentence (8), the causee is an object clitic, though the caused verb has a direct object. In Sentence (9), the clitic *lui* has not climbed. It is

possible to take these cases into account in FRIGRAM, but the interest is not obvious, because of the rarity of their occurrences.<sup>11</sup>

Both in FRIGRAM and in the proposal of Abeillé *et al.* (1997), not all the constraints on the cliticization of the arguments for the causative verb and infinitive are taken into account. For instance, the following ungrammatical sentence, taken from Yates (2002), is accepted.

- (10) \**Pierre lui a fait téléphoner Marie*  
Pierre him made call Marie  
'Pierre made Marie call him.'

Yates (2002) uses the potentiality of LFG to represent the complex constraints on cliticization in causative constructions illustrated by Sentence (10). At the constituent level, the c-structure expresses a flat representation, whereas at the functional level, the f-structure allows a sharing between arguments of the infinitive and the causative verb. He shows that his ideas can be transposed in HPSG.

To transpose them in IG would entail substantial changes in the grammar. Putting the caused verb and its complement in a flat structure at the same level as the causative verb requires a specific entry in the grammar for the caused verb. The usual entries for infinitives no longer work because the mood of the clause is given by the causative verb. This addition of a new entry must be repeated for all subcategorization frames of infinitives at the active voice. This would increase the size and ambiguity of the grammar.

## 7.2

### *Negation*

In French, negation is most often expressed with the clitic *ne* paired with a negative grammatical word which can be an adverb (*pas, guère...*), a pronoun (*personne, nul...*), or a determiner (*aucun...*). To name all these words in a unique manner, we use a non-standard term, *negative satellites*, which expresses that the words must be paired with *ne*. The following examples illustrate different cases of negative satellites. The clitic *ne* and its satellites are in bold.

- (11) *Jean ne mange pas de pommes.*  
Jean eats not apples  
'Jean does not eat apples.'

---

<sup>11</sup> No occurrence of these constructions exists either in Sequoia or in the FTB.

- (12) *Marie ne pense connaître la femme d' aucun ingénieur.*  
 Marie think to know the wife of any engineer  
 'Marie thinks to know the wife of no engineer.'
- (13) *Jean ne travaille avec l' appui de personne.*  
 Jean works with the support of nobody  
 'Jean works with the support of nobody.'
- (14) *Jean ne pense pouvoir travailler que dans sa chambre.*  
 Jean thinks to be able to work only in his room  
 'Jean thinks to be able to work only in his room.'

The pairing of *ne* with one negative satellite is expressed in FRIGRAM with a polarized feature *neg* which is attached at the clause constituting the scope of the negation. Particle *ne* provides the positive feature [*neg* → *true*] to neutralize the dual negative feature [*neg* ← *true*] given by the negative satellite.

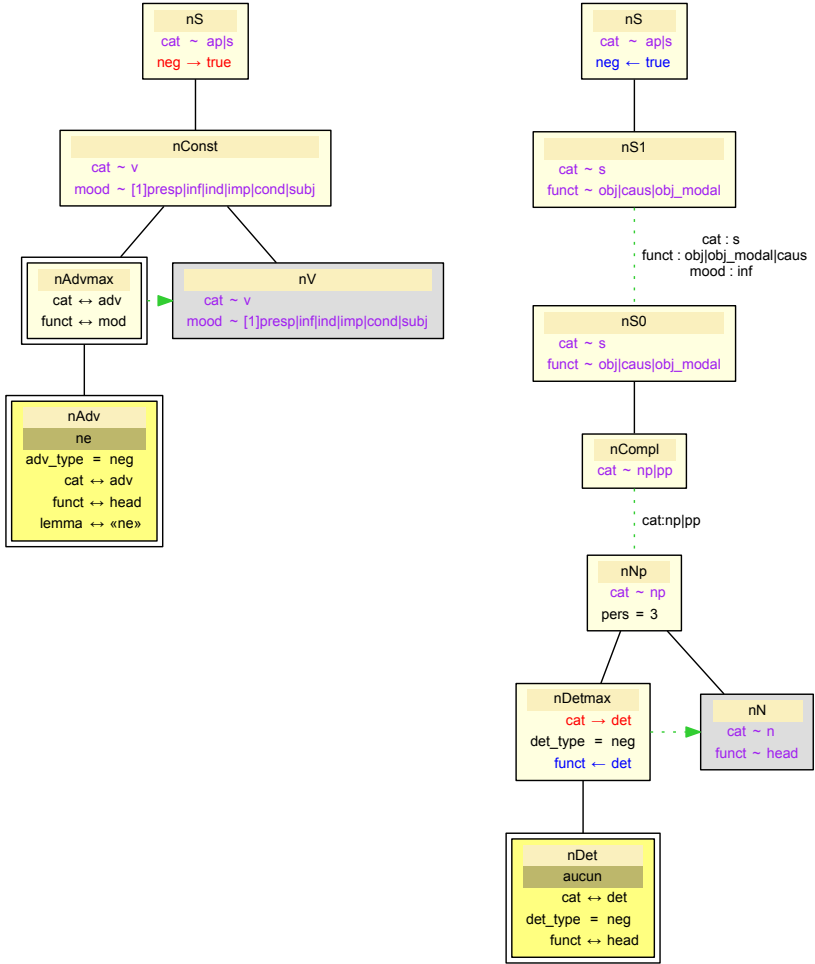
Figure 17 shows the EPTD attached at the clitic *ne* on its left. The node *nS* represents the sentence that is the scope of the negation. It carries the positive feature [*neg* → *true*]. The particle *ne* appears as a clitic put before the verb represented by the node *nV*. The maximal projection of the clitic represented by the node *nAdvmax* cannot receive any modifiers, which is indicated by the fact that the node is closed (double rectangle on the figure).

A difficulty comes from the fact that the negative satellite paired with *ne* can be situated in a constituent that is embedded more or less deeply in the clause that is the scope of the negation. This property applies only to some satellites. In particular, it is true for *aucun* and *personne*, as Sentences (12) and (13) show it. The satellites are situated in noun phrases: *aucun ingénieur* in Sentence (12) and *personne* in Sentence (13). If these noun phrases are noun complements, as in the examples, they can be embedded more or less deeply in a clause and, moreover, this clause can itself be embedded in the clause that is the scope of the negation.

That is the case for Sentence (12): from the satellite word *aucun* to the scope of the negation, there is chain of constituents: a noun phrase *aucun ingénieur*, a prepositional phrase *d'aucun ingénieur*, another noun phrase *la femme d'aucun ingénieur*, and an infinitive *connaître la femme d'aucun ingénieur*.

Figure 17 (on the right) shows the EPTD anchored by *aucun* and used in the parsing of Sentence (12). In this EPTD, the node *nS* repre-

Figure 17:  
EPTD attached  
at the clitic *ne*  
and an EPTD  
attached at the  
negative satellite  
*aucun*



sents the clause that is the scope of the negation. It is labeled with the negative feature [neg ← true].

The node *nS1* represents the immediate sub-constituent of *nS* that is the clause including *aucun*. The clause *nS0* that is the location of *aucun* can be embedded more or less deeply in *nS1*, which is expressed with an underspecified dominance relation from *nS1* to *nS0*. In Sentence (12), the nodes *nS0* and *nS1* are merged and they represent the clause *connaître la femme d'aucun ingénieur*.



The node *nCompl* represents the complement that is a sub-constituent of *nSO* and that is the location of *aucun*, *la femme d'aucun ingénieur* in the example. The node *nNp* represents the noun phrase that is the location of *aucun*, *aucun ingénieur* in the example. Another underspecified dominance relation from *nCompl* to *nNp* expresses the possibility for *nNp* to be embedded more or less deeply in *nCompl*.

Now, even if the location of a satellite word like *aucun* is relatively free, it is not completely free. Here are examples of unacceptable sentences in French.

- (15)\**Jean ne vient parce qu' il connaît aucun invité.*  
 Jean is coming because he knows no guest  
 'Jean does not comes because he doesn't knows any guest.' (intended)
- (16)\**Marie ne pense connaître une fille qui ait aucun défaut.*  
 Marie thinks to know a girl who has no defect  
 'Marie does not think that she knows a girl without any default.' (intended)

Sentence (15) illustrates the fact that *aucun* cannot be located inside an adjunct clause depending on the clause including the particle *ne*. In IG, such a constraint is expressed on underspecified dominance relations with feature structures. In the EPTD on the right of Figure 17, the following feature structure labels the underspecified dominance relation from *nS1* to *nSO*: {[cat : s], [funct : obj | obj\_modal | caus], [mood : inf]}. This feature structure means that each node of the model that is between the nodes *nS1* and *nSO* must be labeled with [cat : s], [mood : inf], and [funct : obj] or [funct : obj\_modal] or [funct : caus]. In other words, the clause where *aucun* is located must be embedded in a sequence of infinitive clauses that are object of transitive verbs, causative verbs, or modal auxiliaries.

Sentence (16) illustrates a constraint on another underspecified dominance relation in the EPTD of Figure 17 on the right: from *nCompl* to *nNp*. This relation is constrained by the feature structure {[cat : np | pp]}. Such a constraint means that between *nCompl* and *nNp* there may be only a sequence of noun or prepositional phrases. This entails the failure of parsing Sentence (16). In this sentence, from *nNp* to *nCompl* there is the following sequence of constituents: a noun phrase *aucun défaut*, a relative clause, *qui ait aucun défaut*, and another noun phrase, *une femme qui ait aucun défaut*. The second constituent of this sequence violates the constraint on the underspecified dominance relation from *nCompl* to *nNp*.

Sentences (13) and (14) show that *personne* and *que* behave in a similar manner as *aucun*, which is modeled in FRIGRAM with a similar EPTD. For *que*, there is a difference: there is only one underspecified dominance relation, between *nS1* and *nS0*.

Another difficulty comes from the fact that one clitic *ne* can be paired with several negative satellites, as Sentence (17) shows. In this sentence, there are two satellite adverbs: *pas* and *que*.

- (17) Jean **ne**            *mange pas que* *des pommes*.  
      Jean does not eat            only            apples  
      ‘Jean does not eat only apples.’

In the IG formalism, one positive feature [ $\text{neg} \rightarrow \text{true}$ ] must be saturated by exactly one negative feature [ $\text{neg} \leftarrow \text{true}$ ]. Our solution is to distinguish between the main and secondary satellite word. The main satellite word brings the negative feature [ $\text{neg} \leftarrow \text{true}$ ] and the secondary satellite word brings a virtual feature [ $\text{neg} \sim \text{true}$ ].

All negative satellites can play the role of main satellites. How should the negative satellites that can also play the role of secondary satellites be determined? We have chosen a pragmatic criterion: if a negative satellite can occur simultaneously with a lot of other negative satellites, we consider it a potential secondary satellite. It is not completely satisfactory because it ignores some cases. For instance, *que* can co-occur with *rien*, *guère*, *jamais*, *pas*, *plus*, whereas *pas* can only co-occur with *que*. As a consequence, *que* is considered a secondary satellite, while *pas* is considered only a main satellite.

Regarding the specific studies about the formalization of the negation syntax in French, we propose to start the comparison with work that is related to the study of the phenomenon from a strictly syntactic point of view. Following Abeillé and Godard (1997), Kim and Sag (2002) propose to model French and English negative adverbs in the framework of HPSG. For French, they restrict the study to the adverbs behaving as *pas*, that is with a very constrained position: *guère*, *jamais*, *plus*... They do not consider *que* which has a more free position.

Their work focuses on the possible positions of the negative adverb in the constituent tree of the sentence, according to the mood of the head verb. They conclude that with non-finite verbs it is a preverbal VP-adjunct, and with finite verbs it is taken as a postverbal complement sister of the other complements in the VP. *Ne pas* put be-

fore an infinitive is considered a VP-modifier. *Ne* put before a finite verb is considered an affix and a lexical rule transforms the lexical entry of the finite verb into a negated entry where a slot for the negative adverb is added to the subcategorization frame of the verb. In this way, the number of negation adverbs that it is possible to put after a finite verb is controlled by the subcategorization frame.

In IG, the notion of a verb phrase does not exist, so that negative adverbs are considered verb modifiers. The constraints about the linear order between the verb and negative adverb according to the mood of the verb are expressed with two different EPTDs corresponding to the two positions. For the preverbal position, the modeling of the negative adverb as a verb modifier entails a small limitation: it is not possible to reflect wide scope of *ne pas* over a conjunction of coordination, as in the following example extracted from Abeillé and Godard (1997).

- (18) *Paul promettait de ne pas lire le journal ou regarder la télévision*  
 Paul promised not to read the newspaper or watch television.

‘Paul promised not to read the newspaper or watch television.’

To take this phenomenon into account, it is sufficient to add a unique EPTD anchored by *ne pas* which has the function of infinitive modifier besides the EPTDs used for *ne* and *pas* with other moods.

In the restricted context chosen by Kim and Sag (2002), the dependency between the clitic *ne* and the negative satellite is relatively simple. Godard (2004) studies this dependency in depth in a general context, where it is more complex. Moreover, she also considers the semantic dimension: the satellite is considered as a quantifier and the clitic *ne* marks the scope of this quantifier. She proposes a model of the dependency in the framework of HPSG. This dependency is distant. Whereas IG uses its system of polarities combined with the relation of underspecified dominance to express distant dependencies, HPSG uses the propagation of features in the syntactic tree from node to node.

### 7.3 *Comparative and consecutive constructions*

Some adverbs, while acting as modifiers, are correlated with conjunctions or prepositions introducing a clause in a comparative or consec-

utive construction, as the following examples show. The adverbs and their correlated grammatical words are in bold.

- (19) *Il connaît les parents de **trop** d'élèves **pour** ne pas venir.*  
He knows the parents of too many students to not come  
'He knows the parents of too many students to not come.'
- (20) *Jean a **tellement** travaillé **qu'**il peut se reposer.*  
Jean has so much worked that he may have a rest  
'Jean has worked so much that he may have a rest.'
- (21) *Le paysage est **plus** ensoleillé **qu'** il ne l' est en hiver.*  
The landscape is more sunny than it is in winter  
'The landscape is more sunny than it is in winter.'
- (22) *Le paysage est **plus** ensoleillé maintenant **qu'** en hiver.*  
The landscape is more sunny now than in winter  
'The landscape is more sunny now than in winter.'

The first two examples illustrate the consecutive construction and the last two illustrate the comparative construction. Like negation, the two constructions use a correlation between two distant grammatical words. This correlation was analyzed from a linguistic point of view either in general French grammars (Grevisse and Goosse 2008; Riegel *et al.* 1999), or in specific studies for the comparative construction (Fuchs *et al.* 2008), but, to our knowledge, there is no specific study of their modeling in grammatical formalisms.

As for negation, the modeling in IG of the correlation between the adverb and conjunction in both constructions uses the system of polarities. We propose to develop on how the comparative construction of Sentence (22) is modeled within FRIGRAM.

Figure 18 represents the EPTDs used for *plus* and *que* in the parsing of Sentence (22). The correlation between the two words is expressed with polarized features, here the features *cat*, *funct*, and *sent\_type* of the nodes *nCompl0* and *nCs*.

In the EPTD of *plus*, the node *nConst* represents the word after its modification by *plus*, *plus ensoleillé* in our example. The node *nC* represents the expression that is the scope of the construction; in our example, it is the adjectival phrase *plus ensoleillé maintenant qu'en hiver*.

In the EPTD of *que*, the node *nCs* represents the clause complemented by *que* which is an argument of the adverb triggering the comparison. The node *nS* represents the clause without its complementizer. In our example, as in most cases, the clause includes an ellipsis.

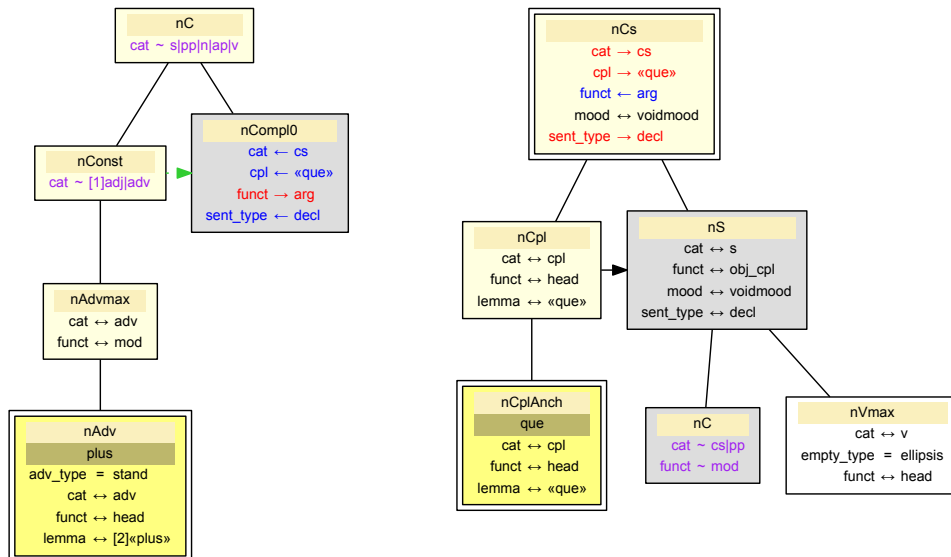


Figure 18: EPTDs anchored by *plus* and *que* for the parsing of Sentence (22)

It reduces to the prepositional phrase *en hiver* and the complete clause would be *il ne l'est en hiver*. The elided verb is represented by the node *nVmax* and the node *nC* represents the complement *en hiver*.

The modeling is similar for the other examples, except for Sentence (19), where *trop d'élèves* is embedded in a constituent which is not at the same level as the clause expressing the consequence *pour ne pas venir*. In FRIGRAM, it is expressed in the EPTD of *trop* with an underspecified dominance relation from the constituent representing the scope of the construction, the whole sentence in our example, to the noun phrase determined by *trop de, trop d'élèves* (see Figure 19).

#### 7.4 Extraction with pied-piping

All relative, cleft, and interrogative clauses with partial interrogation give rise to extraction of constituents. These constituents are put at the beginning of the clause from which they are extracted, and in our approach an empty constituent remains at the initial place as a trace.

Extraction is one of the syntactic phenomena in French that are the most difficult to formalize because it interacts with other phenomena, especially subject inversion and pied-piping. In the limits of this article we propose to focus on pied-piping.

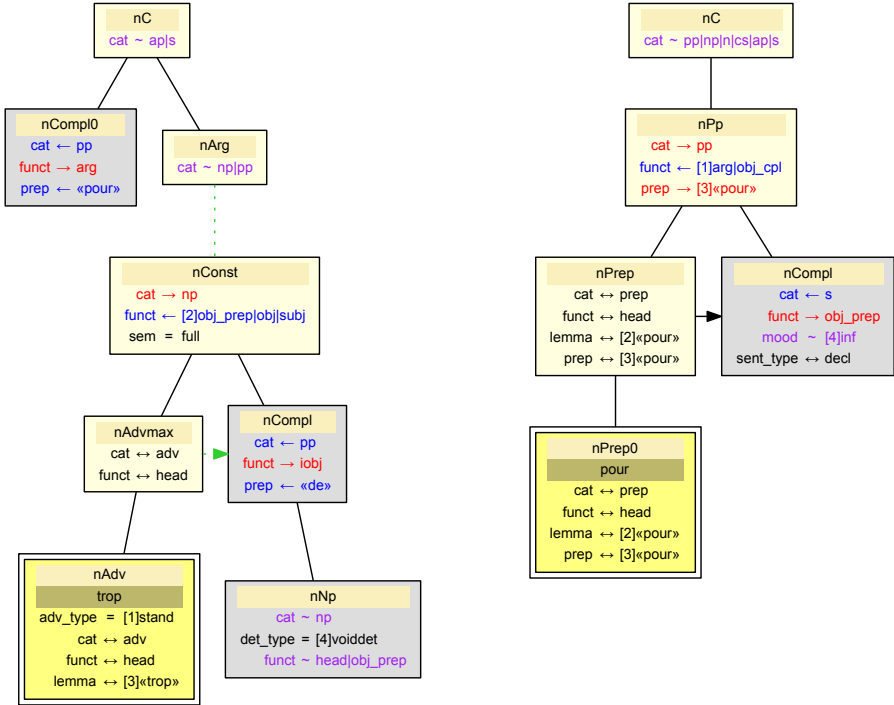


Figure 19: EPTDs anchored by *trop* and *pour* for the parsing of Sentence (19)

The three sentences below illustrate the phenomenon. In each example, the extracted constituent is put in square brackets and the grammatical word triggering extraction, called *wh*-word in the following, is displayed in bold type. The trace of the extracted constituent is marked with the symbol □.

(23) *Marie connaît Jean [dans l' entreprise de **qui**] elle pense*  
 Mary knows John in the company of whom she believes  
*travailler bientôt* □.  
 to work soon

‘Mary knows John, in whose company she believes to work soon.’

(24) *[Au patron de **quelle** entreprise] Sue veut -elle parler* □ ?  
 to the boss of which company Sue wants to-speak  
 ‘Which company does Sue want to speak to the director of?’

(25) *[Au directeur de **laquelle**] Marie veut -elle parler* □ ?  
 to the director of which-one Mary wants to-speak  
 ‘Which one does Mary want to speak to the director of?’

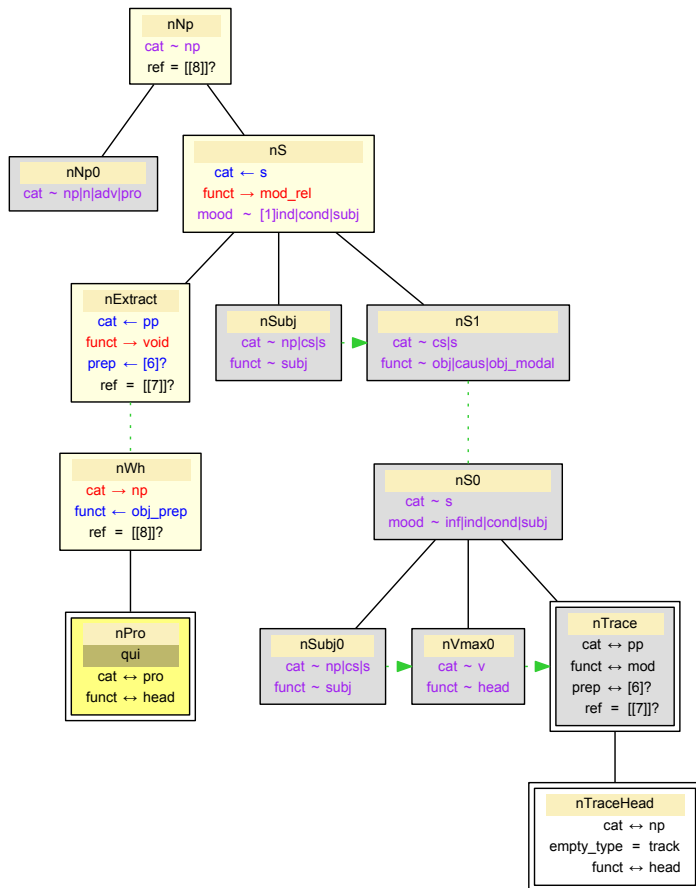


Figure 20: EPTD anchored by the relative pronoun *qui* and used in the parsing of Sentence (23)

Extraction gives rise to an unbounded dependency but in the case of pied-piping, a second unbounded dependency is introduced. Only the pronouns *qui*, *quoi* and *lequel* and the determiner *quel* allow pied-piping. Pied-piping means that the extracted constituent does not necessarily identify with the *wh*-word but it may concern a larger expression including this word.

In Sentence (23), the extracted constituent is *dans l'entreprise de qui* and the extraction is triggered by the relative pronoun *qui*. In this case, the relative pronoun represents a noun phrase more or less deeply embedded in the extracted constituent. Hence, there is a second unbounded dependency between the *wh*-word and the head of the extracted constituent.

Figure 20 shows the EPTD associated with the relative pronoun *qui* in the parsing of Sentence (23). The node *nNp* represents the noun phrase resulting from the modification of *Jean*, represented by the node *nNpO*, by the relative clause *dans l'entreprise de qui elle pense travailler bientôt*.

The extracted constituent *dans l'entreprise de qui* is represented by the node *nExtract*. This node shares the features *cat*, *prep*, and *ref* with *nTrace* but with different polarities. The node *nWh* represents a noun phrase reducing to the *wh*-word *qui* which may be embedded more or less deeply in the extracted constituent expressed with an underspecified dominance relation from *nExtract* to *nWh*. In Sentence (23), the chain of embedded constituents from *nWh* to *nExtract* contains a noun phrase (*qui*), a prepositional phrase (*de qui*), another noun phrase (*l'entreprise de qui*), and finally another prepositional phrase (*dans l'entreprise de qui*). Again, to restrict the possible chain to noun or prepositional phrases, a constraint is associated with the dominance relation from *nExtract* to *nWh*, in the form of the feature structure  $\{[cat : np | pp]\}$ .

A second underspecified dominance relation expresses that the trace of the extracted constituent may be embedded more or less deeply in a sequence of object clauses inside the relative clause. The most external object clause is represented by the node *nS1* and the most internal clause is represented by the node *nSO*. From the node *nS1* to the node *nSO*, there is an underspecified dominance relation. The following feature structure labels the underspecified dominance relation from *nS1* to *nSO*, to impose constraints on intermediate nodes:  $\{[cat : s | cs], [funct : obj | obj\_modal | obj\_cpl | caus]\}$ . In our example, *nS1* and *nSO* are merged in the representation of the object clause *travailler bientôt*.

The phenomenon of extraction with pied-piping is not specific to French. For instance, it is also present in English, with the same difficulties. That is why the study of related work is not restricted to French. Analyses of this phenomenon were proposed within various formalisms. Categorical Grammar (CG) uses the paradigm of parsing as deduction (Moot and Retoré 2012): grammars are lexicalized and the elementary units attached at words are logical types; then, parsing amounts to a proof in a logical framework. The information necessary to realize extraction is attached at *wh*-words, in the same way as in IG.



Instead of an underspecified dominance relation in an EPTD, it is represented by a third order logical type. In the parsing process, which takes the form of a deduction, an additional hypothesis representing a trace of the extracted constituent is introduced and it will be discharged when the extracted constituent will be composed with the rest of the clause that follows. The standard logical kernel, the Lambek Calculus (Lambek 1958), is too rigid to express complex phenomena like middle extraction or islands to extraction. One solution is to enrich the logical framework with modal operators (Morrill 1994; Moortgat 1996). Another solution is to add new deduction rules to the restricted logical framework of AB-grammars (Steedman 2000).

HPSG uses feature structure unification to model the two unbounded dependencies present in extraction with pied-piping (Sag *et al.* 2003). Unlike IG and CG, the information is not only anchored at the *wh*-word: a non-local feature *SLASH* expresses that an argument or an adjunct of a word is lacking; a dual feature is introduced by the *wh*-word. Then, the two features are propagated up inside the constituency structure, and they meet at the top through a filler-gap mechanism.

In LFG, Kaplan and Zaenen (1995) use the mechanism of functional uncertainty to express long distance dependencies in constructions with extraction and pied-piping. In the *c*-structure, the rule representing the concatenation of the extracted constituent with the rest of the clause is associated with an equation expressing a sharing between two features in the *f*-structure: the first feature is attached at the extracted constituent and the second feature is represented by its path from the top to a deep level in the *f*-structure. The underspecification of this path, its uncertainty, is represented with a regular expression using the Kleene closure operator. The principle of functional uncertainty must be related to the underspecified dominance relations of IG and possibility of constraining them with feature structures, even if it is less general.

In TAG (Joshi and Schabes 1997), the adjunction operation makes it possible to represent the dependency of an extracted constituent from a distant predicative expression, but since it is more rigid than the mechanism of PTD superposition, it requires that information must be attached to the verb governing the clause at the source of the extraction. This contributes to the concentration of grammatical information

in verb entries. Moreover, taking pied-piping into account needs an ad hoc mechanism.

8

## COMPARISON WITH OTHER FRENCH GRAMMARS AND EVALUATION OF THE GRAMMAR

There is very little work on the construction of French computational grammars from linguistic knowledge using semi-automatic tools. Historically, a very fruitful work was the PhD thesis of Candito (Candito 1999) about the modular organization of TAGs, with an application to French and Italian. This thesis was a source of inspiration for the development of several French grammars.

A first grammar produced according to this approach and able to parse large corpora was FRMG (Villemonde De La Clergerie 2010). FRMG falls within the TAG formalism and its originality lies in the use of specific operators on nodes to factorize trees: disjunction, guards, repetition, and shuffling. As a consequence, the grammar is very compact with only 207 trees. Moreover, these trees are not written by hand but they are automatically produced from a multiple inheritance hierarchy of classes (using a mechanism which is similar to the one used in XMG).

Another French grammar inspired by Candito (1999) is the French TAG grammar developed by Crabbé (2005). Like FRIGRAM, this grammar was written with XMG. Unlike FRMG, it is constituted of classical TAG elementary trees, hence its more extensive form: it includes 4,200 trees and essentially covers verbs. It was a purely syntactic grammar and it was later extended in the semantic dimension by Gardent and Parmentier (2007) for generation.

Evaluation of parsers is known to be a difficult task in general (Rimell and Clark 2008), mainly because each parser and each treebank is based on a large set of linguistic decisions ranging from the choice of category names and their definition to the choice of the head of constituents. Comparing a parsing result with a gold standard corpus requires conversion from one format to another and this conversion may induce biases.

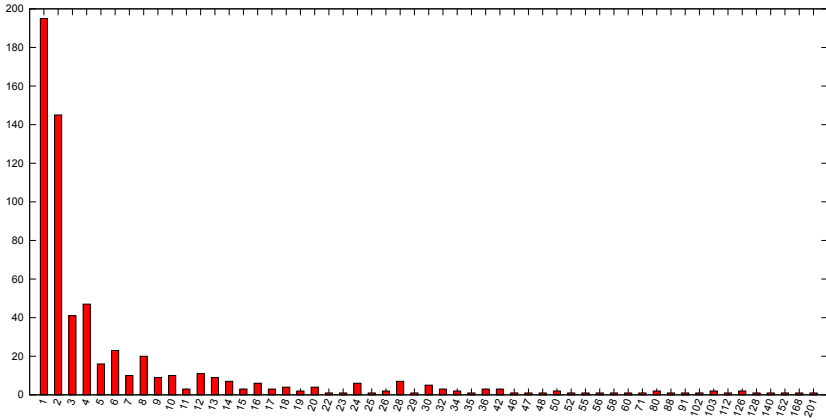
In the case of FRIGRAM, to evaluate the accuracy and coverage is even more problematic for two reasons. First, when building

FRIGRAM, we decide to focus on a set of linguistic phenomena and cover them in an exhaustive way. At the same time, we also decide not to take into account (at least in this version of the grammar) several kinds of phenomena. We think that a fully lexicalized grammar is not the most efficient way of modeling some phenomena that break the basic structure of natural language sentences; for instance, dislocation, non-constituents coordination, parenthetical clauses, parenthesis, or direct reported speech are not taken into account. In Dufour-Lussier *et al.* (2014), we develop further this idea and explain how we plan to use an external and complementary tool, either as a preprocessing or postprocessing step to deal with these phenomena.

The second reason is that there is no robust parser able to deal with IG. The tool developed so far (LEOPAR; Guillaume *et al.* 2008) was designed to experiment with, test and help in grammar development. It was later enriched with filtering algorithms to improve the supertagging stages of the parsing process. Nevertheless, it does not have any robust mechanism to deal with sentences that are not completely covered by the grammar. After filtering steps, deep parsing relies on an exhaustive search of tree description models, which is an NP-hard task. As a consequence, LEOPAR can be used to parse sentences of length up to 15 words.

For the reasons given above, evaluation on treebanks is difficult and will need further experiments with a robust parsing algorithm and processing dedicated to phenomena not covered by the grammar. The French TreeBank (FTB; Abeillé *et al.* 2003) contains 12,351 sentences; 2,769 of these sentences have the length lower than 15. The parser LEOPAR with the FRIGRAM resource is able to parse 44.35% of the 2,769 sentences considered. More recently, a new French corpus, Sequoia (Candito and Seddah 2012), has been built and is freely available; it contains 3,099 sentences taken from different kinds of sources. Again, out of the 1,307 sentences of the length lower than 15, 52.8% are parsed. Unparsed sentences are mainly due to the lack of robustness or phenomena that FRIGRAM does not take into account (unusual coordination, incomplete negation or frozen expression for instance). In the case of the parsed sentences, due to the full search of solutions, the ambiguity can sometimes be high. In Figure 21, we can see that most of the parsed sentences have one or two solutions, but this number can grow up to 200 for some rare cases. It can be observed that

Figure 21:  
Frequency of  
sentences by  
number of  
solutions



most of the ambiguities are due to the PP-attachment; as we do not use semantic or statistical information so far, it is difficult to decide correctly for a PP-attachment in case of ambiguity.

To deal with this ambiguity, we have designed a set of rules that assign a weight to each dependency structure that is produced. It is then possible to estimate the accuracy of the grammar, comparing the best weighted parsing and the gold standard, by giving the labeled and the unlabeled attachment scores. Some preliminary work on ranking based on training on a corpus were conducted but it does not give a significant improvement with respect to hand-crafted rules. The labeled attachment score (LAS) is the proportion of dependency links that are correctly predicted by the grammar, whereas the unlabeled attachment score (UAS) is the proportion of dependency links that have the correct head disregarding the link label. We used Sequoia as the gold standard. On the successfully parsed sentences of Sequoia, the LAS is 81.6% and the UAS is 84.0%. Again, many errors are linked to PP-attachments not correctly ranked by our rules. Recently, partially supervised learning techniques were used to improve the performances of FRMG (Villemonte De La Clergerie 2013); with this hybridization technique, the LAS score of FRMG on Sequoia is 85.21%.

All results on freely available data can be found on the FRIGRAM web page.<sup>12</sup>

<sup>12</sup><http://frig.loria.fr>

Raw corpora are useful to check for the robustness of a parsing system but they have some limitations concerning the coverage of the grammar. Test suites are built to overcome this limitation and give examples of a wider spectrum of grammatical phenomena. Such suites may include not only positive examples but also negative examples to test the overgeneration of the grammar. There exists such a suite for French, the TSNLP (Lehmann *et al.* 1996). On the set of grammatical sentences of the TSNLP, LEOPAR and FRIGRAM are able to parse 86% of the sentences. The remaining sentences correspond to sentences that should be covered by the robustness of the parser rather than by the detailed grammar (like, for instance, unusual kind of coordination, sentences with incomplete negations). For the ungrammatical sentences of the TSNLP, 37% are parsed by LEOPAR and FRIGRAM. The main sources of problems are: sentences that are syntactically correct but semantically incorrect, phonological rules, tricky rules for past participle agreement in French that are not encoded in the grammar.

To try to deal with TSNLP drawbacks, we have designed our own test suite which is complementary to the TSNLP; it contains 944 positive sentences and 192 negative ones. 97.5% of the grammatical sentences are parsed and the ratio is 19.8% for ungrammatical sentences. The reader can find the test suite and the parsing results on the same web page as before. When parsing succeeds, the list of dependency structures produced is also given. The variety of the examples gives a good idea of the coverage of FRIGRAM and the richness of dependency graphs helps to understand the subtlety of the grammar.

The next step to go ahead with FRIGRAM is to solve the bottleneck of the parser LEOPAR in order to parse raw corpora. We need to improve the efficiency of the parser to contain the possible explosion resulting from the increase of the grammar size in combination with the increased sentence length. It is also necessary to take robustness into account in the parsing algorithm and add extra-grammatical procedures to deal with phenomena that we do not want to model by the lexicalized grammar.

For English, Tabatabayi Seifi (2012) is the first attempt to build an interaction grammar, which should be extended in order to have a coverage equivalent to the one of FRIGRAM.

## ACKNOWLEDGEMENTS

The authors wish to thank the reviewers for their useful remarks on previous versions of the article. They also thank Claire Gardent for her careful reading of the final version.

## REFERENCES

- Anne ABEILLÉ, Lionel CLÉMENT, and François TOUSSENEL (2003), Building a Treebank for French, in *Treebanks. Building and Using Parsed Corpora*, pp. 165–187, Kluwer Academic Publishers.
- Anne ABEILLÉ and Danièle GODARD (1997), The Syntax of French Negative Adverbs, in *Negation and polarity: syntax and semantics*, pp. 1–27, John Benjamins publishing Company.
- Anne ABEILLÉ, Danièle GODARD, and Philip MILLER (1997), Les causatives en français, un cas de compétition syntaxique, *Langue française*, 115:62–74.
- Jason BALDRIDGE, Sudipta CHATTERJEE, Alexis PALMER, and Ben WING (2007), DotCCG and VisCCG: Wiki and Programming Paradigms for Improved Grammar Engineering with OpenCCG, in *Proceedings of the Grammar Engineering Across Frameworks Workshop (GEAF 07)*, pp. 5–25, CSLI Studies in Computational Linguistics ONLINE, Dagstuhl, Germany.
- Emily BENDER (2008), Grammar Engineering for Linguistic Hypothesis Testing, in *Proceedings of the Texas Linguistics Society X Conference: Computational Linguistics for Less-Studied Languages*, pp. 16–36.
- Guillaume BONFANTE, Bruno GUILLAUME, and Mathieu MOREY (2009), Dependency Constraints for Lexical Disambiguation, in *Proceedings of the 11th International Conference on Parsing Technology (IWPT 2009)*, pp. 242–253, Paris, France, <http://www.aclweb.org/anthology/W09-3840>.
- Joan BRESNAN (2001), *Lexical-Functional Syntax*, Blackwell Publishers, Oxford.
- Miriam BUTT, Helge DYVIK, Tracy H. KING, Hiroshi MASUICHI, and Christian ROHRER (2002), The Parallel Grammar Project, in *Proceedings of the Workshop on Grammar Engineering and Evaluation (COLING 2002)*, pp. 1–7, Association for Computational Linguistics, Stroudsburg, PA, USA.
- Marie CANDITO and Djamé SEDDAH (2012), Le corpus Sequoia : annotation syntaxique et exploitation pour l'adaptation d'analyseur par pont lexical, in

*Proceedings of the Joint Conference JEP-TALN-RECITAL 2012*, pp. 321–334, ATALA/AFCP, Grenoble, France, <http://www.aclweb.org/anthology/F12-2024>.

Marie-Hélène CANDITO (1999), *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien*, Ph.D. thesis, Université Paris 7.

Jinho D. CHOI and Martha PALMER (2010), Robust Constituent-to-Dependency Conversion for Multiple Corpora in English, in *Proceedings of the 9th International Workshop on Treebanks and Linguistic Theories (TLT-9)*, pp. 55–66, Tartu, Estonia.

Ann COPESTAKE and Dan FLICKINGER (2000), An Open Source Grammar Development Environment and Broad-coverage English Grammar Using HPSG, in *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*, pp. 591–600, Athens, Greece.

Benoit CRABBÉ (2005), *Représentation informatique de grammaires fortement lexicalisées : application à la grammaire d'arbres adjoints*, Ph.D. thesis, Université Nancy2.

Benoit CRABBÉ, Denys DUCHIER, Claire GARDENT, Joseph LE ROUX, and Yannick PARMENTIER (2013), XMG: eXtensible MetaGrammar, *Computational Linguistics*, 39(3):1–66.

Valmi DUFOUR-LUSSIER, Bruno GUILLAUME, and Guy PERRIER (2014), Parsing Coordination Extragrammatically, in Zygmunt VETULANI and Joseph MARIANI, editors, *Human Language Technology. Challenges for Computer Science and Linguistics. 5th Language and Technology Conference, LTC 2011, Poznan, Poland, November 25-27, 2011, Revised Selected Papers*, Human Language Technology Challenges for Computer Science and Linguistics, pp. 55–66, Springer International Publishing.

Catherine FUCHS, Nathalie FOURNIER, and Pierre LE GOFFIC (2008), Structures à subordonnée comparative en français. Problèmes de représentations syntaxiques et sémantiques, *Linguisticae Investigationes*, 31(1):11–61.

Claire GARDENT and Yannick PARMENTIER (2007), SemTAG: a platform for specifying Tree Adjoining Grammars and performing TAG-based Semantic Construction, in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007)*, pp. 13–16, Prague, Czech Republic.

Danièle GODARD (2004), French Negative Dependency, in Francis CORBLIN and Henriëtte DE SWART, editors, *Handbook of French Semantics*, pp. 351–390, Center for the Study of Language and Information.

Maurice GREVISSE and André GOOSSE (2008), *Le bon usage: et son édition Internet*, Grevisse langue française, De Boeck Supérieur.

- Bruno GUILLAUME, Joseph LE ROUX, Jonathan MARCHAND, Guy PERRIER, Karèn FORT, and Jenifer PLANUL (2008), A Toolchain for Grammmarians, in *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008) - Demonstration*, pp. 9–12, Manchester, United Kingdom.
- Bruno GUILLAUME and Guy PERRIER (2009), Interaction Grammars, *Research on Language and Computation*, 7:171–208.
- Hélène HUOT (1982), Constructions infinitives du français : le subordonnant "de", *L'information grammaticale*, 15(1):40–45.
- Angelina IVANOVA, Stephan OEPEN, Lilja ØVRELID, and Dan FLICKINGER (2012), Who Did What to Whom?: A Contrastive Study of Syntacto-semantic Dependencies, in *Proceedings of the Sixth Linguistic Annotation Workshop, LAW VI '12*, pp. 2–11, Association for Computational Linguistics, Jeju, Republic of Korea.
- Aravind K. JOSHI, Leon S. LEVY, and Masako TAKAHASHI (1975), Tree Adjunct Grammars, *Journal of Computer and System Sciences*, 10(1):136–162.
- Aravind K. JOSHI and Yves SCHABES (1997), Tree-Adjoining Grammars, in *Handbook of formal languages*, pp. 69–123, Springer.
- Ronald M. KAPLAN and Annie ZAENEN (1995), Long-distance Dependencies, Constituent Structure, and Functional Uncertainty, *Formal Issues in Lexical-Functional Grammar*, 47:137–165.
- Jong-Bok KIM and Ivan A. SAG (2002), Negation without Head-Movement, *Natural Language & Linguistic Theory*, 20(2):339–412.
- Joachim LAMBEK (1958), The Mathematics of Sentence Structure, *Amer. Math. Monthly*, 65:154–170.
- Sabine LEHMANN, Stephan OEPEN, Sylvie REGNIER-PROST, Klaus NETTER, Veronika LUX, Judith KLEIN, Kirsten FALKEDAL, Frederik FOUVRY, Dominique ESTIVAL, Eva DAUPHIN, Hervé COMPAGNION, Judith BAUR, Lorna BALKAN, and Doug ARNOLD (1996), TSNLP - Test Suites for Natural Language Processing, in *Proceedings of the 16th International Conference on Computational Linguistics (COLING 1996)*, pp. 711–716, Copenhagen, Denmark, <http://aclweb.org/anthology/C96-2120>.
- Michael MOORTGAT (1996), Categorical Type Logics, in J. VAN BENTHEM and A. TER MEULEN, editors, *Handbook of Logic and Language*, pp. 93–177, Elsevier.
- Richard MOOT and Christian RETORÉ (2012), *A Logic for Categorical Grammars: Lambek's Syntactic Calculus*, Springer.
- Glyn V. MORRILL (1994), *Type Logical Grammar*, Kluwer Academic Publishers, Dordrecht and Hingham.
- Stephan OEPEN, Dan FLICKINGER, Jun'ichi TSUJII, and Hans USZKOREIT, editors (2002), *Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing*, CSLI Lecture Notes, CSLI Publications, Stanford.



- Guy PERRIER (2000), Interaction Grammars, in *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pp. 600–606, Sarrebrücken, Germany.
- Guy PERRIER (2014), FRIGRAM: a French Interaction Grammar, Research Report RR-8323, Inria Nancy.
- Carl J. POLLARD and Ivan A. SAG (1994), *Head-Driven Phrase Structure Grammar*, University of Chicago Press.
- Geoffrey K. PULLUM and Barbara C. SCHOLZ (2001), On the Distinction between Model-Theoretic and Generative-Enumerative Syntactic Frameworks, in *Proceedings of the 4th International Conference on Logical Aspects of Computational Linguistics (LACL 2001)*, volume 2099 of *Lecture Notes in Computer Science*, pp. 17–43, Le Croisic, France.
- Martin RIEGEL, Jean-Christophe PELLAT, and René RIOUL (1999), *Grammaire méthodique du français*, Presses universitaires de France.
- Laura RIMELL and Stephen CLARK (2008), Constructing a Parser Evaluation Scheme, in *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation (COLING 2008)*, pp. 44–50, Manchester, United Kingdom.
- James ROGERS and K. VIJAY-SHANKER (1994), Obtaining Trees from their Descriptions: an Application to Tree-Adjoining Grammars, *Computational Intelligence*, 10(4):401–421.
- Laurent ROMARY, Susanne SALMON-ALT, and Gil FRANCOPOULO (2004), Standards going concrete: from LMF to Morphalou, in Michael ZOCK, editor, *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pp. 22–28, Geneva, Switzerland, <http://acl.ldc.upenn.edu/w/W04/W04-2104.bib>.
- Ivan A. SAG, Thomas WASOW, and Emily M. BENDER (2003), *Syntactic Theory: a Formal Introduction*, Center for the Study of Language and Information.
- Mark STEEDMAN (2000), *The Syntactic Process*, Bradford Books, MIT Press.
- Shohreh TABATABAYI SEIFI (2012), An Interaction Grammar for English Verbs, in Rasmus K. RENDSVIG and Sophia KATRENKO, editors, *Proceedings of the ESSLLI 2012 Student Session*, pp. 160–169, Opole, Poland, <http://ceur-ws.org/Vol-954/paper17.pdf>.
- Karel VAN DEN EYNDE and Piet MERTENS (2003), La valence : l’approche pronominale et son application au lexique verbal, *Journal of French Language Studies*, 13:63–104.
- Éric VILLEMONTÉ DE LA CLERGERIE (2010), Building factorized TAGs with meta-grammars, in *Proceedings of the 10th International Conference on Tree Adjoining Grammars and Related Formalisms (TAG+10)*, pp. 111–118, New Haven, CO, USA.

*Guy Perrier, Bruno Guillaume*

Éric VILLEMONTÉ DE LA CLERGERIE (2013), Improving a symbolic parser through partially supervised learning, in *The 13th International Conference on Parsing Technologies (IWPT 2013)*, pp. 54–62, Nara, Japan, <https://hal.inria.fr/hal-00879358>.

XTAG RESEARCH GROUP (2001), A Lexicalized Tree Adjoining Grammar for English, Technical Report IRCS-01-03, IRCS, University of Pennsylvania.

Nicholas YATES (2002), French Causatives: a Biclausal Account in LFG, in *Proceedings of the LFG02 Conference*, pp. 390–407, Athens, Greece.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*

<http://creativecommons.org/licenses/by/3.0/>

