Journal of Language Modelling

VOLUME 5 ISSUE 2
SEPTEMBER 2017

Institute of Computer Science
Polish Academy of Sciences
Warsaw

Journal of Language Modelling

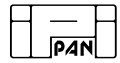
VOLUME 5 ISSUE 2
SEPTEMBER 2017

Editorials

Articles

JOURNAL OF
LANGUAGE MODELLING

ISSN 2299-8470 (electronic version)
ISSN 2299-856X (printed version)
http://jlm.ipipan.waw.pl/



Published by IPI PAN
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 XeLaTeX using the typefaces: *Playfair Display*
by Claus Eggers Sørensen, *Charis SIL* by SIL International,
JLM monogram by Łukasz Dziedzic.

EDITORIAL BOARD

An outline of type-theoretical approaches to lexical semantics

Robin Cooper[1] *and Christian Retoré*[2]
[1] University of Gothenburg
[2] Université de Montpellier & LIRMM

ABSTRACT

We take the opportunity of the publication of some of the papers of the ESSLLI workshop TYTLES (*TYpe Theory and LExical Semantics*, ESSLLI 2015, Barcelona) to provide an overview of the possibilities that type theory offers to model lexical semantics, especially the type-theoretical frameworks that properly model compositional semantics.

Keywords: lexical semantics; compositional semantics; type theory; lambda calculus

ORIGINS OF THIS ISSUE: ESSLLI WORKSHOP ON *TYPE THEORY AND LEXICAL SEMANTICS (2015)*

The program of the ESSLLI 2015 workshop held in Barcelona[1] consisted of twelve selected talks. The corresponding extended abstracts, together with an introduction and a conclusion by the workshop organisers, are available on the web as Cooper and Retoré (2015); it includes:

A. Introduction (slides), by Robin Cooper and Christian Retoré.
B. Justyna Grudzińska and Marek Zawadowski. *A Puzzle about Long-distance Indefinites and Dependent Type Semantics*.
C. Stergios Chatzikyriakidis, Mathieu Lafourcade, Lionel Ramadier and Manel Zarrouk. *Type Theories and Lexical Networks: Using Serious Games as the Basis for Multi-Sorted Typed Systems*.

[1] On 17 August 2017, while writing this introduction, we learnt about the tragic attack in Barcelona, where some friends and colleagues live. We would like to express our sympathy.

D. Staffan Larsson. *Perceptual Meaning in TTR Judgement-based Semantics and Conceptual Spaces*.

E. Simon Dobnik. *Interfacing Language, Spatial Perception and Cognition in Type Theory with Records*.

F. Peter Sutton and Hana Filip. *Probabilistic Mereological TTR and the Mass/Count Distinction*.

G. Ellen Breitholtz. *Are Widows Always Wicked? Learning Concepts through Enthymematic Reasoning*.

H. Bruno Mery. *The Relative Complexity of Constraints in Co-Predicative Utterances*.

I. Daisuke Bekki and Miho Satoh. *Calculating Projections via Type Checking*.

J. Laura Kallmeyer, Timm Lichte, Rainer Osswald, Sylvain Pogodalla and Christian Wurm. *Quantification in Frame Semantics with Hybrid Logic*.

K. Livy Real and Alexandre Rademaker. *An Overview on Portuguese Nominalisation*.

L. Pepijn Kokke. *Formalising type-logical grammars in Agda*.

M. Seohyun Im and Chungmin Lee. *A Developed Analysis of Type Coercion Using Asher's TCL and Conventionality*.

N. Ribeka Tanaka, Koji Mineshima and Daisuke Bekki. *Factivity and Presupposition in Dependent Type Semantics*.

O. Conclusion (slides), by Robin Cooper and Christian Retoré.

Some of these papers were submitted and some of these are now included in this issue of the *Journal of Language Modelling* on *Type theory and lexical semantics*.

Let us briefly present this fruitful connection of lasting interest.

1 A COMPOSITIONAL VIEW OF LEXICAL SEMANTICS

The relation between lexical semantics and type theory is rather unnatural if one thinks of *lexical semantics* as defined, e.g., in the article *Lexical Semantics* in the *Oxford Research Encyclopedia of Linguistics* (Geeraerts 2017):

> Lexical semantics is the study of word meaning. Descriptively speaking, the main topics studied within lexical semantics involve either the internal semantic structure of words, or the semantic relations that occur within the vocabulary. Within the first set, major phenomena include polysemy (in contrast with vagueness), metonymy, metaphor, and prototypicality. Within the second set, dominant topics include lexical fields, lexical relations, conceptual metaphor and metonymy, and frames.

If we have a look from the other side, i.e., philosophy of language, where logic, compositional semantics and type theory live, the connection is at least evoked as in these words from the entry on Word Meaning in the Stanford Encyclopaedia of Philosophy (Gasparri and Marconi 2016):

> Word meaning has played a somewhat marginal role in early contemporary philosophy of language, which was primarily concerned with the structural features of sentences and showed less interest in the format of lexical representations and in the nature of the word-level input to compositional processes. Nowadays, it is well-established that the way we account for word meaning is bound to have a major impact in tipping the balance in favor or against a given picture of the fundamental properties of human language. This entry provides an overview of the way issues related to lexical meaning have been explored in analytic philosophy and a summary of relevant research on the subject in neighboring scientific domains.

So this survey, as well as the workshop, is devoted to the study of lexical semantics in a compositional framework deriving – roughly speaking – from Montague semantics and the lexical issues to be dealt with are:

- word meaning in context (various forms of polysemy),
- relation between meanings,
- relation between lexical networks and lexical semantics.

Observe that from a logical viewpoint, relations between meanings are naturally higher order relations which oblige us to go beyond

first order logic and type theories are naturally higher order – of course reification à la Davidson is possible, but still rather unnatural and less compositional.

Usual techniques for taking into account at least part of lexical semantics are the descriptions using features (e.g. *human/non-human*), argument structures which specify the nature of arguments to predicates and the composition of word vectors that has been quite fashionable recently although it obliges us to leave out some of the logical structure involved in compositional semantics such as negation.

2 SOME ASPECTS OF LEXICAL SEMANTICS IN COMPOSITIONAL FRAMEWORKS

2.1 *Polysemy*

Polysemy is the phenomenon that a single word or expression has several readings. It is common to distinguish various forms of polysemy.

Simple polysemy might be viewed as the coincidence that a word has several unrelated meanings, which in some contexts may be hard to choose between, as in (1).

(1) a. The river flowed by the bank.
b. The bank is near the river.
c. The bank phoned me.

This should be distinguished from words that have several interrelated meanings derived from a root meaning. An institution like a journal or a town have such aspects, which are also called facets, as shown in (2).

(2) a. The journal is printed on pink paper.
b. The journal hired a new commentator.
c. The journal is near the port.

Events are a special case of this, and they play a particular role in semantics. Deverbals may refer to aspects of an action verb such as the process, the result, the place or the material used, as in (3)–(4). There is a rich literature on the topic, see, e.g., the references in Real and Retoré (2014).

(3) a. The signature took three months.
b. The signature is unreadable.

(4) a. The building in front of my house took three months.
b. The building in front of my house is ugly.

There is a special form of polysemy where the two aspects are strongly linked: one aspect does not exclude the other, on the contrary you cannot have one without the other. This has some consequences for the individuation process and the interpretation of the quantifiers, as shown in (5).

(5) a. I carried all the books from the library to the attic.
b. I read all the books in the library.

There are examples that are hard to understand without the context, which can be linguistic or extralinguistic, as in (6).

(6) a. I am parked behind a blue BMW.
b. The ham sandwich asked for a coffee.

2.2 *Co-predication*

Given that compositional semantics is quite interested in the logical structure of sentences, it is normal that it has been studying how one can conjoin the properties of a word, properties which may concern only a single aspect of this word, as in (7)–(10).

(7) Dinner was delicious but took ages. (event/food)

(8) * The salmon we had for lunch was lightning fast. (animal/food)

(9) a. I left my preferred book on logic on the table. (physical/information)
b. I carried the books from the shelf to the attic since I already read them. (physical/information)

(10) a. Liverpool is a poor town and an important harbour. (people/geographic)
b. * Liverpool defeated Chelsea and is an important harbour. (football/geographic)

It can be observed that in some thematic contexts or contrasts a priori infelicitous co-predication may become felicitous.

(11) a. Barcelona won four champions leagues and organised the olympiads.

b. Libourne, a small south-west town, defeated Lille.

Deverbals rarely allow co-predications on their different facets, as discussed in Real and Retoré (2014) and at the workshop, in Livy Real's talk.

3 INTEGRATING LEXICAL SEMANTICS INTO A COMPOSITIONAL AND COMPUTATIONAL FRAMEWORK

Standard lexical semantics, including distributional semantics as used in natural language processing, involving big data, machine learning, information retrieval, and so on, is mainly concerned with *what a sentence or a text is about* in terms of empirically grounded word meanings. For instance, word vectors are derived from the cooccurrences of words in texts. They are especially good for the study of semantic similarity: the cosine measure of similarity or the products of vectors by matrices may model the combination of a verb and its object or of an adjective with a noun, etc.

Formal semantics is rather concerned with logical and pragmatic relations: *what a sentence (or discourse) asserts, denies, supposes*, how noun phrases and pronouns (co)refer to individuals and sets, in which situations (or worlds) sentences are true. It is also concerned with the interpretation of modality, aspect and tense. Usually intepretation assumes that lexical meaning has been determined in some way external to the semantics. It is carried out in two steps: word meanings are combined according to syntactic analysis into a logical formula, which is thereafter interpreted in terms of some semantics, usually possible worlds semantics, although other interpretations of logical formulas are possible, like situation semantics or game-theoretical semantics.

These two approaches are complementary, and an adequate theory of semantics should take both into account. For instance, if one is looking for an answer to the question whether Geach was a student of Wittgenstein, one can find in the French Wikipedia (contradicting the English version):

(12) In 1941, [Geach] married Elisabeth Anscombe, through whom he got in contact with Wittgenstein. Although he never attended the lectures of the latter, he was strongly influenced by him.

Both word meaning (“student of”, “to attend some lectures by”) and sentence/discourse structure are needed to understand this text (scope of the negative “never”, reference of pronouns like “he”, “whom”, “the latter”, “his” and “him”).

In Human Machine Interaction large scale analysis on the fly is not practical but proper understanding is still important. For instance, if a parent says “the children want pizza” to some McDonald’s-like automaton, the person who treats the order needs to know whether to prepare one or several pizzas and in this case the system should know that this has not been determined by the utterance and should therefore ask for a clarification.

3.1 *Pustejovsky’s generative lexicon: a framework for polysemy*

Important and pioneering work on polysemy has been carried out by Pustejovsky. Although those questions have been studied at least since the 1970s (Apresjan 1974; Bierwisch 1979, 1983; Nunberg 1979, 1995; Cruse 1986; see, e.g., Lauer 2004 or Dölling 2018 for survey and comparisons), Pustejovsky (1991, 1995) was the first to propose a formal compositional framework for handling word meaning and the transformation of word meaning in context.

The basic components of Pustejovsky’s approach are:

- a compositional (generative) view of word meaning,
- a formal framework: word meaning as complex feature structures (he way they combine is less specified),
- computational tractability.

There are four levels in an entry of the generative lexicon:

- lexical typing structure: giving an explicit type for a word positioned within a type system for the language,
- argument structure: specifying the number and nature of arguments to a predicate,
- event structure: defining the event type of the expression and any subevent structure it may have,
- qualia structure: a structural differentiation of the predicative force for a lexical item organised in four quale:
  - formal: the basic category which distinguishes the meaning of a word within a larger domain,

- constitutive: the relation between an object and its constituent parts,
- telic: the purpose or function of the object, if there is one,
- agentive: the factors involved in the object's origins or "coming into being".

Types play an important role in the generative lexicon. There are base types organised as an ontology. Functional types are also used, in particular, in the argument structure.

To sum up, the generative lexicon is the first compositional semantic framework that integrates some aspects of lexical meaning. Some of the structures and notions involved in this framework are fully formalised, but not all of them. For instance, the structure of the entries is completely formalised. Nevertheless some aspects remain to be made precise, such as the set of base types and their ontology. When it comes to the way lexical items combine, the composition modes and rules are mainly described in terms of examples, whereas automated semantic analysis would require a general procedure as well as a precise correspondence between syntactic operations and semantic rules. So one may wonder whether this framework is already able to compute the semantics of a whole complex sentence, or of a small discourse.

It is worth noticing that some important parts of the generative lexicon can be learnt, in particular the qualia structure (Claveau *et al.* 2003). It is still an open question whether other fields than qualia structure can be learnt.

3.2 *Lexical semantics and compositionality*

3.2.1 Selectional restrictions

One way to start addressing lexical issues in compositional semantics concerns selectional restrictions, as in (13).

(13) The chair barked.

(14) Dictionary: "barks" is said of an animal, usually a dog.

A commonly adopted idea is that infelicitous semantic composition is a type mismatch: a predicate P over A entities (a function from objects of type A to propositions or truth values) is applied to an entity t of type B with $B \neq A$:

$$P^{A \to \text{prop}}(t^{B})$$

as it is the case in example (13):

$$\textit{Bark}^{\textit{animal}\to\textit{prop}}(\textit{the chair})^{\textit{physical object}}$$

This constraint needs to be relaxed in certain contexts. While (13) sounds strange, (15a) is much easier to interpret and (15b) provides a naturally occurring example.

(15) a. I was so late for registration that the secretary barked at me.

b. Bow Wow barked at on Twitter for claiming he flies private (`https://www.cnet.com/news/bow-wow-barked-at-on-twitter-for-claiming-he-flies-private/`, 27/8/2017)

Observe that meaning transfers are idiosyncratic. In French you can say that either a tyre or a car is "punctured", as in (16a) and (16b). Correspondingly, in English you can say that a tyre or a car has a puncture, as in (16c). However, while you can say that a tyre is flat or punctured, as in (16d), in English you cannot say that a car is flat or punctured, as in (16e).

(16) a. Le pneu est crevé.
The tyre is punctured.

b. Ma voiture est crevée.
My car is punctured.
My car has a puncture.

c. My tyre/car has a puncture.

d. My tyre is flat/punctured.

e. #My car is flat/punctured.

Idiosyncratic phenomena can even be observed in the same language. Indeed some words with the same "ontological type" may have different meanings. For instance in French, of two words designating a set of students, namely *classe* (class/classroom) and *promotion* (year group) only *classe* may mean classroom.

(17) a. La classe de CP a été repeinte pendant les
The class of 1st-year was repainted during the
vacances.
holidays.

b. #La promotion 2015 a été repeinte pendant les
The year-group 2015 was repainted during the
vacances.
holidays.

One issue is whether adaptation of word meaning to context should be word-driven or type-driven. A related issue is whether the base types should be related to ontological classes or to linguistic behaviours, different answers being developed by Asher (2011), Retoré (2014), Kinoshita *et al.* (2017), Chatzikyriakidis and Luo (2017), Mery and Retoré (2017).

4 USING TYPES FOR LEXICAL SEMANTICS

There are three broad areas relating to the lexicon represented in the papers here which suggest that a type-theoretical approach can be useful. These are:

- dynamic aspects of the lexicon,
- use of dependent types,
- probability.

We will discuss each of these in turn.

4.1 *Dynamic aspects of the lexicon*

There is general agreement in these papers that lexical meaning is to be treated dynamically. This idea relates, of course, to the original work on the generative lexicon and notions of coercion. But it also relates to the fact that we are constantly learning new words and meanings for words, that lexical meaning is in flux.

In type-theoretical approaches there is a focus on the types of objects rather than the sets of objects (witnesses or inhabitants) which belong to those types. In introducing types we attempt to define the conditions under which an object would belong to the type rather than simply associating a set of objects with the type. This means that we can adapt a type theory to models where the set of witnesses of a type may change dynamically over time, without the type itself thereby changing. It introduces the possibility of modelling how we observe new witnesses of a type as we discover more of the world. This is

different from a Montagovian notion of sense: a function from possible worlds and times to extensions. If you discover a new object at a given world and time, then the Montagovian sense is different.

Another dynamic aspect offered by a type theory is that the type associated with a word can change over time. Many modern type theories offer a notion of structured types (such as record types) which allows us to give an account of change not available in a Montagovian sense. For example, a record type can be changed by adding or removing a field whereas in a Montagovian sense the only structure we have is that of the set of ordered pairs which is the function from world-time pairs to extensions.

Papers relating to some kind of dynamic aspect of types in this issue are those by Chatzikyriakidis *et al.* and Dobnik *et al.*

The notion of structured type figures indirectly in the paper by Kallmeyer *et al.* The notion of frame which they introduce in terms of hybrid logic relates intuitively to the notion of frame in terms of record types discussed in Cooper (2016). It would be interesting, for example, to explore whether the expressions of hybrid logic can be thought of as record types modelling event types.

4.2 *Dependent types*

Dependent types are parametrised types which return a type depending on what objects are provided for their parameters. They can be thought of a functions from objects to types. A classical use of dependent types is for donkey anaphora, as first presented in Sundholm (1986) and discussed in Ranta (1994). However, a number of other uses have been pointed out in the literature. In the papers in this issue their use is discussed for presupposition (Tanaka *et al.*).

4.3 *Probability*

In standard type theory judgements that objects are of a given type are categorical: either an object is of a type or it is not. However, it seems intuitive that agents make probabilistic judgements: it is probable that a given object is of a given type, but it is not certain. Cooper *et al.* (2015) proposed a probabilistic type theory that could be used for natural language semantics and in this issue Sutton *et al.* apply this to the analysis of the mass/count distinction.

5 CONCLUSION

This special issue represents a broad span of approaches using different type theories but, as we have tried to point out in this introduction, they share a number of common assumptions and goals. This bodes well for future research on *type theory and lexical semantics*.

REFERENCES

Juri APRESJAN (1974), Regular Polysemy, *Linguistics*, 14:5–32.

Nicholas ASHER (2011), *Lexical Meaning in context – a web of words*, Cambridge University press.

Manfred BIERWISCH (1979), Wörtliche Bedeutung - eine pragmatische Gretchenfrage, in G. GREWENDORF, editor, *Sprechakttheorie und Semantik*, pp. 119–148, Surkamp, Frankfurt.

Manfred BIERWISCH (1983), Semantische und konzeptuelle Repräsentation lexikalischer Einheiten, in R. RŮŽIČKA and W. MOTSCH, editors, *Untersuchungen zur Semantik*, pp. 61–99, Akademie-Verlag, Berlin.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2017), On the Interpretation of Common Nouns: Types Versus Predicates, in Stergios CHATZIKYRIAKIDIS and Zhaohui LUO, editors, *Modern Perspectives in Type Theoretical Semantics*, pp. 43–70, Springer.

Vincent CLAVEAU, Pascale SÉBILLOT, Cécile FABRE, and Pierrette BOUILLON (2003), Learning Semantic Lexicons from a Part-of-Speech and Semantically Tagged Corpus Using Inductive Logic Programming, *Journal of Machine Learning Research*, 4:493–525.

Robin COOPER (2016), Frames as Records, in Annie FORET, Glyn MORRILL, Reinhard MUSKENS, Rainer OSSWALD, and Sylvain POGODALLA, editors, *Formal Grammar: 20th and 21st International Conferences FG 2015, Barcelona, Spain, August 2015, Revised Selected Papers FG 2016, Bozen, Italy, August 2016, Proceedings*, number 9804 in LNCS, pp. 3–18, Springer.

Robin COOPER, Simon DOBNIK, Shalom LAPPIN, and Staffan LARSSON (2015), Probabilistic Type Theory and Natural Language Semantics, *Linguistic Issues in Language Technology*, 10(4):1–45.

Robin COOPER and Christian RETORÉ (2015), *Extended abstracts of the ESSLLI 2015 workshop TYTLES: Types Theory and Lexical Semantics*, HAL Archives Ouvertes, `https://hal.archives-ouvertes.fr/hal-01584832`.

D.A. CRUSE (1986), *Lexical semantics*, Cambridge textbooks in linguistics, Cambridge University Press, ISBN 9780521276436, `http://books.google.fr/books?id=xDSBaet2uSsC`.

Johannes DÖLLING (2018), Systematic polysemy, in Lisa MATTHEWSON, Cécile MEIER, Hotze RULLMANN, and Thomas Ede ZIMMERMANN, editors, *The Blackwell Companion to Semantics*, Blackwell.

Luca GASPARRI and Diego MARCONI (2016), Word Meaning, in Edward N. ZALTA, editor, *The Stanford Encyclopedia of Philosophy*, Metaphysics Research Lab, Stanford University, spring 2016 edition, `https://plato.stanford.edu/archives/spr2016/entries/word-meaning/`.

Dirk GEERAERTS (2017), Lexical Semantics, in *Oxford Research Encyclopedia of Linguistics*, Oxford University Press, doi:10.1093/acrefore/9780199384655.013.29, `http://linguistics.oxfordre.com/view/10.1093/acrefore/9780199384655.001.0001/acrefore-9780199384655-e-29`.

Eriko KINOSHITA, Koji MINESHIMA, and Daisuke BEKKI (2017), An Analysis of Selectional Restrictions with Dependent Type Semantics, in Setsuya KURAHASHI, Yuiko OHTA, Sachiyo ARAI, Ken SATOH, and Daisuke BEKKI, editors, *New Frontiers in Artificial Intelligence - JSAI-isAI 2016 Workshops, LENLS, HAT-MASH, AI-Biz, JURISIN and SKL, Kanagawa, Japan, November 14-16, 2016, Revised Selected Papers*, volume 10247 of *Lecture Notes in Computer Science*, pp. 19–32, Springer.

Sven LAUER (2004), *A Comparative Study Of Current Theories Of Polysemy In Formal Semantics*, Master's thesis, Cognitive science Osnabrück - Computational Linguistics, `http://cogsci.uni-osnabrueck.de/~CL/`.

Bruno MERY and Christian RETORÉ (2017), Classifiers, Sorts, and Base Types in the Montagovian Generative Lexicon and Related Type Theoretical Frameworks for Lexical Compositional Semantics, in Stergios CHATZIKYRIAKIDIS and Zhaohui LUO, editors, *Modern Perspectives in Type Theoretical Semantics*, pp. 163–188, Springer.

Geoffrey NUNBERG (1979), The Non-Uniqueness of Semantic Solutions: Polysemy, *Linguistic and Philosophy*, 3:143–184.

Geoffrey NUNBERG (1995), Transfers of meaning, *Journal of semantics*, 12(2):109–132.

James PUSTEJOVSKY (1991), The Generative Lexicon, *Computational Linguistics*, 17(4):409–441.

James PUSTEJOVSKY (1995), *The generative lexicon*, M.I.T. Press.

Aarne RANTA (1994), *Type-Theoretical Grammar*, Clarendon Press, Oxford.

Livy REAL and Christian RETORÉ (2014), Deverbal semantics and the Montagovian generative lexicon ΛTy_n, *Journal of Logic Language and Information*, doi:10.1007/s10849-014-9187-y, 10.1007/s10849-014-9187-y.

Christian RETORÉ (2014), The Montagovian Generative Lexicon ΛTy_n: a Type Theoretical Framework for Natural Language Semantics, in Ralph MATTHES

and Aleksy SCHUBERT, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 202–229, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, ISBN 978-3-939897-72-9, ISSN 1868-8969, doi:10.4230/LIPIcs.TYPES.2013.202.

Göran SUNDHOLM (1986), Proof Theory and Meaning, in Dov GABBAY and Franz GUENTHNER, editors, *Handbook of Philosophical Logic, Vol. III*, Reidel, Dordrecht.

Scope ambiguities, monads and strengths

Justyna Grudzińska[1] *and Marek Zawadowski*[2]
[1] Institute of Philosophy, University of Warsaw, Warsaw, Poland
[2] Institute of Mathematics, University of Warsaw, Warsaw, Poland

ABSTRACT

In this paper, we will discuss three semantically distinct scope assignment strategies: traditional movement strategy, polyadic approach, and continuation-based approach. Since generalized quantifiers on a set X are elements of $\mathscr{C}(X)$, which is the value of the continuation monad $\mathscr{C}$ on X, quantifier phrases are interpreted as $\mathscr{C}$-computations, in all three approaches. The main goal of this paper is to relate the three strategies to the computational machinery connected to the monad $\mathscr{C}$ (strength and derived operations). As will be shown, both the polyadic approach and the continuation-based approach make heavy use of monad constructs. In the traditional movement strategy, monad constructs are not used but we still need them to explain how the three strategies are related and what can be expected of them with regard to handling scopal ambiguities in simple sentences.

Keywords: scope ambiguity, continuation monad, strength

1 MULTI-QUANTIFIER SENTENCES AND THREE SCOPE-ASSIGNMENT STRATEGIES

Multi-quantifier sentences can be ambiguous, with different readings corresponding to how various quantifier phrases (QPs) are semantically related in the sentence. For example,

(1) Every girl likes a boy

admits of the subject wide-scope reading ($S > O$) where each girl likes a potentially different boy, and the object wide-scope reading ($O > S$) where there is one boy whom all the girls like. As the number of QPs in a sentence increases, the number of distinct readings also increases.

Thus a simple sentence with three QPs admits of six possible readings, and in general a simple sentence with n QPs will be (at least) $n!$ ways ambiguous (we only consider readings where QPs are linearly ordered – what we will call asymmetric readings). In this paper, we will discuss three semantically distinct scope-assignment strategies:

Strategy A: Traditional movement strategy (Cooper 1983; May 1978; Montague 1973).

Strategy B: Polyadic approach (Keenan 1992, 1987; May 1985; Van Benthem 1989; Zawadowski 1989).

Strategy C: Continuation-based approach (Barker 2002; Barker and Shan 2014; Bekki and Asai 2009; De Groote 2001; Kiselyov and Shan 2014).

In all three strategies, QPs are interpreted as generalized quantifiers. A generalized quantifier on a set X is of type $\mathscr{C}(X) = (X \to \mathbf{t}) \to \mathbf{t}$ (with $\mathbf{t} = \{\mathbf{true}, \mathbf{false}\}$). The main difference between the three approaches lies in the semantic operations used to compute the truth-value of the relevant multi-quantifier sentences.

1.1 *Strategy A*

Strategy A has been implemented in various ways, using May's QR (1978), Montague's Quantifying In Rule (1973), or Cooper Storage (1983). In this strategy, the scope relations for multi-quantifier sentences like (1) are derived by applying quantifiers to the predicate (of type $\mathscr{P}(X \times Y) = (X \times Y) \to \mathbf{t}$) one by one – the later the quantifier is introduced, the wider its scope. In the terminology to be adopted in this paper, strategy A makes use of what we will call, after Mostowski, partial **mos**-operations:

$$\mathbf{mos}_Y : \mathscr{C}(Y) \times \mathscr{P}(X \times Y) \longrightarrow \mathscr{P}(X)$$

defined by a lambda term as:

$$\mathbf{mos}_Y = \lambda Q_{:\mathscr{C}(Y)}.\lambda c_{:\mathscr{P}(X\times Y)}.\lambda x_{:X}.Q(\lambda y_{:Y}.c(x,y));$$

and total **mos**-operations:

$$\mathbf{mos}_X : \mathscr{C}(X) \times \mathscr{P}(X) \longrightarrow \mathbf{t}$$

defined by a lambda term as:

$$\mathbf{mos}_X = \lambda Q_{:\mathscr{C}(X)}.\lambda c_{:\mathscr{P}(X)}.Q(c).$$

Strategy A can be straightforwardly extended to account for sentences involving three or more QPs (by allowing permutations of QPs).

1.2 *Strategy B*

Strategy B involving polyadic quantification was introduced and developed in the works of May (1985), Keenan (1987, 1992), Zawadowski (1989) and Van Benthem (1989). In this strategy, the scope relations for multi-quantifier sentences like (1) can be derived by turning a sequence of quantifiers into a polyadic quantifier, using what we will call left and right **pile′up**-operations (also known as iterations):

$$\mathbf{pile'up}^l, \mathbf{pile'up}^r : \mathscr{C}(X) \times \mathscr{C}(Y) \longrightarrow \mathscr{C}(X \times Y)$$

defined, for $M \in \mathscr{C}(X)$ and $N \in \mathscr{C}(Y)$, by lambda terms as:

$$\mathbf{pile'up}^l(M,N) = \lambda c_{:\mathscr{P}(X\times Y)}.M(\lambda x_{:X}.N(\lambda y_{:Y}c(x,y))$$

and

$$\mathbf{pile'up}^r(M,N) = \lambda c_{:\mathscr{P}(X\times Y)}.N(\lambda y_{:Y}.M(\lambda x_{:X}c(x,y)).$$

The polyadic quantifier thus formed is only then applied to the predicate. Again, strategy B can be straightforwardly extended to account for sentences involving three or more QPs (by allowing permutations of QPs).

1.3 *Strategy C*

Strategy C, the most recent, involves continuations and was first proposed in the works of Barker (2002) and De Groote (2001), and then further developed and modified in the works of Barker and Shan (2014), Kiselyov and Shan (2014) and Bekki and Asai (2009). Continuation-based strategies can be divided into two groups: those that locate the source of scope-ambiguity in the rules of semantic composition, and those that attribute it to the lexical entries for the quantifier words. In this paper, we consider only the first group: operation-based approaches (as in Barker 2002). In this strategy, a predicate gets lifted ('continuized'), i.e. a predicate of type $X \to \mathbf{t}$ will be lifted

to an expression of type $\mathscr{C}(X \to \mathbf{t}) = ((X \to \mathbf{t}) \to \mathbf{t}) \to \mathbf{t}$; etc. Scope relations for multi-quantifier sentences like (1) are derived by first combining the lifted predicate with the object QP, and then merging the result thus obtained with the subject QP, using the so-called **CPS** transforms:[1]

$$\mathbf{CPS}^{l}(ev), \mathbf{CPS}^{r}(ev) : \mathscr{C}(X) \times \mathscr{C}(X \to Y) \longrightarrow \mathscr{C}(Y)$$

given, for $M \in \mathscr{C}(X)$ and $N \in \mathscr{C}(X \to Y)$, by

$$\mathbf{CPS}^{l}(ev)(M,N) = \lambda c_{:\mathscr{P}(Y)}.M(\lambda x_{:X}.N(\lambda g_{:X\to Y}.c(g\ x)))$$

and

$$\mathbf{CPS}^{r}(ev)(M,N) = \lambda c_{:\mathscr{P}(Y)}.N(\lambda g_{:X\to Y}.M(\lambda x_{:X}.c(g\ x))).$$

Strategy C can be seen as a compelling alternative to the traditional movement strategy (Strategy A), and the polyadic approach (Strategy B), for a uniform non-movement (in situ) analysis of quantifiers. However, it cannot be straightforwardly extended to account for sentences involving three QPs.

As will be explained below, a generalized quantifier on a set X is an element of $\mathscr{C}(X)$, the value of the continuation monad $\mathscr{C}$ on X. In this paper, we will show that the continuation monad can be taken as a common basis for the three scope-assignment strategies just described. This will allow us to present these strategies against a uniform background and explicitly spell out the semantic operations used in each strategy. Two of the three strategies, B and C, use strength: the additional structure that exists on the continuation monad. This shows that the **pile′up** operations employed in the now widely accepted and well-understood strategy B, and the **CPS** operations employed in the less popular strategy C, considered more difficult, are in fact very close in spirit. We thus hope that our results will help to make the continuation-based strategy more popular.

The remaining part of this paper is organized as follows. We first introduce the notion of monad, starting with some informal remarks,

[1] In standard categorial grammar approaches, the scope relations for multi-quantifier sentences like (1) can be obtained via higher-order verb types (Hendriks 1993). For a comparison of standard type-shifting approaches and continuation-based strategies, see e.g. Barker and Shan (2014).

followed by a definition, and examples relevant to linguistics. We then introduce the continuation monad itself. Next, we define the notion of bi-strong monads, and show how the relevant algebraic operations (**pile′ups**, T-transforms and, in particular, **CPS**-transforms) are to be derived from strengths. Then we precisely state three specific implementations of the scope-assignment strategies: the traditional movement strategy (as implemented in May 1978), the polyadic approach (as in May 1985), and the continuation-based approach (as proposed in Barker 2002). With this background, we can explain how the three strategies are related, and what can be expected of them with regard to handling scopal ambiguities in simple sentences. An appendix contains the relevant proofs.

2 MONADS AND STRENGTHS

It is widely accepted that the notion of monad (also called 'triple') was first introduced in 1958 by Godement under the name of 'standard construction' (Godement 1958). It was soon realized that any pair of adjoint functors gives rise to a monad. Later, in 1965, it was discovered independently by Kleisli (1965) and by Eilenberg *et al.* (1965) that any monad is induced by an adjunction. For many years, the Eilenberg-Moore algebras were the most popular with mathematicians. It was Moggi who in 1989 used monads to build semantics for programming languages (Moggi 1991). Soon afterwards, Wadler employed monads to model side-effects in functional programming (Wadler 1990). In these new applications, the Kleisli algebras gained more importance. In both cases, monads are used to extend the notion of a function. After such a prelude, it did not take long for these ideas to be adopted in linguistics. The Kleisli construction can be thought of as an extension of a function/transformation f:

$$f : X \longrightarrow Y$$

between two sets, X and Y, which somehow reflects the fact that such a transformation is not considered as a mere mapping of arguments to values, but that there is also a particular computational process related to this association. This process, when applied to an element x of the domain set X, can indeed result in returning a value $f(x)$ of the codomain set Y. But it can also provide a more involved result

belonging to a set $T(Y)$, related in some way, but possibly bigger (or even much bigger) than Y itself. Thus we can think about such an extended[2] transformation between X and Y as a mere function:

$$f : X \longrightarrow T(Y)$$

from the set X to the extension $T(Y)$ of the set Y. This seems to be an intuitively clear and simple idea, but then we need to see whether we can still work with such 'extended functions' as we can with ordinary functions. The answer varies depending on how demanding we are. The minimum (which should be adequate for most purposes) that we should expect from these 'extended functions' is that they compose, that this composition should be associative, and that there should be 'extended functions' that act as if they were 'doing nothing' (i.e. like identities). Once we agree that these expectations are natural, we can try to specify the reasonable condition to guarantee this for construction T, i.e. that T should be a monad. Then the unit (return) $\eta_X : X \to T(X)$ acts as an identity on X, and the multiplication $\mu_Z : T^2(Z) \to T(Z)$, together with the fact that T is a functor, can be used to define the composition of the two 'extended functions', $f : X \longrightarrow T(Y)$ and $g : Y \to T(Z)$, as follows:

$$X \xrightarrow{\quad f \quad} T(Y) \xrightarrow{\quad T(g) \quad} T^2(Z) \xrightarrow{\quad \mu_Z \quad} T(Z)$$

The conditions imposed on T ensure that η_X is in fact the identity on X and that the composition thus defined is associative.

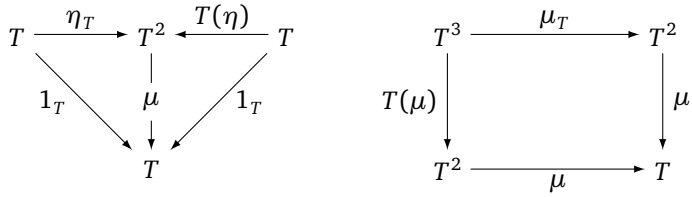
It is fair to say that the above is a short mathematician's introduction to monads, as used by computer scientists. In fact, the very notion of monad is usually formulated differently by computer scientists. This is, we think, due to the fact that the actual computation of the set $T(Y)$ even for the finite set Y can easily be infinite. This can be taken as a form of potential infinity. But then the second iteration $T(T(Y)) = T^2(Y)$, needed to express the multiplication μ, is even more challenging (since it requires applying the functor T to an already potentially complicated set $T(Y)$). The computer scientists' solution to this problem is to consider the combined operation **bind** : $T(X) \to (X \to T(Y)) \to T(Y)$ (instead of the multiplication μ),

[2] In some degenerate cases, the set $T(Y)$ might even be smaller than Y.

which never uses the second iteration of T. No matter how we define the monad T, the Kleisli category is the same. The potential gain from this extension for linguistics is that some processes which could not be described as compositional processes, applying (ordinary) functions to arguments, can become compositional after all, if we relax our notion of function to the 'extended function' we described above. The so-called continuation semantics for natural language, or 'strategy C' in this paper, is an illustration of such a phenomenon.

2.1 *Monads – definition and examples*

For unexplained notions related to category theory, we refer the reader to standard textbooks on the subject. We shall be exclusively working in the Cartesian closed category of sets *Set*. The category *Set* of sets has sets as objects. A morphism in *Set* from an object (set) X to an object (set) Y is a function[3] $f : X \longrightarrow Y$ from X to Y. A *monad* on *Set* is a triple (T, η, μ) where T : *Set* $\longrightarrow$ *Set* is an endofunctor (the underlying functor of the monad), $\eta : 1_{Set} \longrightarrow T$ and $\mu : T^2 \longrightarrow T$ are natural transformations (the first from the identity functor on *Set* to T, the second from the composition of T with itself to T) making the following diagrams commute:



These diagrams express the essence of the algebraic calculations. We shall explain their meaning while describing the list monad below. The symbols η and μ are often referred to as the *unit* and *multiplication* of the monad, respectively, while T is its functor part. When η and μ are clear from the context, it is customary to refer to the whole monad (T, η, μ) as T.

Before we focus on the continuation monad, the main notion of computation considered in this paper, we shall illustrate the concept

[3] We always consider functions with specified domains and codomains. For a pedantic reader, a function can be thought of as a triple $\langle X, Y, f \rangle$, such that X and Y are sets and f is a subset of the product $X \times Y$, which is total and univalued.

with some examples also relevant to linguistics (see e.g. Charlow 2014, and Shan 2002).

Examples of monads

1. The *Identity monad* is the simplest possible monad, but it is not very interesting. In this case, the functor T and the natural transformations η and μ are identities. For this monad, the notion of a T-computation in X is just an element of X, as the function $f : X \longrightarrow T(Y)$ is just $f : X \longrightarrow Y$.
2. The *Maybe monad* is the simplest non-trivial monad. The functor T sends every set X to the set $T(X) = X + \{\bot\}$ (the disjoint sum of X and singleton $\{\bot\}$), and every function $f : X \longrightarrow Y$ to a function $T(f) : T(X) \longrightarrow T(Y)$, such that, for $x \in T(X)$:

$$T(f)(x) = \begin{cases} x & \text{if } x \in X \\ \bot & \text{if } x = \bot \end{cases}$$

   So T adds to X an additional element $\bot$, called *bottom* or *nothing*. The component at X of the natural transformation η is a function $\eta_X : X \longrightarrow X + \{\bot\}$, such that $\eta_X(x) = x$, i.e. it sends x to the same x but in the set $X + \{\bot\}$. The component at X of the natural transformation μ is a function $\mu_X : X + \{\bot, \bot'\} \longrightarrow X + \{\bot\}$, such that, for $x \in X + \{\bot, \bot'\}$:

$$\mu_X(x) = \begin{cases} x & \text{if } x \in X \\ \bot & \text{if } x = \bot \text{ or } x = \bot' \end{cases}$$

   i.e. it sends x in X to the same x, and two bottoms $\bot$ and $\bot'$ in $T^2(X)$ to the only bottom $\bot$ in $T(X)$.
   For this monad, the notion of a T-computation in X consists of elements of X, and an additional computation $\bot$, which says that we do not get a value in X. The function $X \longrightarrow T(Y)$ carries the same information as a partial function $X \hookrightarrow Y$. So this monad allows partial computations to be treated as total computations.
3. The *Exception monad* is less trivial than the maybe monad. We are given a fixed set of exceptions E and, for a set X, the monad functor is $T(X) = X + E$, i.e. the disjoint union of X and E. If E is empty, it is the identity monad; if E is a singleton, then it is a maybe monad; otherwise is it like the maybe monad but with many options for nothingness.

4. The *List monad* or *monoid monad* is even more interesting than the previous monad, and we shall work it out in detail. It is not needed for the applications in the paper but it provides some insights before we move on to the continuation monad. To any set X, the list monad functor associates the set $T(X)$ of (finite) words over X (treated as an alphabet). This includes the empty word ε. To a function $f : X \longrightarrow Y$, the functor T associates the function $T(f) : T(X) \longrightarrow T(Y)$, sending the word $x_1x_2\ldots x_n$ over X to the word $f(x_1)f(x_2)\ldots f(x_n)$ over Y. The component at X of the natural transformation η is a function $\eta_X : X \longrightarrow T(X)$, such that $\eta_X(x) = x$, i.e. it sends the letter x to the one-letter word x in $T(X)$. The component at X of the natural transformation μ is a function $\mu_X : T^2(X) \longrightarrow T(X)$. Note that $T^2(X) = T(T(X))$ is the set of words whose letters are words over the alphabet X. Thus it can be thought of as a list of lists. Applying μ_X to such a list of lists flattens it to a single list. A three-letter word $t = (x_1x_2)(x_3x_4x_5)\varepsilon$ is a typical element of $T^2(X)$. The result of flattening T is the list $\mu_X(T) = x_1x_2x_3x_4x_5$ in $T(X)$. We can think of a word w as a term/word/computation $u = y_1y_2y_3$, in which we intend to substitute the term $v_1 = x_1x_2$ for variable y_1, the term $v_2 = x_3x_4x_5$ for variable y_2, and the term $v_3 = \varepsilon$ for variable y_3, i.e. $u[y_1\backslash v_1, y_2\backslash v_2, y_3\backslash v_3]$. Now the multiplication μ can be thought of as an actual substitution. With this interpretation, one can understand the intuitions behind the monad diagrams. In the left triangle, an element of $T(X)$, say $x_1x_2x_3$, is mapped through $\eta_{T(X)}$ to a single-letter word $(x_1x_2x_3)$ and μ_X flattens it back to $x_1x_2x_3$, as required for the triangle to commute. In other words, the substitution $y[y\backslash v]$ results in v. In the right triangle, the map $T(\eta_x)$ sends, say $x_1x_2x_3$, to the letter word $(x_1)(x_2)(x_2)$, with each letter being a single-letter word. Thus, again, flattening such a list gives $x_1x_2x_3$ back, as required. In other words, the substitution $y_1y_2y_3[y_1\backslash x_1, y_2\backslash x_2, y_3\backslash x_3]$ results in $x_1x_2x_3$. The commutation of the square diagram, in this case, expresses the fact that, if we have a list of lists of lists and we flatten it in two different ways, starting either with the upper two levels of lists, or with the lower two levels, and then we flatten the results again to get the ordinary lists over X in $T(X)$, these lists coincide. On a more conceptual level, this square expresses the fact that evaluation

commutes with substitution. In this sense, these diagrams capture the essence of all algebraic calculations.

For this monad, the notion of a T-computation in X consists of words over X to be evaluated/multiplied in a monoid when elements of X will be (interpreted) in a monoid. The function $f : X \longrightarrow T(Y)$ is just a function $f : X \longrightarrow T(Y)$ sending elements of X to words over Y. So this monad allows a list of values for a given input.

2.2 *Notation*

Before we explain the notion of computation that accompanies the continuation monad, we restate the monad in a more functional way. To do this, we need to introduce some form of notation. As *Set* is a Cartesian closed category, it is customary to denote functions between sets using λ notation. One can think of it as if we were to work in the internal language of *Set*, i.e. λ theory, where all functions have their names represented. For sets X and Y, we shall use $X \times Y$ to denote the binary product of X and Y, and $X \to Y$ to denote the set of functions from X to Y. As is customary, we associate $\to$ to the right, i.e. $X \to Y \to Z$ means $X \to (Y \to Z)$, and this set is naturally bijective with $(X \times Y) \to Z$. If we have a function:

$$f : X \times Y \longrightarrow Z,$$

then by:

$$\lambda y_{:Y}.f : X \longrightarrow Y \to Z$$

we denote its exponential adjunction, i.e. the function from X to the set of functions $Y \to Z$, such that, for an element $x \in X$, $\lambda y_{:Y}.f(x)$ is a function from Y to Z such that, for an element $y \in Y$, $(\lambda y_{:Y}.f)(x)(y)$ is by definition equal to $f(x, y)$. Note that, in the expression $(\lambda y_{:Y}.f)(x)(y)$, the first occurrence of y is an occurrence of a variable (as it is part of the name of a function), whereas the second occurrence of y in this expression denotes an element of the set Y.

Then π_i will denote the projection on i-component from the product. Any function $\sigma : \{1, \ldots m\} \longrightarrow \{1, \ldots, n\}$ induces a generalized projection denoted:

$$\pi_\sigma = \langle \pi_{\sigma(1)}, \ldots, \pi_{\sigma(m)} \rangle : X_1 \times \ldots \times X_n \longrightarrow X_{\sigma(1)} \times \ldots \times X_{\sigma(m)}.$$

We will use this notation mainly when σ is bijective, i.e. when π_σ is just a permutation of the component for the product.

We have a fixed set of truth values $\mathbf{t} = \{\mathbf{true}, \mathbf{false}\}$. We shall use the usual (possibly infinitary) operations on this set. For a set X, we write $\mathscr{P}(X) = X \to \mathbf{t}$, i.e. the (functional) powerset of X.

2.3 *Continuation monad*

The *Continuation monad*, the most important for us, is denoted $\mathscr{C}$. Its functor part (also denoted $\mathscr{C}$), at the level of objects, is just a twice-iterated power-set construction, i.e. for set X, $\mathscr{C}(X) = \mathscr{P}^2(X)$. At the level of morphisms, it is an inverse image of an inverse image, i.e., function $f : X \longrightarrow Y$ induces an inverse image function between powersets:

$$\mathscr{P}(f) = f^{-1} : \mathscr{P}(Y) \longrightarrow \mathscr{P}(X)$$
$$h \mapsto h \circ f,$$

in λ-notation,

$$\mathscr{P}(f) = \lambda h_{:\mathscr{P}(Y)}.\lambda x_{:X}.h(f\ x).$$

Taking again an inverse image function, we have

$$\mathscr{C}(f) = \mathscr{P}(f^{-1}) : \mathscr{C}(X) \longrightarrow \mathscr{C}(Y)$$

$$Q \mapsto Q \circ f^{-1},$$

in λ-notation:

$$\mathscr{C}(f)(Q) = \lambda h_{:\mathscr{P}(Y)}.Q(\lambda x_{:X}.h(f\ x)),$$

for $Q \in \mathscr{C}(X)$.

The unit $\eta_X : X \longrightarrow \mathscr{C}(X)$ is given by:

$$\eta_X(x) = \lambda h_{:\mathscr{P}(X)}.h(x), \qquad \text{for } x \in X.$$

The multiplication $\mu_X : \mathscr{C}^2(X) \longrightarrow \mathscr{C}(X)$ can be explained in terms of η:

$$\mu_X = \mathscr{P}(\eta_{\mathscr{P}(X)}) : \mathscr{P}^4(X) \longrightarrow \mathscr{P}^2(X).$$

In other words, $\mu_X(\mathscr{F}) : \mathscr{P}(X) \longrightarrow \mathbf{t}$ is a function such that:

$$\mu_X(\mathscr{F})(h) = \mathscr{F}(\eta_{\mathscr{P}(X)}(h))$$

for

$$\mathscr{F} : \mathscr{P}^3(X) \longrightarrow \mathbf{t} \quad \text{and} \quad h : X \longrightarrow \mathbf{t}.$$

In λ-notation, we write:

$$\mu_X(\mathscr{F})(h) = \mathscr{F}(\lambda D_{:\mathscr{C}(X)}.D(h)).$$

Now we can look at the notion of computation related the continuation monad. Consider the function:

$$f : X \longrightarrow \mathscr{C}(Y).$$

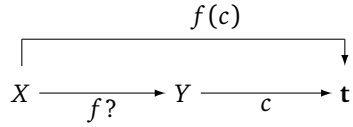
By exponential adjunction (uncurrying), it corresponds to a function:

$$f' : \mathscr{P}(Y) \times X \longrightarrow \mathbf{t}$$

and again, by exponential adjunction (currying), it corresponds to a function:

$$f'' : \mathscr{P}(Y) \longrightarrow \mathscr{P}(X).$$

Thus a $\mathscr{C}$-computation from X to Y is a function that sends functions from $\mathscr{P}(Y) = Y \to \mathbf{t}$ to functions in $\mathscr{P}(X)$. So instead of having for a given element $x \in X$ a direct answer to the question what is the value of f at x, i.e. the element $f(x)$ in Y, we are given for every continuation function $c : Y \longrightarrow \mathbf{t}$ a value in the answer type $\mathbf{t}$ that could be thought of as $c(f(x))$ (if there were an element in Y that could be reasonably called $f(x)$). We can draw a picture illustrating the situation:



Instead of 'procedure' $f?$ computing y's from x's (that we do not have), we provide a continuation $f(c)$ for any continuation (of the computation) c. If $f?$ were indeed a genuine function $f? : X \longrightarrow Y$, then $f(c)$ would be the composition $c \circ f?$.

2.4 *Bi-strong monads*

As noted in Moggi (1991), a monad has to be strong, in order to have a well-behaved notion of computation.[4] Fortunately, all monads on

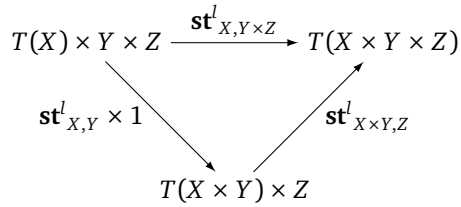
[4] As the notion of strength is new in this context, we shall briefly recall its history. There are three manifestations of strength on a functor. Historically, the

Set are strong. More precisely, all monads on *Set* can be canonically equipped with two strengths, left and right, and these strengths are compatible in a precise technical sense. This additional structure on the continuation monad will be essential when we analyze the meaning of multi-quantifier sentences.

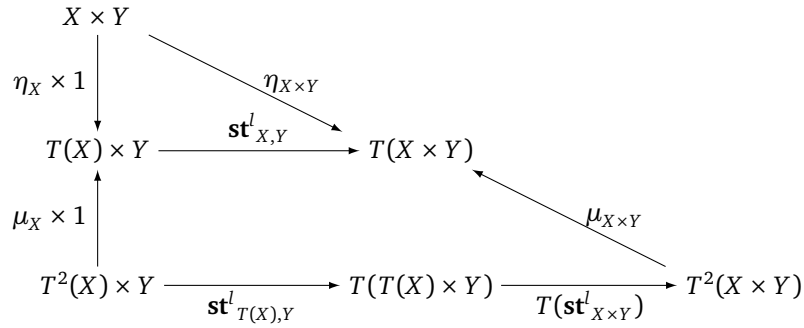
Let (T, η, μ) be a monad on *Set*. The *left strength* is a natural transformation with components:

$$\mathbf{st}^l{}_{X,Y} : T(X) \times Y \longrightarrow T(X \times Y)$$

for sets X and Y, making the following two diagrams commute:



and

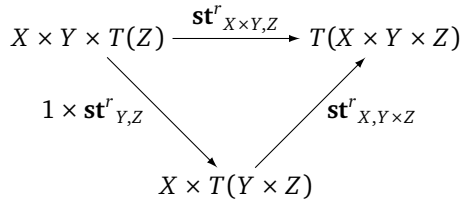


The *right strength* is a natural transformation with components:

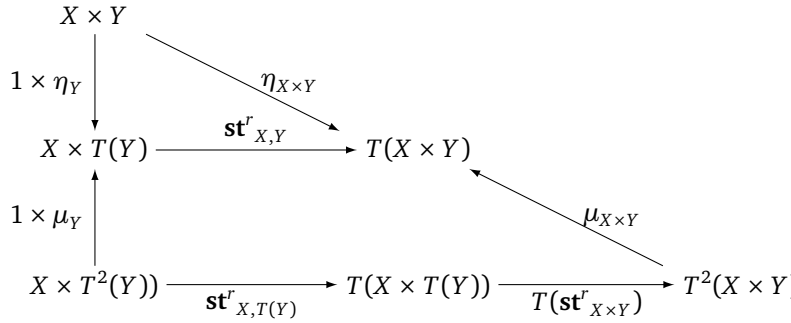
$$\mathbf{st}^r{}_{X,Y} : X \times T(Y) \longrightarrow T(X \times Y)$$

for sets X and Y, making the following two diagrams commute:

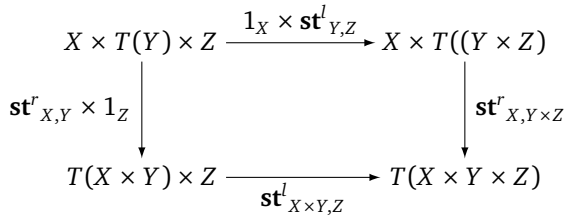
first one was the notion of enrichment of a functor (c.f. Eilenberg and Kelly 1966). Tensorial strength (i.e., natural transformation of type $X \otimes T(Y) \longrightarrow T(X \otimes Y)$ used in this paper) was introduced in Kock (1970) and further developed in Kock (1972). Cotensorial strength (i.e., natural transformation of type $T(X \to Y) \longrightarrow X \to T(Y)$) introduced in Kock (1971) has also proved useful in some contexts. In symmetric monoidal closed categories, these concepts are equivalent (c.f. Kock (1971)).



and



The monad (T, η, μ) on *Set* together with two natural transformations $\mathbf{st}^l$ and $\mathbf{st}^r$ of right and left strength is a *bi-strong monad* if, for any sets X, Y, Z, the following square commutes:



As we already mentioned, each monad (T, η, μ) on *Set* is bi-strong. We shall define the right and left strength. Fix sets X and Y. For $x \in X$ and $y \in Y$, we have functions:

$$l_y : X \longrightarrow X \times Y, \quad \text{and} \quad r_x : Y \longrightarrow X \times Y,$$

such that:

$$l_y(x) = \langle x, y \rangle, \quad \text{and} \quad r_x(y) = \langle x, y \rangle.$$

The left and right strength:

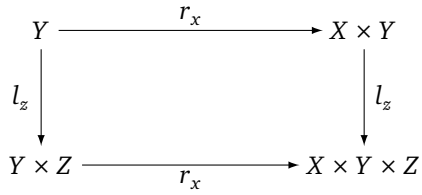
$$\mathbf{st}^l{}_{X,Y} : T(X) \times Y \longrightarrow T(X \times Y) \quad \text{and} \quad \mathbf{st}^r{}_{X,Y} : X \times T(Y) \longrightarrow T(X \times Y)$$

are given respectively for $x \in X$, $s \in T(X)$, $y \in Y$ and $t \in T(Y)$ by:

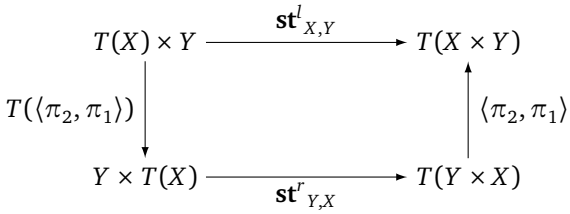
$$\mathbf{st}^l{}_{X,Y}(s, y) = T(l_y)(s) \quad \text{and} \quad \mathbf{st}^r{}_{X,Y}(x, t) = T(r_x)(t).$$

When it does not lead to confusion, we drop the indices $_{X,Y}$.

It is not difficult to verify that the above defines left ($\mathbf{st}^l$) and right ($\mathbf{st}^r$) strength on the monad T. Since for any $x \in X$ and $z \in Z$, the following square commutes:



they are compatible and make the monad T bi-strong. Note that these strengths are related by the following diagram:



Examples of strength on monads in Set

1. Maybe monad. The left strength $\mathbf{st}^l{}_{X,Y} : (X + \{\bot\}) \times Y \longrightarrow (X \times Y) + \{\bot\}$ is given by:

$$\mathbf{st}^l(x, y) = \begin{cases} \bot & \text{if } x = \bot \\ \langle x, y \rangle & \text{otherwise.} \end{cases}$$

   Right strength is similar.

2. List monad. The left strength $\mathbf{st}^l : T(X) \times Y \longrightarrow T(X \times Y)$ is given by:

$$\mathbf{st}^l(\vec{x}, y) = \begin{cases} \varepsilon & \text{if } \vec{x} = \varepsilon \\ \langle x_1, y \rangle, \ldots, \langle x_n, y \rangle & \text{if } \vec{x} = x_1, \ldots, x_n. \end{cases}$$

   Right strength is similar.

3. Continuation monad. We shall describe the strength morphisms by lambda terms. The left strength is:

$$\mathbf{st}^l = \lambda N_{:\mathscr{C}(X)}.\lambda y_{:Y}.\lambda c_{:\mathscr{P}(X\times Y)}.\\ N(\lambda x_{:X}.c(x,y)) : \mathscr{C}(X)\times Y \longrightarrow \mathscr{C}(X\times Y)$$

and the right strength is:

$$\mathbf{st}^r = \lambda x_{:X}.\lambda M_{:\mathscr{C}(Y)}.\lambda c_{:\mathscr{P}(X\times Y)}.\\ M(\lambda y_{:Y}.c(x,y)) : X\times \mathscr{C}(Y) \longrightarrow \mathscr{C}(X\times Y).$$

2.5 *Combining computations in arbitrary monad T on Set*

Using both strengths, we can define two **pile′up** natural transformations, left and right. For any sets X and Y, the *left pile up* $\mathbf{pile'up}^l{}_{X,Y}$ is defined from the diagram:

$$\begin{array}{ccc} T(X)\times T(Y) & \xrightarrow{\mathbf{pile'up}^l{}_{X,Y}} & T(X\times Y) \\ \Big\downarrow{\scriptstyle \mathbf{st}^l{}_{X,T(Y)}} & & \Big\uparrow{\scriptstyle \mu_{X\times Y}} \\ T(X\times T(Y)) & \xrightarrow[T(\mathbf{st}^r{}_{X,Y})]{} & T^2(X\times Y) \end{array}$$

In the above diagram, the function $\mathbf{pile'up}^l{}_{X,Y}$ is defined as a composition of three operations: the first takes the T-computation on X 'outside' to be a computation on $X\times T(Y)$, the second takes the T-computation on Y 'outside' to be a T-computation on $X\times Y$. In this way, we have T-computations coming from X on T-computations coming from Y on $X\times Y$. Now the last morphism $\mu_{X\times Y}$ flattens these two levels to one, i.e. the T-computation on T-computations to T-computations.

The *right pile up* $\mathbf{pile'up}^r{}_{X,Y}$ is defined from the diagram:

$$\begin{array}{ccc} T(X)\times T(Y) & \xrightarrow{\mathbf{pile'up}^r{}_{X,Y}} & T(X\times Y) \\ \Big\downarrow{\scriptstyle \mathbf{st}^r{}_{T(X),Y}} & & \Big\uparrow{\scriptstyle \mu_{X\times Y}} \\ T(T(X)\times Y) & \xrightarrow[T(\mathbf{st}^l{}_{X,Y})]{} & T^2(X\times Y) \end{array}$$

This operation takes the T-computations in reverse order and so they pile up in the opposite way.

If these **pile′up** operations agree for all sets X and Y, the monad is called *commutative*. In our list of monads, both the identity and maybe monads are commutative. The exception, list and continuation monads are not commutative. Most monads, including the continuation monad $\mathscr{C}$, are not commutative. It should be noted that even if the monad T is not commutative, both lift morphisms agree for pairs in which at least one component comes from the actual value (not an arbitrary T-computation). In other words, the functions:

$$T(X_1) \times T(X_2) \underset{\mathbf{pile'up}^r{}_{X_1,X_2}}{\overset{\mathbf{pile'up}^l{}_{X_1,X_2}}{\rightrightarrows}} T(X_1 \times X_2)$$

are equalized by both the following morphisms:

$$X_1 \times T(X_2) \xrightarrow{\eta_{X_1} \times 1} T(X_1) \times T(X_2)$$

and

$$T(X_1) \times X_2 \xrightarrow{1 \times \eta_{X_2}} T(X_1) \times T(X_2)$$

Both $\mathbf{pile'up}^l$ and $\mathbf{pile'up}^r$ are associative. All this is shown in the Appendix.

Examples of **pile′up***-operations*

1. Maybe monad. The left and right **pile′up**s coincide in this case, as in any commutative monad. We have

   $$\mathbf{pile'up}^l{}_{X,Y} = \mathbf{pile'up}^r{}_{X,Y} : (X + \{\bot\}) \times (Y + \{\bot'\}) \longrightarrow (X \times Y) + \{\bot\}$$

   given by:

   $$\mathbf{pile'up}^l(x,y) = \mathbf{pile'up}^r(x,y) = \begin{cases} \bot & \text{if } \{x,y\} \cap \{\bot,\bot'\} \neq \emptyset \\ \langle x,y \rangle & \text{otherwise.} \end{cases}$$

2. List monad. The left pile'up $\mathbf{pile'up}^l : T(X) \times T(Y) \longrightarrow T(X \times Y)$ is given by:

   $$\mathbf{pile'up}^l(\langle x_1 \ldots x_n \rangle, \langle y_1 \ldots y_m \rangle) =$$
   $$= \langle x_1, y_1 \rangle \langle x_1, y_2 \rangle \ldots \langle x_1, y_m \rangle \langle x_2, y_1 \rangle \ldots \langle x_n, y_{m-1} \rangle \langle x_n, y_m \rangle$$

   and the right pile'up $\mathbf{pile'up}^r : T(X) \times T(Y) \longrightarrow T(X \times Y)$ is given by:

   $$\mathbf{pile'up}^r(\langle x_1 \ldots x_n \rangle, \langle y_1 \ldots y_m \rangle) =$$
   $$= \langle x_1, y_1 \rangle \langle x_2, y_1 \rangle \ldots \langle x_n, y_1 \rangle \langle x_1, y_2 \rangle \ldots \langle x_{n-1}, y_m \rangle \langle x_n, y_m \rangle.$$

3. Continuation monad. Both **pile′up** operations:

$$\mathbf{pile'up}^{l}, \mathbf{pile'up}^{r} : \mathscr{C}(X) \times \mathscr{C}(Y) \longrightarrow \mathscr{C}(X \times Y)$$

can be defined, for $M \in \mathscr{C}(X)$ and $N \in \mathscr{C}(Y)$, by lambda terms, such as:

$$\mathbf{pile'up}^{l}(M,N) = \lambda c_{:\mathscr{P}(X\times Y)}.M(\lambda x_{:X}.N(\lambda y_{:Y}c(x,y))$$

and

$$\mathbf{pile'up}^{r}(M,N) = \lambda c_{:\mathscr{P}(X\times Y)}.N(\lambda y_{:Y}.M(\lambda x_{:X}c(x,y)).$$

The calculations for these operations are in the Appendix.
Thus in the case of the continuation monad, 'piling up' computations one on top of the other is nothing but putting (interpretations of) quantifiers (= computations in the continuation monad) in order, either the first before the second or the second before the first.

2.6 *T-transforms on arbitrary monad T on Set*

There are two (binary) T-transformations, right and left. For a function $f : X \times Y \longrightarrow T(Z)$, the left T-transform is defined as the composition:

$$\begin{array}{ccc} T(X)\times T(Y) & \xrightarrow{\mathbf{TR}^{l,T}{}_{X,Y}(f)} & T(Z) \\ \Big\downarrow{\scriptstyle \mathbf{pile'up}^{l}} & & \Big\uparrow{\scriptstyle \mu_Z} \\ T(X\times Y) & \xrightarrow[T(f)]{} & T^2(Z) \end{array}$$

and the right T-transform is defined as the composition:

$$\begin{array}{ccc} T(X)\times T(Y) & \xrightarrow{\mathbf{TR}^{r,T}{}_{X,Y}(f)} & T(Z) \\ \Big\downarrow{\scriptstyle \mathbf{pile'up}^{r}} & & \Big\uparrow{\scriptstyle \mu_Z} \\ T(X\times Y) & \xrightarrow[T(f)]{} & T^2(Z) \end{array}$$

The most popular T-transforms are for the evaluation morphism:

$$ev : X \times (X \to Y) \longrightarrow Y$$

but there are also other morphisms with useful transforms.

Examples of T-transforms and in particular **CPS***-transforms*

1. The evaluation map $ev : X \times (X \to Y) \longrightarrow Y$ gives rise to application transforms:

$$\mathbf{TR}^{l,T}(ev), \mathbf{TR}^{r,T}(ev) : T(X) \times T(X \to Y) \longrightarrow T(Y).$$

   When T is the continuation monad $\mathscr{C}$, they are the usual **CPS**-transforms $\mathbf{CPS}^{l}(ev), \mathbf{CPS}^{r}(ev) : \mathscr{C}(X) \times \mathscr{C}(X \to Y) \longrightarrow \mathscr{C}(Y)$ given by:

$$\mathbf{CPS}^{l}(ev)(M,N) = \lambda h_{:\mathscr{P}(Y)}.M(\lambda x_{:X}.N(\lambda g_{:X\to Y}.h(g\ x)))$$

   for $M \in \mathscr{C}(X)$ and $N \in \mathscr{C}(X \to Y)$.The right transform is similar.

2. Various evaluation maps are typically defined as maps from a product. Thus they give rise to various T-transforms. We list some of them below, mainly to introduce notation that will be used later. The definitions are given by lambda terms.
   (a) Left evaluation:

$$\mathbf{eps}^{l}{}_{X} = \lambda h_{:\mathscr{P}(X)}.\lambda x_{:X}.h(x) : \mathscr{P}(X) \times X \longrightarrow \mathbf{t};$$

   (b) Right evaluation:

$$\mathbf{eps}^{r}{}_{X} = \lambda x_{:X}.\lambda h_{:\mathscr{P}(X)}.h(x) : X \times \mathscr{P}(X) \longrightarrow \mathbf{t};$$

   (c) Left partial evaluation:

$$\mathbf{eps}^{l,X}_{Y}(\mathbf{eps}^{l}{}_{Y}) = \lambda c_{:\mathscr{P}(X\times Y)}.\lambda y_{:Y}.\lambda x_{:X}.c(x,y) : \mathscr{P}(X \times Y) \times Y \longrightarrow \mathscr{P}(X);$$

   (d) Right partial evaluation:

$$\mathbf{eps}^{r,X}_{Y}(\mathbf{eps}^{r}{}_{Y}) = \lambda y_{:Y}.\lambda c_{:\mathscr{P}(X\times Y)}.\lambda x_{:X}.c(x,y) : Y \times \mathscr{P}(X \times Y) \longrightarrow \mathscr{P}(X).$$

3. What we call Mostowski maps are maps similar to **eps**es that are the algebraic counterpart of the interpretation of generalized quantifiers by Mostowski. Again, we give a definition for total and partial case.
   (a) Left Mostowski:

$$\mathbf{mos}^{l}{}_{X} = \lambda Q_{:\mathscr{C}(X)}.\lambda c_{:\mathscr{P}(X)}.Q(c) : \mathscr{C}(X) \times \mathscr{P}(X) \longrightarrow \mathbf{t};$$

(b) Right Mostowski:

$$\mathbf{mos}^{r}{}_{X} = \lambda c_{:\mathscr{P}(X)}.\lambda Q_{:\mathscr{C}(X)}.Q(c) : \mathscr{P}(X) \times \mathscr{C}(X) \longrightarrow \mathbf{t};$$

(c) Left partial Mostowski:

$$\mathbf{mos}^{l}{}_{Y} = \lambda Q_{:\mathscr{C}(Y)}.\lambda c_{:\mathscr{P}(X\times Y)}.\lambda x_{:X}.Q(\lambda y_{:Y}.c(x,y)) : \mathscr{C}(Y) \times \mathscr{P}(X \times Y) \longrightarrow \mathscr{P}(X);$$

(d) Right partial Mostowski:

$$\mathbf{mos}^{r}{}_{Y} = \lambda c_{:\mathscr{P}(X\times Y)}.\lambda Q_{:\mathscr{C}(Y)}.\lambda x_{:X}.Q(\lambda y_{:Y}.c(x,y)) : \mathscr{P}(X \times Y) \times \mathscr{C}(Y) \longrightarrow \mathscr{P}(X).$$

3 SCOPE-ASSIGNMENT STRATEGIES

Using the notions connected to the continuation monad introduced above, we shall now precisely state and compare three strategies (A, B, and C) for determining the meaning of multi-quantifier sentences.

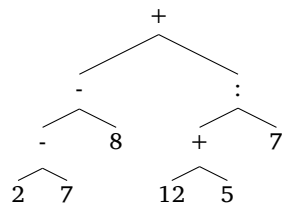
3.1 *General remarks*

In each strategy, the starting point is the surface structure tree of a sentence. This tree is rewritten so as to obtain formal structure trees that correspond to all the available meanings of the sentence. Finally, we relabel those trees to obtain computation trees[5] that provide the semantics for the sentence in each of its readings.

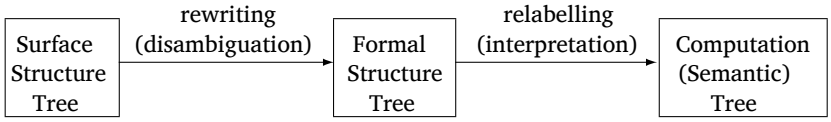
[5] We think of computation trees by analogy with mathematical expressions, e.g.

$$((2-7)-8)+((12+5):7)$$

that can be represented as:

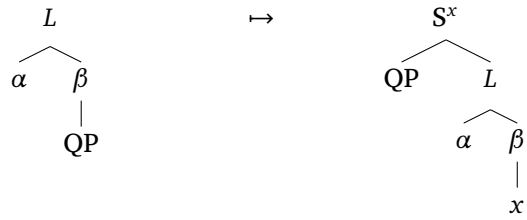


i.e. a labeled binary tree where the leaves are labeled with values and the internal nodes are labeled with operations that will be applied in the computation to the values obtained from the computations of the left and right subtrees.



Rewriting. Scope-assignment strategies can be divided into two families: movement analyses (rewriting rules include QR, Predicate Collapsing, and possibly Rotation) and in situ analyses (no rewriting rules). Below we define three rewrite rules on trees: QR Rule, Predicate Collapsing, and Rotation.

- QR (Quantifier Raising) Rule
  - applies when we have a chosen QP in a leaf of a tree;
  - adjoins QP to S;
  - indexes S with the variable bound by the raised QP.



  (L-label, α-subtree, β-subtree.)
- Predicate Collapsing
  - applies when all the leaves under the node labeled S are labeled with variables (not QPs);
  - collapses the whole subtree with the root S to a single leaf labeled with the variables x_1, x_2, x_3 from the leaves under the S-node.

- Rotation
  - applies to a tree with two nodes labeled with S'es superscripted with some variables: the mother labeled $S^{\vec{x}}$ and its right daughter labeled $S^{\vec{y}}$;

– it rotates left the subtree with the root labeled $S^{\vec{x}}$;
– the root of this subtree is labeled $S^{\vec{x}\vec{y}}$ and the (new) left daughter is labeled *Polyadic*.

(α-subtree, β-subtree, γ-subtree.)

Relabelling. In each scope-assignment strategy, the leaves in the computation tree have the same labels: QPs are interpreted as $\mathscr{C}$-computations, and predicates are interpreted as usual or lifted. The main difference between the three approaches consists in the shape of the formal structure trees and the operations (**eps**es, **mos**es, **pile′up**s, **CPS**es) used as labels for the inner nodes of the computation trees.

3.2 *Strategy A*

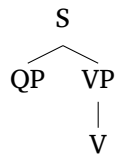
In the traditional movement strategy (as implemented in May 1978)

- Surface structure trees are rewritten (disambiguated) as formal structure trees (Logical Forms) via:
  – QR Rule;
  – Predicate Collapsing.
- Formal structure trees (LFs) are relabelled as computation trees as follows:
  – S^x (root of a subtree representing a formula) is interpreted as a suitably typed **mos**-operation (the only operation allowed);
  – S (leaf of a tree) is interpreted as a predicate;
  – QP (leaf of a tree) is interpreted as a generalized quantifier $\|Q\|$ quantifying over a set X (i.e. as a $\mathscr{C}$-computation on X).

We will illustrate each strategy with examples involving one, two and three QPs.

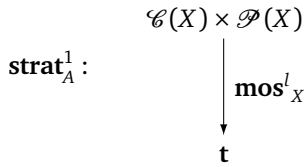
Sentence with one QP, e.g. *Every kid (most kids) entered.*

(A1) Surface structure tree:



(A1) Formal structure tree (LF) and the corresponding computation tree:

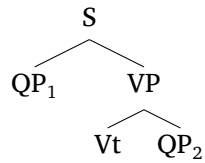
The computation tree in (A1) gives rise to the following general map:



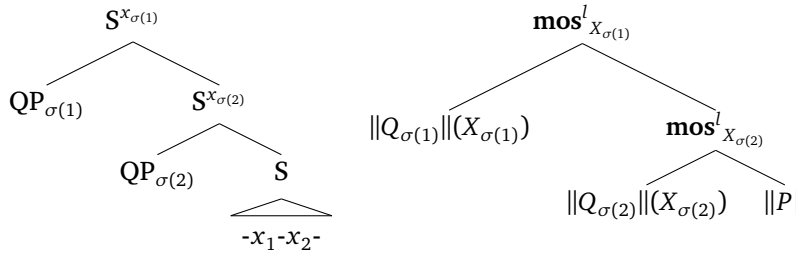
In this case, there is one such map, so strategy A yields one reading for a sentence with one QP.

Sentence with two QPs, e.g. *Every girl likes a boy*.

(A2) Surface structure tree:



(A2) Formal structure tree (LF) and the corresponding computation tree:



The computation tree in (A2) gives rise to the following general map, with $\sigma \in S_2$ (where S_2 is the set of permutations of the set $\{1,2\}$):

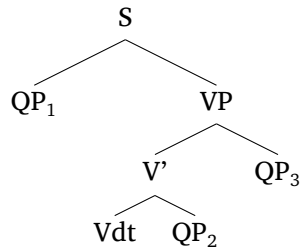
$$\begin{array}{ll} & \mathscr{C}(X_1) \times \mathscr{P}(X_1 \times X_2) \times \mathscr{C}(X_2) \xrightarrow{\bar{\pi}_{\sigma(i)}} \mathscr{C}(X_{\sigma(i)}) \\ \mathbf{strat}_A^{2,\sigma} : & \downarrow \langle \bar{\pi}_{\sigma(1)}, \bar{\pi}_{\sigma(2)}, \pi_2 \rangle \\ & \mathscr{C}(X_{\sigma(1)}) \times \mathscr{C}(X_{\sigma(2)}) \times \mathscr{P}(X_1 \times X_2) \\ & \downarrow 1 \times \mathbf{mos}^l{}_{X_{\sigma(2)}} \\ & \mathscr{C}(X_{\sigma(1)}) \times \mathscr{P}(X_{\sigma(1)}) \\ & \downarrow \mathbf{mos}^l{}_{X_{\sigma(1)}} \\ & \mathbf{t} \end{array}$$

where $\bar{\pi}_{\sigma(i)}$ is the projection on the 1st factor if $\sigma(i) = 1$, and on the 3rd factor if $\sigma(i) = 2$, i.e. as it should be. This convention will be used in all similar diagrams without any further explanations.

There are two such maps corresponding to the two permutations σ of $\{1,2\}$. These maps are different in general. Thus strategy A yields two (both) asymmetric readings for a sentence with two QPs.

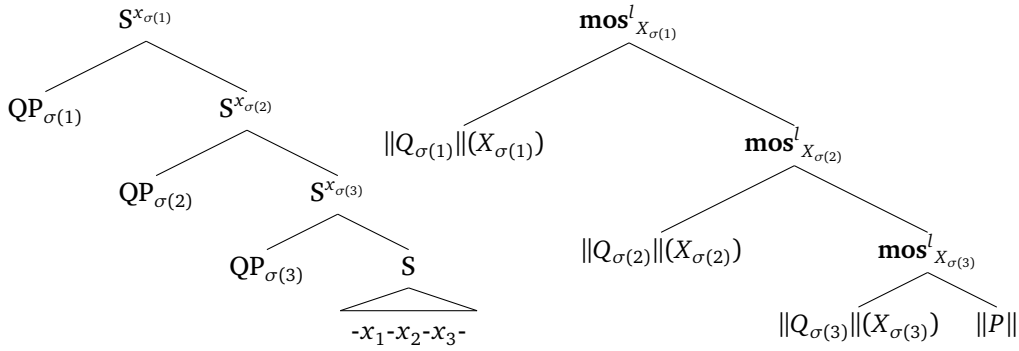
Sentence with three QPs, e.g. *Some teacher gave every student most books.*

(A3) Surface structure tree:[6]



[6] In this paper, we adopt the structure postulated by Chomsky (1993).

(A3) Formal structure tree (LF) and the corresponding computation tree:



The computation tree in (A3) gives rise to the following general map, with $\sigma \in S_3$ (where S_3 is the set of permutations of the set $\{1,2,3\}$):

$$\mathbf{strat}_A^{3,\sigma}:\quad \mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2\times X_3)\times\mathscr{C}(X_2)\times\mathscr{C}(X_3)$$
$$\downarrow\ \langle\bar{\pi}_{\sigma(1)},\bar{\pi}_{\sigma(2)},\bar{\pi}_{\sigma(3)},\pi_2\rangle$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)})\times\mathscr{C}(X_{\sigma(3)})\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow\ 1\times 1\times\mathbf{mos}^{l}{}_{X_{\sigma(3)}}$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)})\times\mathscr{P}(\ldots\times\widehat{X_{\sigma(3)}}\times\ldots)$$
$$\downarrow\ 1\times\mathbf{mos}^{l}{}_{X_{\sigma(2)}}$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{P}(X_{\sigma(1)})$$
$$\downarrow\ \mathbf{mos}^{l}{}_{X_{\sigma(1)}}$$
$$\mathbf{t}$$

There are six such maps corresponding to the six permutations σ of $\{1,2,3\}$. These maps are different in general. Thus strategy A yields 6 asymmetric readings for a sentence with three QPs.

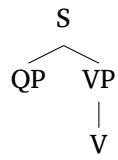
3.3 *Strategy B*

In the polyadic approach (as implemented in May 1985):

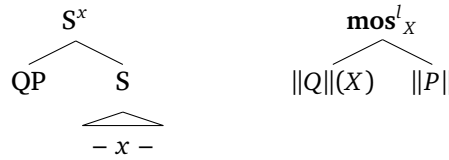
- Surface structure trees are rewritten (disambiguated) as formal structure trees (Polyadic Logical Forms) via:
  - QR Rule;
  - Predicate Collapsing;
  - Rotation.
- Formal structure trees (PLFs) are relabelled as computation trees as follows:
  - *Polyadic* (root of a subtree representing a polyadic quantifier) is interpreted as a suitably typed **pile′up**-operation (we can choose whether to use only $\mathbf{pile'up}^l$ or $\mathbf{pile'up}^r$ and then stick to that decision).
  - S^x, S, QP are interpreted as above.

Sentence with one QP, e.g. *Every kid (most kids) entered.*

(B1) Surface structure tree:



(B1) Formal structure tree (PLF) and the corresponding computation tree:



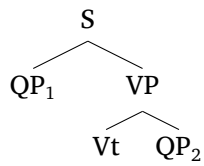
The computation tree in (B1) gives rise to the following general map:

$$\mathbf{strat}^1_B : \mathscr{C}(X) \times \mathscr{P}(X) \xrightarrow{\mathbf{mos}^l{}_X} \mathbf{t}$$

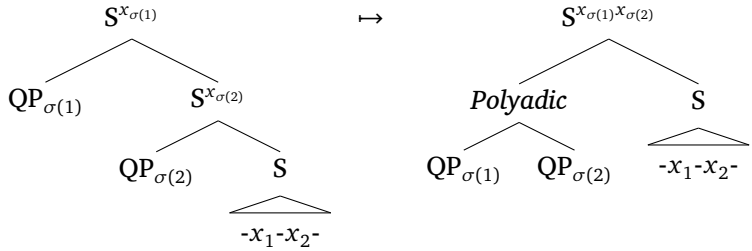
In this case, there is one such map, so strategy B yields one reading for a sentence with one QP.

Sentence with two QPs, e.g. *Every girl likes a boy*.

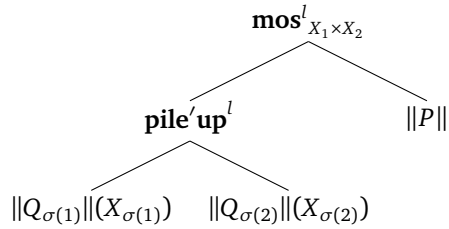
(B2) Surface structure tree:



(B2) Formal structure tree (PLF) obtained from LF in (A2) via rotation:



and the corresponding computation tree:



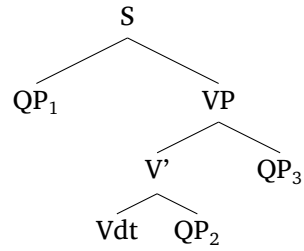
The computation tree in (B2) gives rise to the following general map, with $\sigma \in S_2$:

$$\mathbf{strat}_B^{2,\sigma}:\quad \begin{array}{c} \mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2)\times\mathscr{C}(X_2)\\ \downarrow \langle \bar{\pi}_{\sigma(1)}, \bar{\pi}_{\sigma(2)}, \pi_2\rangle\\ \mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)})\times\mathscr{P}(X_1\times X_2)\\ \downarrow \mathbf{pile}'\mathbf{up}^l\times 1\\ \mathscr{C}(X_{\sigma(1)}\times X_{\sigma(2)})\times\mathscr{P}(X_1\times X_2)\\ \downarrow \mathscr{C}(\pi_{\sigma^{-1}})\times 1\\ \mathscr{C}(X_1\times X_2)\times\mathscr{P}(X_1\times X_2)\\ \downarrow \mathbf{mos}^l{}_{X_1\times X_2}\\ \mathbf{t}\end{array}$$

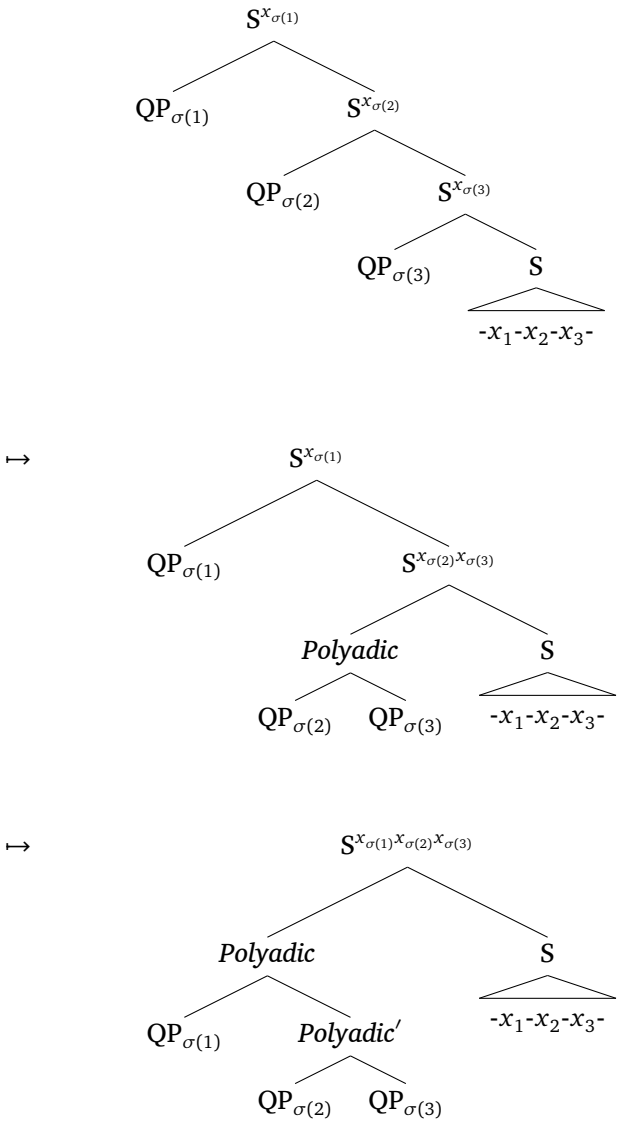
There are two such maps, corresponding to the two permutations σ of $\{1,2\}$ combined with a $\mathbf{pile}'\mathbf{up}^l$-operation (here, we can also choose to use both **pile′ups** instead and no permutations at all). These maps are different in general. Thus strategy B yields two (both) asymmetric readings for a sentence with two QPs.

Sentence with three QPs, e.g. *Some teacher gave every student most books.*

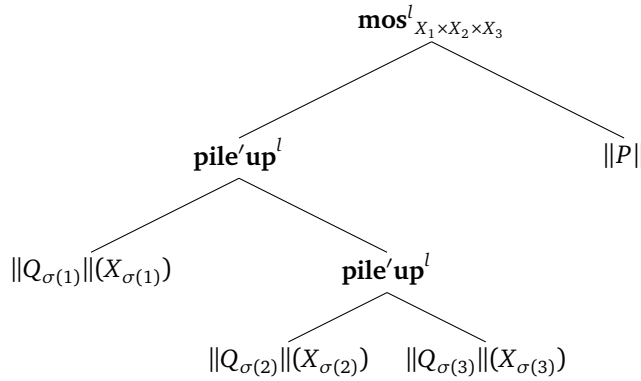
(B3) Surface structure tree:



(B3) Formal structure tree (PLF) obtained from LF in (A3) via rotation:



and the corresponding computation tree:



The computation tree in (B3) gives rise to the following general map, with $\sigma \in S_3$:

$$\mathbf{strat}_B^{3,\sigma}:$$

$$\mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2\times X_3)\times\mathscr{C}(X_2)\times\mathscr{C}(X_3)$$
$$\downarrow \langle \bar{\pi}_{\sigma(1)}, \bar{\pi}_{\sigma(2)}, \bar{\pi}_{\sigma(3)}, \pi_2\rangle$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)})\times\mathscr{C}(X_{\sigma(3)})\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow 1\times \mathbf{pile'up}^{l}\times 1$$
$$\mathscr{C}(X_{\sigma(1)})\times\mathscr{C}(X_{\sigma(2)}\times X_{\sigma(3)})\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow \mathbf{pile'up}^{l}\times 1$$
$$\mathscr{C}(X_{\sigma(1)}\times X_{\sigma(2)}\times X_{\sigma(3)})\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow \mathscr{C}(\pi_{\sigma^{-1}})\times 1$$
$$\mathscr{C}(X_1\times X_2\times X_3)\times\mathscr{P}(X_1\times X_2\times X_3)$$
$$\downarrow \mathbf{mos}^{l}{}_{X_1\times X_2\times X_3}$$
$$\mathbf{t}$$

There are six such maps, corresponding to the six permutations σ of $\{1,2,3\}$ combined with a $\mathbf{pile}'\mathbf{up}^{l}$-operation (here, we can also choose to use $\mathbf{pile}'\mathbf{up}^{r}$ instead). These maps are different in general. Thus strategy B yields 6 asymmetric readings for a sentence with three QPs.

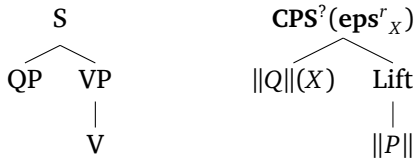
3.4 *Strategy C*

In the continuation-based strategy approach (as proposed in Barker 2002):

- A surface structure tree is rewritten as a formal structure tree via:
  - no rewriting rules (formal structure trees are just surface structure trees – this is what is understood by in situ).
- Relabelling formal structure trees (= surface structure trees) as computation trees follows this procedure:
  - S, VP, V' (roots of a (sub)tree with some (possibly all) arguments provided) are interpreted as suitably typed **CPS**-operations (left and right);
  - V, Vt, Vdt (leaves of a tree) are interpreted as 'continuized' predicates (1-, 2-, 3-ary, respectively).

Sentence with one QP, e.g. *Every kid (most kids) entered.*

(C1) Surface structure tree and the corresponding computation tree:



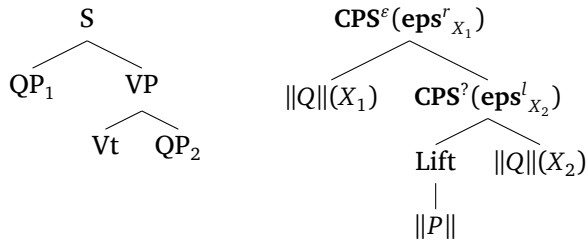
The computation tree in (C1) gives rise to the following general map:

$$\mathbf{strat}^{1}_{c}: \quad \begin{array}{c} \mathscr{C}(X)\times\mathscr{P}(X) \\ \downarrow{\scriptstyle 1\times\eta_{\mathscr{P}(X)}} \\ \mathscr{C}(X)\times\mathscr{C}\mathscr{P}(X) \\ \downarrow{\scriptstyle \mathbf{CPS}^{?}(\mathbf{eps}^{r}{}_{X})} \\ \mathscr{C}(\mathbf{t}) \end{array} \xrightarrow[ev_{id_{\mathbf{t}}}]{} \mathbf{t}$$

We use $\mathbf{CPS}^{?}$ when it does not matter whether we apply $\mathbf{CPS}^{l}$ or $\mathbf{CPS}^{r}$. This is the case when one of the arguments is a lifted element (like interpretations of predicates in this strategy). Strategy C yields one reading for a sentence with one QP.

Sentence with two QPs, e.g. *Every girl likes a boy.*

(C2) Surface structure tree and the corresponding computation tree:



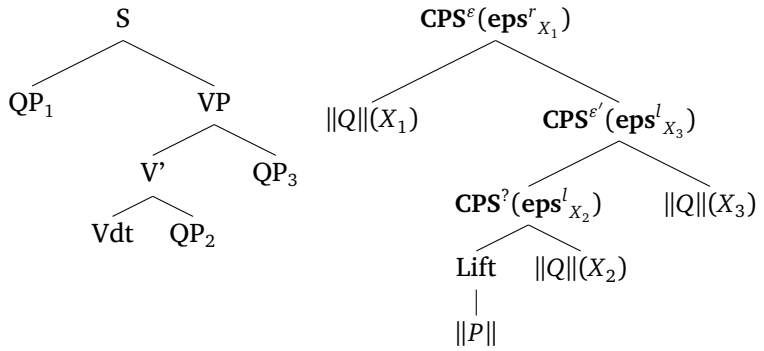
The computation tree in (C2) gives rise to the following general map:

$$\mathbf{strat}_C^{2,\varepsilon}:\quad \begin{array}{c} \mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2)\times\mathscr{C}(X_2) \\ \downarrow\ 1\times\eta_{\mathscr{P}(X_1\times X_2)}\times 1 \\ \mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1\times X_2)\times\mathscr{C}(X_2) \\ \downarrow\ 1\times\mathbf{CPS}^{?}(\mathbf{eps}^{l}{}_{X_2}) \\ \mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1) \\ \downarrow\ \mathbf{CPS}^{\varepsilon}(\mathbf{eps}^{r}{}_{X_1}) \\ \mathscr{C}(\mathbf{t}) \xrightarrow[ev_{id_{\mathbf{t}}}]{} \mathbf{t} \end{array}$$

with $\varepsilon \in \{l, r\}$. Depending on whether we use $\mathbf{CPS}^{l}$ or $\mathbf{CPS}^{r}$, we get the relevant one of the two asymmetric readings for a sentence with two QPs. Strategy C yields two readings for a sentence with two QPs, corresponding to the two forms of **CPS**.

Sentence with three QPs, e.g. *Some teacher gave every student most books.*

(C3) Surface structure tree and the corresponding computation tree:



The computation tree in (C3) gives rise to the following general map:

$$\mathbf{strat}_C^{3,\varepsilon',\varepsilon}:\quad \begin{array}{c} \mathscr{C}(X_1)\times\mathscr{P}(X_1\times X_2\times X_3)\times\mathscr{C}(X_2)\times\mathscr{C}(X_3) \\ \downarrow\ 1\times\eta_{\mathscr{P}(X_1\times X_2\times X_3)}\times 1\times 1 \\ \mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1\times X_2\times X_3)\times\mathscr{C}(X_2)\times\mathscr{C}(X_3) \\ \downarrow\ 1\times\mathbf{CPS}^{?}(\mathbf{eps}^{l}{}_{X_2})\times 1 \\ \mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1\times X_3)\times\mathscr{C}(X_3) \\ \downarrow\ 1\times\mathbf{CPS}^{\varepsilon'}(\mathbf{eps}^{l}{}_{X_3}) \\ \mathscr{C}(X_1)\times\mathscr{C}\mathscr{P}(X_1) \\ \downarrow\ \mathbf{CPS}^{\varepsilon}(\mathbf{eps}^{r}{}_{X_1}) \\ \mathscr{C}(\mathbf{t}) \xrightarrow[ev_{id_{\mathbf{t}}}]{} \mathbf{t} \end{array}$$

Strategy C provides four asymmetric readings for the sentence, such that QP in subject position can be placed either first or last only

(corresponding to the four possible combinations of the two forms of **CPS**). Thus it yields four out of the six readings accounted for by strategies A and B. Of course, it is a matter of empirical discovery which readings are available for such sentences, and the status of the two missing 'interleaved' interpretations (*every* > *some* > *most* and *most* > *some* > *every*) is still under discussion.

The tables below summarize the main features of the three approaches.

Passing from Surface Structure Trees to Formal Structure Trees

Strategy	***A***	***B***	***C***
Rewrite rules	*QR, Predicate Collapsing*	*QR, Predicate Collapsing, Rotation*	*No rewrite rules (in situ)*

Passing from Formal Structure Trees to Computation Trees

Strategy	***A***	***B***	***C***
Relabelling inner nodes	$S^x \mapsto$ **mos**	$S^{\vec{x}} \mapsto$ **mos** *Polyadic* $\mapsto$ **pile′up**	$S, VP, V' \mapsto$ **CPS**
Relabelling leaves	$S \mapsto$ *relation* $QP \mapsto$ *𝒞-comp.*	$S \mapsto$ *relation* $QP \mapsto$ *𝒞-comp.*	$V,\ Vt,\ Vdt \mapsto$ *continuized relation* $QP \mapsto$ *𝒞-comp.*

The semantics for sentences with intransitive or transitive verbs, as defined by strategies A, B, and C, will be equivalent. The semantics for sentences with ditransitive verbs, as defined by strategies A, and B, will be equivalent, providing six asymmetric readings for the sentence. The semantics for sentences with ditransitive verbs, as defined by strategy C, will provide four asymmetric readings for the sentence, such that QP in subject position can be placed either first or last only, corresponding to four out of the six readings accounted for by strategies A and B. The proofs are given in the Appendix.

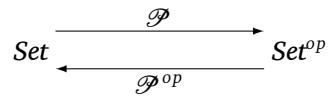
4 CONCLUSIONS AND FUTURE WORK

We compared three scope-assignment strategies for simple multi-quantifier sentences: the traditional movement strategy, the polyadic approach, and the continuation-based approach. These strategies can be viewed as instances of the same general pattern: first transform the SS-tree of a sentence so as to obtain the shape of the computation tree, then relabel the leaves of that tree, with the interpretation of lexical items (predicate and QPs), and the inner nodes, using algebraic operations, and finally evaluate this computation in order to get the truth value of the whole sentence. We have shown that while the traditional movement strategy is very close to the original semantics for the logic with generalized quantifiers due to Mostowski, the polyadic approach and the continuation-based strategy are in fact cognate in spirit, as they can both be defined using operations derived from the strength of the continuation monad. As the polyadic strategy is well-understood among linguists, we hope that our results will help to make the continuation-based strategy more popular. With the continuation monad as a common basis for the three scope-assignment strategies discussed, it is also easy to identify their relative merits and weaknesses. Traditional and polyadic approaches cannot provide a non-movement (in situ) analysis of quantifiers. The continuation-based strategy is in situ but does not account for all the asymmetric readings possible for sentences involving three QPs. As discussed in Bekki and Asai (2009) and proved in this paper, it only provides four out of the six readings possible for such sentences. In the sequel to this paper, we show how to overcome this problem, keeping the resulting strategy in situ (Grudzinska and Zawadowski 2016). We take the results of this work to be the first step towards an in situ semantics that will be sufficient to account for the whole range of possible readings for multi-quantifier sentences.

5 APPENDIX

5.1 *The continuation monad*

In this subsection, we gather all the basic facts (sometimes repeated from the text) of the *continuation monad* $\mathscr{C}$ on *Set*. We have an adjunction:



where both $\mathscr{P}$ and $\mathscr{P}^{op}$ are the contravariant powerset functors[7] with the domains and codomains as displayed. In particular, for $f : X \longrightarrow Y$, the function $\mathscr{P}(f) = f^{-1} : \mathscr{P}(Y) \longrightarrow \mathscr{P}(X)$ is given by:

$$f^{-1}(h) = h \circ f$$

for $h : Y \longrightarrow \mathbf{t}$. Function $\eta_X : X \longrightarrow \mathscr{C}(X)$, the component at set X of the unit of this adjunction $\eta : 1_{Set} \longrightarrow \mathscr{P}\mathscr{P}^{op} = \mathscr{C}$, is given by:

$$\eta_X(x) = \lambda h_{:\mathscr{P}(X)}.h(x).$$

Function $\varepsilon_X : X \longrightarrow \mathscr{C}(X)$, the component at set X of the co-unit of this adjunction $\varepsilon : 1_{Set} \longrightarrow \mathscr{P}^{op}\mathscr{P}$, is given by (essentially the same formula):

$$\varepsilon_X(x) = \lambda h_{:\mathscr{P}^{op}(X)}.h(x)$$

for $x \in X$. The function $\mathscr{C}(f) : \mathscr{C}(X) \longrightarrow \mathscr{C}(Y)$, for $Q : \mathscr{P}(X) \longrightarrow \mathbf{t} \in \mathscr{C}(X)$, is a function $\mathscr{C}(f)(Q) : \mathscr{P}(Y) \longrightarrow \mathbf{t}$ given by:

$$\mathscr{C}(f)(Q)(h) = Q(h \circ f)$$

for $h : Y \longrightarrow \mathbf{t}$.

The monad induced by this adjunction is the continuation monad. Its multiplication is given by the co-unit of the above adjunction transported back to *Set*, i.e. $\mu = \mathscr{P}^{op}(\varepsilon_{\mathscr{P}})$. For X in *Set*, the function

$$\mu_X : \mathscr{C}^2(X) \longrightarrow \mathscr{C}(X)$$

is given by:

$$\mu_X(\mathscr{R}) = \mathscr{R} \circ \eta_{\mathscr{P}(X)} \qquad \text{for} \quad \mathscr{R} \in \mathscr{C}^2(X).$$

In λ-notation we write:

$$\mu_X(\mathscr{F})(h) = \mathscr{F}(\lambda D_{:\mathscr{C}(X)}.D(h)).$$

[7] Note that this is in contrast with the functor $\mathscr{P}$, where $\mathscr{P}$ is the covariant power-set functor.

The left strength for the monad $\mathscr{C}$ is:

$$\mathbf{st}^l : \mathscr{C}(X) \times Y \longrightarrow \mathscr{C}(X \times Y)$$

for $M \in \mathscr{C}(X)$ and $y \in Y$, given by:

$$\mathbf{st}^l(M, y) = \lambda c_{:\mathscr{P}(X\times Y)}.M(\lambda x_{:X}.c(x, y)) : \mathscr{P}(X \times Y) \longrightarrow \mathbf{t}$$

and the right strength, for $x \in X$ and $N \in \mathscr{C}(Y)$, is given by:

$$\mathbf{st}^r(x, N) = \lambda c_{:\mathscr{P}(X\times Y)}.N(\lambda y_{:Y}.c(x, y)) : \mathscr{P}(X \times Y) \longrightarrow \mathbf{t}.$$

The left pile'up operation:

$$\mathbf{pile'up}^l : \mathscr{C}(X) \times \mathscr{C}(Y) \longrightarrow \mathscr{C}(X \times Y)$$

is the following composition:

$$\mathscr{C}(X) \times \mathscr{C}(Y) \xrightarrow{\mathbf{st}^l} \mathscr{C}(X \times \mathscr{C}(Y)) \xrightarrow{\mathscr{C}(\mathbf{st}^r)} \mathscr{C}^2(X \times Y) \xrightarrow{\mu_{X\times Y}} \mathscr{C}(X \times Y)$$

where, for $Q \in \mathscr{C}(X)$, $Q' \in \mathscr{C}(Y)$, $c \in \mathscr{P}(X \times \mathscr{C}(Y))$, we have:

$$\mathbf{st}^l(Q, Q')(c) = Q(\lambda x_{:X} c(x, Q))$$

and, for $d \in \mathscr{C}(X \times \mathscr{C}(Y))$, $\mathscr{U} \in \mathscr{P}\mathscr{C}(X \times Y)$, we have:

$$\mathscr{C}(\mathbf{st}^r)(d)(\mathscr{U}) = d(\mathscr{U} \circ \mathbf{st}^r).$$

Now, using the above formulas, we can calculate $\mathbf{pile'up}^l$ as the composition on $Q \in \mathscr{C}(X)$, $Q' \in \mathscr{C}(Y)$, and $c \in \mathscr{P}(X \times Y)$ as follows:

$$\begin{aligned}
\mathbf{pile'up}^l(Q, Q')(c) &= \mu_{X\times Y}(\mathscr{C}(\mathbf{st}^r)(\mathbf{st}^l(Q, Q')))(c) \\
&= \mathscr{C}(\mathbf{st}^r)(\mathbf{st}^l(Q, Q'))(\lambda D_{:\mathscr{C}(X\times Y)} D(c)) \\
&= \mathbf{st}^l(Q, Q')((\lambda D_{:\mathscr{C}(X\times Y)} D(c)) \circ \mathbf{st}^r) \\
&= Q(\lambda x_{:X}((\lambda D_{:\mathscr{C}(X\times Y)} D(c)) \circ \mathbf{st}^r)(x, Q')) \\
&= Q(\lambda x_{:X}((\lambda D_{:\mathscr{C}(X\times Y)} D(c))(\mathbf{st}^r(x, Q')) \\
&= Q(\lambda x_{:X}\mathbf{st}^r(x, Q')(c)) \\
&= Q(\lambda x_{:X} Q'(\lambda y_{:Y} c(x, y)))
\end{aligned}$$

Similarly, we can show that:

$$\mathbf{pile'up}^r(Q,Q')(c) = Q'(\lambda y_{:Y} Q(\lambda x_{:X} c(x,y))).$$

One can easily verify that **pile′up**'s are related by:

$$\mathbf{pile'up}^r{}_{X,Y} = \mathscr{C}(\pi_{(2,1)}) \circ \mathbf{pile'up}^l{}_{Y,X} \circ \pi_{(2,1)}.$$

5.2 *Some properties of* **pile′up** *operations*

Lemma 5.1 (pile′up lemma) **pile′up***s on pairs where one element is continuized agree and are equal to the corresponding strength.*

Proof. We have to show that the functions:

$$T(X_1) \times T(X_2) \underset{\mathbf{pile'up}^r{}_{X_1,X_2}}{\overset{\mathbf{pile'up}^l{}_{X_1,X_2}}{\rightrightarrows}} T(X_1 \times X_2)$$

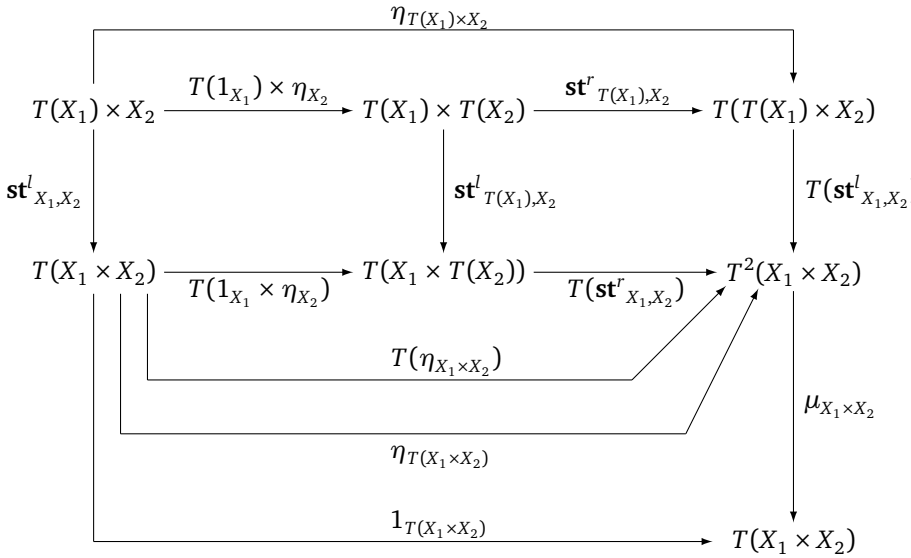
are equalized by both:

$$X_1 \times T(X_2) \xrightarrow{\eta_{X_1} \times T(1_{X_2})} T(X_1) \times T(X_2)$$

and

$$T(X_1) \times X_2 \xrightarrow{T(1_{X_1}) \times \eta_{X_2}} T(X_1) \times T(X_2)$$

and their composition with these functions is equal to strength morphisms. Using the diagram:



we shall show that:

$$\mathbf{pile}'\mathbf{up}^r{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) = \mathbf{st}^l{}_{X_1,X_2} = \mathbf{pile}'\mathbf{up}^l{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}).$$

The other cases are symmetric. We have:

$$\mathbf{pile}'\mathbf{up}^r{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) = \quad (\text{def of } \mathbf{pile}'\mathbf{up}^r)$$

$$= \mu_{X_1 \times X_2} \circ T(\mathbf{st}^l{}_{X_1,X_2}) \circ \mathbf{st}^r{}_{T(X_1),X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) \quad (\eta \text{ strong w.r.t. } \mathbf{st}^r)$$

$$= \mu_{X_1 \times X_2} \circ T(\mathbf{st}^l{}_{X_1,X_2}) \circ \eta_{T(X_1) \times X_2} \quad (\eta \text{ nat transf})$$

$$= \mu_{X_1 \times X_2} \circ \eta_{T(X_1 \times X_2)} \circ \mathbf{st}^l{}_{X_1,X_2} \quad (T \text{ monad})$$

$$= \mathbf{st}^l{}_{X_1,X_2}$$

To show the remaining equation, we can continue the penultimate formula above as follows:

$$\mathbf{pile}'\mathbf{up}^r{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) = \ldots = \mu_{X_1 \times X_2} \circ \eta_{T(X_1 \times X_2)}) \circ \mathbf{st}^l{}_{X_1,X_2}$$

$$= \quad (T \text{ monad})$$

$$= \mu_{X_1 \times X_2} \circ T(\eta_{X_1 \times X_2}) \circ \mathbf{st}^l{}_{X_1,X_2} \quad (\eta \text{ strong w.r.t. } \mathbf{st}^r)$$

$$= \mu_{X_1 \times X_2} \circ T(\mathbf{st}^r{}_{X_1,X_2}) \circ T(1_{X_1} \times \eta_{X_2}) \circ \mathbf{st}^l{}_{X_1,X_2} \quad (\mathbf{st}^l \text{ nat transf})$$

$$= \mu_{X_1 \times X_2} \circ T(\mathbf{st}^r{}_{X_1,X_2}) \circ \mathbf{st}^l{}_{X_1,X_2} \circ T(1_{X_1} \times \eta_{X_2}) \quad (\text{def of } \mathbf{pile}'\mathbf{up}^l)$$

$$= \mathbf{pile}'\mathbf{up}^l{}_{X_1,X_2} \circ (T(1_{X_1}) \times \eta_{X_2}) \qquad \diamondsuit$$

Corollary 5.2 *The left and right* **CPS***-operation on pairs where one element is continuized agree.*

Proof. The corollary states that, for any sets X, Y, Z and a function $f : X \times Y \longrightarrow Z$, both morphisms:

$$X \times T(Y) \xrightarrow{\eta_X \times 1} T(X) \times T(Y)$$

and

$$T(X) \times Y \xrightarrow{1 \times \eta_Y} T(X) \times T(Y)$$

equalize the pair of morphisms:

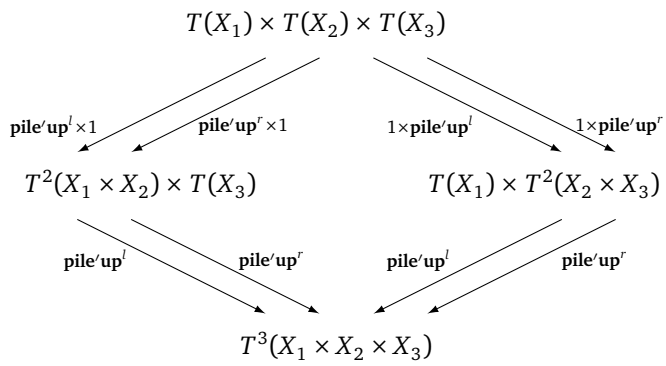
$$T(X) \times T(Y) \underset{\mathbf{CPS}^r(f)}{\overset{\mathbf{CPS}^l(f)}{\rightrightarrows}} Z$$

This immediately follows from the above lemma and the definition of **CPS**es. ♢

Using binary **pile′up** operations, we can define eight ternary **pile′up** operations:

$$T(X_1) \times T(X_2) \times T(X_3) \longrightarrow T(X_1 \times X_2 \times X_3)$$

out of the following diagram:



However, both $\mathbf{pile'up}^l$ and $\mathbf{pile'up}^r$ operations are associative (Proposition 5.3 below) and hence only six of them are different, in general.

Proposition 5.3 *Both* $\mathbf{pile'up}^l$ *and* $\mathbf{pile'up}^r$ *operations are associative on any monad on Set.*

Proof. In fact, $\mathbf{pile'up}^l$ and $\mathbf{pile'up}^r$ are associative on any bi-strong monad in the monoidal category. We shall show this fact for a monad T on *Set* with canonical strength.

We need to show that:

$$\mathbf{pile'up}^r \circ (\mathbf{pile'up}^r \times 1) = \mathbf{pile'up}^r \circ (1 \times \mathbf{pile'up}^r)$$

and

$$\mathbf{pile'up}^l \circ (\mathbf{pile'up}^l \times 1) = \mathbf{pile'up}^l \circ (1 \times \mathbf{pile'up}^l)$$

But as **pile′up**s are mutually definable, either of these equalities readily implies the other. We shall show the second equality. For sets X_1, X_2, X_3, using all the assumptions, we have:

$\mathbf{pile'up}^l{}_{X_1 \times X_2, X_3} \circ (\mathbf{pile'up}^l{}_{X_1, X_2} \times 1_{T(X_3)}) =$

$$= \mu_{X_1\times X_2\times X_3}\circ T(\mathbf{st}^r{}_{X_1\times X_2,X_3})\circ \underline{\mathbf{st}^l{}_{X_1\times X_2,T(X_3)}\circ(\mu_{X_1\times X_2}\times T(1_{X_3}))}$$
$$\circ(T(\mathbf{st}^r{}_{X_1,X_2})\times 1_{T(X_3)})\circ(\mathbf{st}^l{}_{X_1,T(X_2)}\times 1_{T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ \underline{T(\mathbf{st}^r{}_{X_1\times X_2,X_3})\circ\mu_{X_1\times X_2\times T(X_3)}}\circ T(\mathbf{st}^l{}_{X_1\times X_2,T(X_3)})$$
$$\circ \underline{\mathbf{st}^l{}_{T(X_1\times X_2),T(X_3)}\circ(T(\mathbf{st}^r{}_{X_1,X_2})\times T(1_{X_3}))}\circ(\mathbf{st}^l{}_{X_1,T(X_2)}\times 1_{T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1\times X_2,T(X_3)})\circ T(\mathbf{st}^l{}_{X_1\times X_2,T(X_3)})$$
$$\circ(T(\mathbf{st}^r{}_{X_1,X_2}\times 1_{X_3}))\circ \underline{\mathbf{st}^l{}_{T(X_1\times X_2),T(X_3)}\circ(\mathbf{st}^l{}_{X_1,T(X_2)}\times 1_{T(X_3)})}$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1\times X_2,T(X_3)})\circ \underline{T(\mathbf{st}^l{}_{X_1\times X_2,T(X_3)})}$$
$$\underline{\circ(T(\mathbf{st}^r{}_{X_1,X_2}\times 1_{T(X_3)}))}\circ \mathbf{st}^l{}_{X_1,T(X_2)\times T(X_3)}$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1\times X_2,T(X_3)})\circ T(\mathbf{st}^r{}_{X_1,X_2\times T(X_3)})$$
$$\circ \underline{T(1_{X_1}\times \mathbf{st}^l{}_{X_2,T(X_3)})\circ \mathbf{st}^l{}_{X_1,T(X_2)\times T(X_3)}}$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ \underline{T^2(\mathbf{st}^r{}_{X_1\times X_2,T(X_3)})}\circ T(\mathbf{st}^r{}_{X_1,X_2\times T(X_3)})$$
$$\circ \mathbf{st}^l{}_{X_1,T(X_2\times T(X_3))}\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})\circ \underline{T^2(1_{X_1}\times \mathbf{st}^r{}_{X_2,X_3})}$$
$$\underline{\circ T(\mathbf{st}^r{}_{X_1,X_2\times T(X_3)})}\circ \mathbf{st}^l{}_{X_1,T(X_2\times T(X_3))}\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})\circ T(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})$$
$$\circ \underline{T(1_{X_1}\times T(\mathbf{st}^r{}_{X_2,X_3}))\circ \mathbf{st}^l{}_{X_1,T(X_2\times T(X_3))}}\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ \underline{\mu_{T(X_1\times X_2\times X_3)}\circ T^2(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})\circ T(\mathbf{st}^r{}_{X_1,T(X_2\times X_3)})}$$
$$\circ \mathbf{st}^l{}_{X_1,T^2(X_2\times X_3)}\circ(T(1_{X_1})\times T(\mathbf{st}^r{}_{X_2,X_3}))\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ T(\mathbf{st}^r{}_{X_1,X_2\times X_3})\circ \underline{T(1_{X_1}\times \mu_{X_2\times X_3})\circ \mathbf{st}^l{}_{X_1,T^2(X_2\times X_3)}}$$
$$\circ(T(1_{X_1})\times T(\mathbf{st}^r{}_{X_2,X_3}))\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mu_{X_1\times X_2\times X_3}\circ T(\mathbf{st}^r{}_{X_1,X_2\times X_3})\circ \mathbf{st}^l{}_{X_1,T(X_2\times X_3)}\circ(T(1_{X_1})\times \mu_{X_2\times X_3})$$
$$\circ(T(1_{X_1})\times T(\mathbf{st}^r{}_{X_2,X_3}))\circ(T(1_{X_1})\times \mathbf{st}^l{}_{X_2,T(X_3)})$$
$$= \mathbf{pile'up}^l{}_{X_1,X_2\times X_3}\circ(1_{T(X_3)}\times \mathbf{pile'up}^l{}_{X_2,X_3}) \qquad \diamondsuit$$

5.3 Arity one: intransitive verbs

Proposition 5.4 *The semantics for sentences with intransitive verbs, as defined by strategies A, B, and C, will be equivalent.*

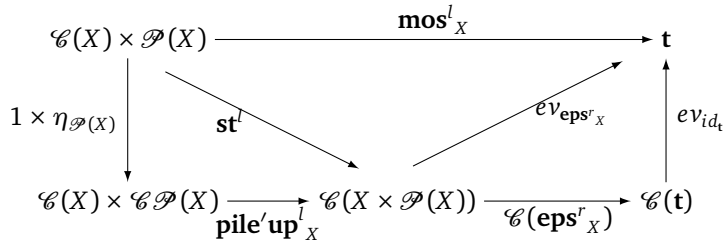
Proof. In case of a sentence with an intransitive verb, the semantics will be defined by the morphisms $\mathbf{strat}^1_A$, $\mathbf{strat}^1_B$, and $\mathbf{strat}^1_C$. We need to show that they are equal. We have:

$$\mathbf{strat}^1_A = \mathbf{mos}^l{}_X = \mathbf{strat}^1_B.$$

$\mathbf{strat}^1_C$ is the composition of the following morphisms:

$$\mathscr{C}(X) \times \mathscr{P}(X) \xrightarrow{1 \times \eta_{\mathscr{P}(X)}} \mathscr{C}(X) \times \mathscr{C}\mathscr{P}(X) \xrightarrow{\mathbf{CPS}^l(\mathbf{eps}^r{}_X)} \mathscr{C}(\mathbf{t}) \xrightarrow{ev_{id_{\mathbf{t}}}} \mathbf{t}$$

Thus we need to show that this composition is equal to $\mathbf{mos}^l{}_X$. Consider the following diagram:



The left triangle commutes, as a consequence of Lemma 5.1. To see that the central triangle commutes, we take $M \in \mathscr{C}(X)$ and $h \in \mathscr{P}(X)$, and calculate:

$$\begin{aligned} ev_{\mathbf{eps}^r{}_X} \circ \mathbf{st}^r(Q,h) &= ev_{\mathbf{eps}^r{}_X}(\lambda D_{:\mathscr{P}(X\times\mathscr{P}(X))} M(\lambda x_{:X} D(x,h))) \\ &= M(\lambda x_{:X} \mathbf{eps}^r{}_X(x,h)) \\ &= M(\lambda x_{:X} h(x)) \\ &= N(h) \\ &= \mathbf{mos}^l(N,h). \end{aligned}$$

Finally, to see that the right triangle commutes, we take $N \in \mathscr{C}(X \times \mathscr{P}(X))$ and calculate:

$$\begin{aligned} ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^r{}_X)(N) &= ev_{id_{\mathbf{t}}}(\lambda c_{:\mathscr{P}(\mathbf{t})} N(c \circ \mathbf{eps}^r{}_X)) \\ &= N(\mathbf{eps}^r{}_X) \\ &= ev_{\mathbf{eps}^r{}_X}(N). \end{aligned}$$

Thus the whole diagram commutes, and hence $\mathbf{strat}^1_C = \mathbf{mos}^l{}_X$, as required. $\diamondsuit$

The above proof also shows the following technical lemma.

Lemma 5.5 *For any set X, the following diagram commutes:*

$$\begin{array}{ccc} \mathscr{C}(X) \times \mathscr{P}(X) & \xrightarrow{\mathbf{mos}^l{}_X} & \mathbf{t} \\ \downarrow{\scriptstyle \mathbf{st}^l} & & \uparrow{\scriptstyle ev_{id_{\mathbf{t}}}} \\ \mathscr{C}(X \times \mathscr{P}(X)) & \xrightarrow[\mathscr{C}(\mathbf{eps}^r{}_X)]{} & \mathscr{C}(\mathbf{t}) \end{array}$$

5.4 *Arity two: transitive verbs*

Proposition 5.6 *The semantics for sentences with transitive verbs, as defined by strategies A, B, and C, will be equivalent, providing two asymmetric readings for the sentence.*

Proof. In the case of sentences with transitive verbs, the semantics will be defined by morphisms $\mathbf{strat}^{2,\sigma}_A$, $\mathbf{strat}^{2,\sigma}_B$, and $\mathbf{strat}^{2,\varepsilon}_C$, with $\sigma \in S_2 = \{id_2, \tau\}$ and $\varepsilon \in \{l, r\}$. We need to show the equalities:

$$\mathbf{strat}^{2,\sigma}_A = \mathbf{strat}^{2,\sigma}_B,$$

for $\sigma \in S_2$, and

$$\mathbf{strat}^{2,id_2}_B = \mathbf{strat}^{2,l}_C, \qquad \mathbf{strat}^{2,\tau}_B = \mathbf{strat}^{2,r}_C.$$

To show the first equality, with $Q_1 \in \mathscr{C}(X_1)$, $Q_2 \in \mathscr{C}(X_2)$, and $P \in \mathscr{P}(X_1 \times X_2)$, we have:

$$\begin{aligned}
&\mathbf{strat}^{2,\sigma}_A(Q_1, Q_2, P) = \\
&= \mathbf{mos}^l{}_{X_{\sigma(1)}}(Q_{\sigma(1)}, \mathbf{mos}^l{}_{X_{\sigma(2)}}(Q_{\sigma(2)}, P)) \\
&= \mathbf{mos}^l{}_{X_{\sigma(1)}}(Q_{\sigma(1)}, \lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.P(x_1, x_2))) \\
&= Q_{\sigma(1)}(\lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.P(x_1, x_2))) \\
&= Q_{\sigma(1)}(\lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.P(\pi_{\sigma^{-1}}(x_{\sigma(1)}, x_{\sigma(2)})))) \\
&= \mathbf{pile'up}^l(Q_{\sigma(1)}, Q_{\sigma(2)})(P \circ \pi^{-1}) \\
&= \mathscr{C}(\pi_{\sigma^{-1}})(\mathbf{pile'up}^l(Q_{\sigma(1)}, Q_{\sigma(2)}))(P) \\
&= \mathbf{strat}^{2,\sigma}_B(Q_1, Q_2, P)
\end{aligned}$$

To show the remaining two equalities, let us first note that if either $\sigma = id_2$ and $\varepsilon = l$ or $\sigma = \tau$ and $\varepsilon = r$, we have:

$$\mathbf{pile'up}^{\varepsilon} = \mathscr{C}(\pi_{\sigma^{-1}}) \circ \mathbf{pile'up}^{l} \circ \pi_{\sigma}.$$

Thus we shall assume the above equation relating σ with ε, and, with $Q_1 \in \mathscr{C}(X_1)$, $Q_2 \in \mathscr{C}(X_2)$, and $P \in \mathscr{P}(X_1 \times X_2)$, we obtain:[8]

$\mathbf{strat}_C^{2,\varepsilon} =$

$= ev_{id_t} \circ \mathbf{CPS}^{\varepsilon}(\mathbf{eps}^{r}{}_{X_1}) \circ (1 \times \mathbf{CPS}^{?}(\mathbf{eps}^{r}{}_{X_2})) \circ (1 \times 1 \times \eta_{\mathscr{P}(X_1 \times X_2)})$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ \mathbf{pile'up}^{\varepsilon} \circ (\mathscr{C}(1) \times \mathscr{C}(\mathbf{eps}^{r}{}_{X_2})) \circ (1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times \eta)$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ \mathscr{C}(1 \times \mathbf{eps}^{r}{}_{X_2}) \circ \mathbf{pile'up}^{\varepsilon} \circ (1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times \eta)$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{pile'up}^{\varepsilon} \circ (1 \times \mathbf{pile'up}^{?}) \circ (1 \times 1 \times \eta)$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{pile'up}^{?} \circ (\mathbf{pile'up}^{\varepsilon} \times 1) \circ (1 \times 1 \times \eta)$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{pile'up}^{?} \circ (1 \times \eta) \circ (1 \times \mathbf{pile'up}^{\varepsilon})$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{st}^{l} \circ (1 \times \mathbf{pile'up}^{\varepsilon})$

$= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2}) \circ \mathbf{st}^{l} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times 1) \circ (\mathbf{pile'up}^{l} \times 1) \circ (\pi_{\sigma} \times 1)$

$= \mathbf{mos}^{l}{}_{X_1 \times X_2} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times 1) \circ (\mathbf{pile'up}^{l} \times 1) \circ (\pi_{\sigma} \times 1)$

$= \mathbf{strat}_B^{2,\sigma}$

In the above calculations, we used the definition of **CPS**es, the naturality of $\mathbf{pile'up}^{\varepsilon}$, the relations between **eps** morphisms, the associativity of $\mathbf{pile'up}^{\varepsilon}$ (Proposition 5.3), the properties of product morphisms, the **pile′up** lemma, and finally, Lemma 5.5.

Here and below, $\mathbf{CPS}^{?}$, $\mathbf{pile'up}^{?}$ stands for either $\mathbf{CPS}^{l}$, $\mathbf{pile'up}^{l}$ or $\mathbf{CPS}^{r}$, $\mathbf{pile'up}^{r}$,whichever is more convenient at the time, as it does not influence the end result. ♢

[8] The diagram illustrating these calculations would be too big to fit on a page but the reader is encouraged to draw one.

5.5 Arity three: ditransitive verbs

Proposition 5.7 *The semantics for sentences with ditransitive verbs, as defined by strategies A and B, will be equivalent, providing six asymmetric readings of the sentence.*

Proof. In the case of sentences with ditransitive verbs, the semantics will be defined by the morphisms $\mathbf{strat}_A^{3,\sigma}$, $\mathbf{strat}_B^{3,\sigma}$, and $\mathbf{strat}_C^{2,\varepsilon}$, with $\sigma \in S_3$ and $\varepsilon \in \{l, r\}$. We need to show the equalities:

$$\mathbf{strat}_A^{3,\sigma} = \mathbf{strat}_B^{3,\sigma},$$

for $\sigma \in S_3$.

The calculations are similar to those for transitive verbs. We present them for completeness. With $Q_1 \in \mathscr{C}(X_1)$, $Q_2 \in \mathscr{C}(X_2)$, $Q_3 \in \mathscr{C}(X_3)$, and $P \in \mathscr{P}(X_1 \times X_2 \times X_3)$, we have:

$$\mathbf{strat}_A^{3,\sigma}(Q_1, Q_2, Q_3, P) =$$

$$= \mathbf{mos}^l{}_{X_{\sigma(1)}}(Q_{\sigma(1)}, \mathbf{mos}^l{}_{X_{\sigma(2)}}(Q_{\sigma(2)}, \mathbf{mos}^l{}_{X_{\sigma(3)}}(Q_{\sigma(3)}, P)))$$

$$= Q_{\sigma(1)}(\lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.Q_{\sigma(3)}(\lambda x_{\sigma(3):X_{\sigma(3)}}.P(x_1, x_2, x_3))))$$

$$= Q_{\sigma(1)}(\lambda x_{\sigma(1):X_{\sigma(1)}}.Q_{\sigma(2)}(\lambda x_{\sigma(2):X_{\sigma(2)}}.Q_{\sigma(3)}(\lambda x_{\sigma(3):X_{\sigma(3)}}.P(\pi_{\sigma^{-1}}(x_{\sigma(1)}, x_{\sigma(2)}, x_{\sigma(3)})))))$$

$$= \mathbf{pile}'\mathbf{up}^l(Q_{\sigma(1)}, \mathbf{pile}'\mathbf{up}^l(Q_{\sigma(2)}, Q_{\sigma(3)}))(P \circ \pi_{\sigma^{-1}})$$

$$= \mathscr{C}(\pi_{\sigma^{-1}})(\mathbf{pile}'\mathbf{up}^l(Q_{\sigma(1)}, \mathbf{pile}'\mathbf{up}^l(Q_{\sigma(2)}, Q_{\sigma(3)})))(P)$$

$$= \mathbf{strat}_B^{2,\sigma}(Q_1, Q_2, Q_3, P)$$

as required. ♢

Proposition 5.8 *The semantics for sentences with ditransitive verbs, as defined by strategy C, provides four asymmetric readings of the sentence, such that QP in subject position can be placed either first or last only. Thus they correspond to four out of the six readings accounted for by strategies A and B.*

Proof. In the case of sentences with ditransitive verbs, the semantics, according to strategies B and C, are defined by the morphisms $\mathbf{strat}_B^{3,\sigma}$,

$\mathbf{strat}_C^{3,\varepsilon,\varepsilon'}$, respectively. As we shall show, these morphisms are equal whenever $\sigma \in S_3$ is related to the pair $\langle \varepsilon', \varepsilon \rangle \in \{l, r\}^2$ via the relation:

$$\mathbf{pile}'\mathbf{up}^{\varepsilon'} \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon}) = \mathscr{C}(\pi_{\sigma^{-1}}) \circ \mathbf{pile}'\mathbf{up}^{l} \circ (1 \times \mathbf{pile}'\mathbf{up}^{l}) \circ \pi_{\sigma}$$

As $\mathbf{pile}'\mathbf{up}^{l}$ leaves the order intact and $\mathbf{pile}'\mathbf{up}^{r}$ swaps the order, we can see that we have the following correspondence:

σ	$\langle \varepsilon', \varepsilon \rangle$
$(1,2,3)$	$\langle l, l \rangle$
$(1,3,2)$	$\langle l, r \rangle$
$(2,3,1)$	$\langle r, l \rangle$
$(3,2,1)$	$\langle r, r \rangle$
$(2,1,3)$	–
$(3,1,2)$	–

Thus we shall assume that σ is related to the pair $\langle \varepsilon, \varepsilon' \rangle$, and, with $Q_1 \in \mathscr{C}(X_1)$, $Q_2 \in \mathscr{C}(X_2)$, $Q_3 \in \mathscr{C}(X_3)$, and $P \in \mathscr{P}(X_1 \times X_2 \times X_3)$, we obtain:[9]

$$\begin{aligned}
&\mathbf{strat}_C^{3,\varepsilon',\varepsilon} = \\
&= ev_{id_t} \circ \mathbf{CPS}^{\varepsilon'}(\mathbf{eps}^{r}{}_{X_1}) \circ (1 \times \mathbf{CPS}^{\varepsilon}(\mathbf{eps}^{r}{}_{X_2})) \circ (1 \times 1 \times \mathbf{CPS}^{?}(\mathbf{eps}^{r}{}_{X_3})) \\
&\quad \circ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ \mathbf{pile}'\mathbf{up}^{\varepsilon'} \circ (\mathscr{C}(1) \times \mathscr{C}(\mathbf{eps}^{r}{}_{X_2})) \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon}) \\
&\quad \circ (\mathscr{C}(1) \times \mathscr{C}(1) \times \mathscr{C}(\mathbf{eps}^{r}{}_{X_3})) \circ (1 \times 1 \times \mathbf{pile}'\mathbf{up}^{?}) \circ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ (\mathscr{C}(1 \times \mathbf{eps}^{r}{}_{X_2})) \circ \mathbf{pile}'\mathbf{up}^{\varepsilon'} \circ (\mathscr{C}(1) \times \mathscr{C}(1 \times \mathbf{eps}^{r}{}_{X_3})) \\
&\quad \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon}) \circ (1 \times 1 \times \mathbf{pile}'\mathbf{up}^{?}) \circ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1}) \circ (\mathscr{C}(1 \times \mathbf{eps}^{r}{}_{X_2})) \circ (\mathscr{C}(1 \times 1 \times \mathbf{eps}^{r}{}_{X_3})) \circ \mathbf{pile}'\mathbf{up}^{\varepsilon'} \\
&\quad \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon}) \circ (1 \times 1 \times \mathbf{pile}'\mathbf{up}^{?}) \circ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_t} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile}'\mathbf{up}^{\varepsilon'} \\
&\quad \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon}) \circ (1 \times 1 \times \mathbf{pile}'\mathbf{up}^{?}) \circ (1 \times 1 \times 1 \times \eta)
\end{aligned}$$

[9] The diagram illustrating these calculations would again be too big to fit on a page, but the reader is encouraged to draw one.

$$
\begin{aligned}
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile}'\mathbf{up}^{\varepsilon'} \\
&\quad \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon}) \circ (1 \times 1 \times \mathbf{pile}'\mathbf{up}^{?}) \circ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile}'\mathbf{up}^{\varepsilon'} \\
&\quad \circ (1 \times \mathbf{pile}'\mathbf{up}^{?}) \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon} \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{\varepsilon'} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{\varepsilon} \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \\
&\overset{*}{=} ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{pile}'\mathbf{up}^{?} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times \mathscr{C}(1)) \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_{\sigma} \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_{\sigma} \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \\
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ\ (1 \times 1 \times 1 \times \eta) \circ (\pi_{\sigma} \times 1) \\
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ\ (1 \times \times \eta) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_{\sigma} \times 1) \\
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ\ (1 \times 1 \times \eta) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_{\sigma} \times 1) \\
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{pile}'\mathbf{up}^{?} \\
&\quad \circ (1 \times \eta) \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_{\sigma} \times 1) \\
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathscr{C}(\pi_{\sigma^{-1}} \times 1) \circ \mathbf{st}^{l\,l} \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_{\sigma} \times 1) \\
&= ev_{id_{\mathbf{t}}} \circ \mathscr{C}(\mathbf{eps}^{r}{}_{X_1 \times X_2 \times X_3}) \circ \mathbf{st}^{l\,l} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times \mathscr{C}(1)) \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_{\sigma} \times 1) \\
&= \mathbf{mos}^{l}_{X_1 \times X_2 \times X_3} \circ (\mathscr{C}(\pi_{\sigma^{-1}}) \times \mathscr{C}(1)) \\
&\quad \circ (\mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (1 \times \mathbf{pile}'\mathbf{up}^{l} \times 1) \circ (\pi_{\sigma} \times 1) \\
&= \mathbf{strat}^{3,\sigma}_{B}
\end{aligned}
$$

In the above calculations, we used the definition of **CPS**es, the naturality of **pile′up**s (four times in three non-consecutive steps!), the relations between **eps** morphisms, the associativity of **pile′up**s (Proposition 5.3), the relations between σ and $\langle \varepsilon', \varepsilon \rangle$ in the equation marked with $* \overset{*}{=}$, the properties of product morphisms (three consecutive steps), the **pile′up** lemma, the naturality of strength, and finally, Lemma 5.5. $\diamondsuit$

6 ACKNOWLEDGMENTS

This article is funded by the National Science Center on the basis of decision DEC-2016/23/B/HS1/00734. The authors would like to thank the anonymous reviewers for valuable comments.

REFERENCES

Chris BARKER (2002), Continuations and the nature of quantification, *Natural language semantics*, 10(3):211–242.

Chris BARKER and Chung-chieh SHAN (2014), *Continuations and natural language*, volume 53, Oxford Studies in Theoretical Linguistics.

Daisuke BEKKI and Kenichi ASAI (2009), Representing covert movements by delimited continuations, in *JSAI International Symposium on Artificial Intelligence*, pp. 161–180, Springer.

Simon CHARLOW (2014), *On the semantics of exceptional scope*, Ph.D. thesis, New York University.

Noam CHOMSKY (1993), *Lectures on government and binding: The Pisa lectures*, 9, Walter de Gruyter.

Robin COOPER (1983), *Quantification and semantic theory*, Dordrecht: Reidel.

Philippe DE GROOTE (2001), Type raising, continuations, and classical logic, in *Proceedings of the thirteenth Amsterdam Colloquium*, pp. 97–101.

Samuel EILENBERG and G Max KELLY (1966), Closed categories, in *Proceedings of the Conference on Categorical Algebra*, pp. 421–562, Springer.

Samuel EILENBERG, John C MOORE, *et al.* (1965), Adjoint functors and triples, *Illinois Journal of Mathematics*, 9(3):381–398.

Roger GODEMENT (1958), *Topologie algébrique et théorie des faisceaux*, volume 13, Hermann Paris.

Justyna GRUDZINSKA and Marek ZAWADOWSKI (2016), Continuation semantics for multi-quantifier sentences: operation-based approaches, *arXiv preprint arXiv:1608.00255*.

Herman HENDRIKS (1993), *Studied flexibility: Categories and types in syntax and semantics*, Institute for Logic, Language and Computation.

Edward L KEENAN (1987), Unreducible n-ary quantifiers in natural language, in *Generalized quantifiers*, pp. 109–150, Springer.

Edward L KEENAN (1992), Beyond the Frege boundary, *Linguistics and Philosophy*, 15(2):199–221.

Oleg KISELYOV and Chung-chieh SHAN (2014), Continuation hierarchy and quantifier scope, in *Formal Approaches to Semantics and Pragmatics*, pp. 105–134, Springer.

Heinrich KLEISLI (1965), Every standard construction is induced by a pair of adjoint functors, *Proceedings of the American Mathematical Society*, 16(3):544–546.

Anders KOCK (1970), Monads on symmetric monoidal closed categories, *Archiv der Mathematik*, 21(1):1–10.

Anders KOCK (1971), Closed categories generated by commutative monads, *Journal of the Australian Mathematical Society*, 12(04):405–424.

Anders KOCK (1972), Strong functors and monoidal monads, *Archiv der Mathematik*, 23(1):113–120.

Robert MAY (1978), *The grammar of quantification.*, Ph.D. thesis, Massachusetts Institute of Technology.

Robert MAY (1985), *Logical Form: Its structure and derivation*, volume 12, MIT press.

Eugenio MOGGI (1991), Notions of computation and monads, *Information and computation*, 93(1):55–92.

Richard MONTAGUE (1973), The proper treatment of quantification in ordinary English, in *Approaches to natural language*, pp. 221–242, Springer.

Chung-chieh SHAN (2002), Monads for natural language semantics, *arXiv preprint cs/0205026*.

Johan VAN BENTHEM (1989), Polyadic quantifiers, *Linguistics and Philosophy*, 12(4):437–464.

Philip WADLER (1990), Comprehending monads, in *Proceedings of the 1990 ACM conference on LISP and functional programming*, pp. 61–78, ACM.

Marek ZAWADOWSKI (1989), Formalization of the feature system in terms of preorders, *Feature System for Quantification Structures in Natural Language [3]*, pp. 155–175.

Type Theories and Lexical Networks: using Serious Games as the basis for Multi-Sorted Typed Systems*

Stergios Chatzikyriakidis[1], *Mathieu Lafourcade*[2], *Lionel Ramadier*[3], *and Manel Zarrouk*[4]

[1] Centre for Linguistic Theory and Studies in Probability (CLASP), Department of Philosophy, Linguistics and Theory of Science, University of Gothenburg; Open University of Cyprus

[2] LIRMM, University of Montpellier

[3] Radiology Dept. CHU Montpellier

[4] National University of Ireland, Galway

ABSTRACT

In this paper, we show how a rich lexico-semantic network which has been built using serious games, JeuxDeMots, can help us in grounding our semantic ontologies in doing formal semantics using rich or modern type theories (type theories within the tradition of Martin Löf). We discuss the issue of base types, adjectival and verbal types, hyperonymy/hyponymy relations as well as more advanced issues like homophony and polysemy. We show how one can take advantage of this wealth of lexical semantics in a formal compositional semantics framework. We argue that this is a way to sidestep the problem of deciding what the type ontology should look like once a move to a many sorted type system has been made. Furthermore, we show how this kind of information can be extracted from a lexico-semantic network like JeuxDeMots and inserted into a proof-assistant like Coq in order to perform reasoning tasks.

Keywords: Lexical Networks, JeuxDeMots, Type Theory, Type Ontologies, Formal Semantics, Natural Language Inference

*The first author supported by a grant from the Swedish Research Council for the establishment of the Centre for Linguistic Theory and Studies in Probability (CLASP) at the University of Gothenburg.

1 INTRODUCTION

Modern Type Theories (MTTs), i.e. Type Theories within the tradition of Martin-Löf (1975); Martin-Löf (1984), have become a major alternative to Montague Semantics (MS) in the last twenty years. A number of influential approaches using MTTs have been proposed throughout this period (Ranta 1994; Luo 2011; Retoré 2014; Cooper *et al.* 2014), showing that the rich typing system offered by these approaches (type many-sortedness, dependent types, type universes among other things) has considerable advantages over simple typed systems predominantly used in mainstream formal semantics. A further important aspect for considering the use of MTTs over traditional Montagovian frameworks concerns the proof-theoretic nature of the former but not of the latter.[1] This latter fact makes MTTs a suited formal semantics language to perform reasoning tasks, as these are exemplified for example in work on inference using proof-assistant technology (Chatzikyriakidis and Luo 2014b,a; Bernardy and Chatzikyriakidis 2017). However, this expressiveness of typing comes with a cost. For example, how does one decide on the base types to be represented? On the one hand, we do have a way to get a more fine-grained type system unlike the monolithic domain of entities found in MS, but on the other hand, constructing such a type ontology is not at all a straightforward and easy task. Different approaches and assumptions have been put forward w.r.t this issue. For example Luo (2011, 2012); Chatzikyriakidis and Luo (2017b) proposed to treat CNs as types, in effect arguing that every CN is a type (roughly a one to one correspondence between common nouns and types). Approaches like Retoré (2014) on the other hand, take a more moderate view and build their typing ontology according to classifier systems, i.e. the intuitions for deciding which types are to be represented or not are taken from classifier systems found in a number of natural languages. On the other hand, work in lexical-semantic networks have provided us with structured lexicons specifying elaborate

[1] At least in the way it is employed in the Montagovian setting, simple type theory can be viewed as model theoretic. However, there is interesting work on the proof theory of simple type theory. The higher order theorem prover LEO-II Benzmüller *et al.* (2007) is an example of such work. We are grateful to an anonymous reviewer for pointing this out to us.

lexical and semantic relations. A classic such case is e.g. WordNet Fellbaum (1998). A very promising line of research in lexico-semantic network construction concerns networks which are built collaboratively by using Games with a Purpose (GWAPs). This is the case of the Lexical Network JeuxDeMots (JDM) (Lafourcade 2007b). JDM is constructed through many GWAPs along with a contributive tool (Diko) which allows players/users to contribute directly and to browse the knowledge base.

Given this background, what we want to propose in this paper is the grounding of our semantic ontologies, as well as any other information needed in order to perform reasoning tasks using MTT semantics, in JDM. In order to do this, we present some first thoughts on how such an endeavour can be accomplished by looking at the way a translation procedure from JDM to MTTs can be performed. Issues to be discussed include the domain of base types, instances of these types, adjectival and verbal types, hyponymy/hypernomy relations, as well as more advanced issues like homophony and polysemy. We then show how one can exploit this translation procedure by extracting this information from JDM in order to feed a reasoning device that implements an MTT. We show some easy cases of inference that are taken care of via a combination of the lexical semantics information extracted from JDM and the proof theoretic power of MTTs (performed by the proof-assistant Coq) and further show how JDM can actually help us in order to reason with cases where reasoning with implicit premises is at play. The structure of the paper is as follows: in Section 2, the JDM project is described as well as the produced lexical network. In Section 3, we describe two main endogenous inference mechanisms (deductive and inductive scheme), followed by a discussion on the annotation of relations between terms. Then, in Section 4, we discuss the building of type ontologies using information from JDM and propose a number of translation procedures between JDM and an MTT. The section also includes a brief intro to MTT semantics, highlighting aspects of the theory that will play a role in this paper the most. Lastly, in Section 5 we look at the possibility of performing natural language inference tasks using MTT semantics powered by information drawn from JDM. We present a number of inference cases that rely mostly on lexical-semantic information taken by JDM and the proof-theoretic power of MTT semantics using the proof-assistant Coq.

2 BUILDING A LEXICAL NETWORK

JeuxDeMots[2], a project launched in September 2007, aims to build a large lexico-semantic network (Lafourcade 2007a). The network is composed of terms (nodes or vertices) and typed relations (links between nodes). It contains terms and possible refinements in the same spirit as WordNet synsets (Miller 1995), although being organized as decision trees. There are more than 80 different relation types which occurrences are directed, weighted, and possibly annotated (Lafourcade *et al.* 2015).

2.1 *GWAPs*

The game JeuxDeMots is a two player GWAP (Game With A Purpose, see von Ahn and Dabbish 2008), where people are supposed to *earn and collect* words. The main mechanism whereby this goal is achieved is the provision of lexical and semantic associations to terms proposed by the system.

When a Player (let's call him/her A) starts a game, a term T, along with some instructions concerning the type of lexical relation (e.g. synonym, antonym, domain, etc.), is displayed. The term T could have been chosen from the database by the system or offered to be played by other players. Player A has a limited amount of time (around 60 seconds) to enter terms which, to his/her mind, are relevant w.r.t. both the term T and the lexical relation. The maximum number of terms a player can enter is limited, thus encouraging the player to think carefully about his/her choices. A screenshot of the user interface is shown in Figure 1.

The very same term T, along with the same set of instructions, will be later given to another player, Player B, for whom the process is identical. In order to make the game more entertaining, the two players score points for words they both choose. Score calculation is explained in Lafourcade (2007a) and was designed to increase both precision and recall in the construction of the database. The more 'original' a proposition given by both players is, the more it is rewarded. Figure 2 shows an end of game with collected rewards. Answers given by both players are displayed and those common to both players, as well as their scores, are highlighted.

[2] http://www.jeuxdemots.org

Figure 1: Screenshot of an ongoing game with the target verb *fromage* (cheese). Several propositions have been given by the user and are listed on the right hand side

Figure 2: Screenshot of the game result with the target noun *fromage*. Proposals of both players are displayed, along with points won by both

For a target term T, common answers from both players are inserted into the database. Answers given by only one of the two players are not, thus reducing noise and the chance of database corruption. The semantic network is, therefore, constructed by connecting terms by typed and weighted relations, validated by pairs of players. These relations are labeled according to the instructions given to the players and weighted according to the number of pairs of players who choose them. Initially, prior to putting the game online, the database was populated with nodes. However if a pair of players suggests a non-existing term, the new node is added to the database.

In the interest of quality and consistency, it was decided that the validation process would involve anonymous players playing together. A relation is considered valid if and only if it is given by at least one pair of players. This validation process is similar to that presented by von Ahn and Dabbish (2004) for the indexing of images, by Lieberman *et al.* (2007) and von Ahn *et al.* (2006) to collect common sense knowledge, and Siorpaes and Hepp (2008) for knowledge extraction. As far as we know, this technique has never been used for building semantic networks. Similar Web-based systems already exist in NLP, such as Open Mind Word Expert (Mihalcea and Chklovski 2003), which aims to create large sense-tagged corpora with the help of Web users, and SemKey Marchetti *et al.* (2007), which makes use of WordNet and Wikipedia to disambiguate lexical forms referring to concepts, thus identifying semantic keywords.

For the design of JeuxDeMots, we could have chosen to take into account all of the players' answers according to their frequency from the very outset. The database would have grown much quicker this way, but to the detriment of quality. The rationale behind this choice was to limit the impact of fanciful answers or errors due to misinterpreted instructions or terms. The integration of rarer terms and expressions is slower; nevertheless, these terms are added to the database eventually, once the more common solutions have been exhausted, thanks to the process of creating *taboo* terms. Once a relation with term T has been proposed by a large number of pairs of players, it becomes taboo. During a game, taboo terms are displayed along with term T, discouraging (but not forbidding) players from entering them. In this way, players are encouraged to make other, more original choices. Therefore, more infrequent terms eventually find

their way into the database, and the chances of error are reduced to a minimum.

Even if a relation becomes taboo, its weight can, and does, evolve. However, this tends to be done slowly as the relation is proposed to the players less often. It is important to allow relation weights to continue to evolve, as we can hardly consider such a relation as complete. Eventually, a given term can become taboo when involved in several different relation types. The fact that taboo relations continue to evolve is essential, otherwise the weights of two given relations could become equal and then information about the relative strength relations would be lost.

The approach presented here complements that developed by Zock and Bilac (2004) and Zock and Schwab (2008) who tried to create an index based on the notion of association to assist users in navigating the Web or elsewhere, or to help a person find a word on the tip of their tongue. Their approach is bottom-up, i.e. the terms are known (based on word proximity in corpora), but the nature of the link isn't. This has to be inferred, which is far from an easy task. In our case, we provide one of the two terms, term T as well as the relation type. It is the target terms which interest us. Our approach is top-down.

Some other games[3] complement the main game of JDM. Their purpose is to cross validate the information collected in the main game, or to accelerate the relation harvesting for some specific types of relations. For instance, there are games for collecting word polarity (positive, negative, and neutral), for sentiments associated with words, guessing games, sorting games, location preposition games, and so on.

Since September 2007, around 1.5 million matches have been played for JDM, a total of 25 000 hours of cumulative playing. More than 250 million matches have been played for the other games of the JDM platforms.[4]

2.2 *Direct crowdsourcing*

Playing games in order to fill the lexical network is a kind of indirect crowdsourcing, where people (players) do not negotiate their contri-

[3] http://imaginat.name/JDM/Page_Liens_JDMv1.html

[4] http://www.jeuxdemots.org/jdm-about.php

bution beforehand. In some cases, direct crowdsourcing (with negotiation between contributors) is desirable. Indeed, some lexical relation might be complicated enough to be playable without some linguistic knowledge. This is for example the case for TELIC ROLE, which is the goal/purpose of an object (or action). For instance, a *butcher knife* has the telic role of *cutting meat*. It is to be differentiated from the INSTRUMENT of a predicate, which indicates what can be done with the object. A butcher knife could be used to *stab someone*, but this is not its telic role.

In some other cases (depending on each term), a given relation might not be productive enough to be playable. For example, the CAN PRODUCE relation for *cow* could reasonably be *milk*, but there are not many other answers.

All theses considerations lead to the need of a more direct crowdsourcing interface. The Diko[5] service allows to visualize and contribute to the JDM lexical network. A voting mechanism is at the core of the validation (or invalidation) of proposed relations between terms.

2.3 *Inside the JDM Lexical Network*

As mentioned above, the structure of the lexical network we are building relies on the notions of nodes and relations between nodes, as it was initially introduced in the end of 1960s by Collins and Quillian (1969), developed in Sowa and Zachman (1992), used in the small worlds by Gaume *et al.* (2007), and more recently clarified by Polguère (2014). Every node of the network is composed of a label (which is a term or an expression, or potentially any kind of string) grouping together all of its possible meanings.

The relations between nodes are typed. Each type corresponding to specific semantics that could be more or less precise. Some of these relations correspond to lexical functions, some of which have been made explicit by Mel'cuk and Zholkovsky (1988) and Polguère (2003). We would have liked our network to contain all the lexical functions defined by Mel'cuk, but, considering the principle of our software, JDM, this is not viable. Indeed, some of these lexical functions are too specialized and typically aim at some generative procedure (instead of

[5] http://www.jeuxdemots.org/diko.php

automatic text analysis and understanding), as in our case. For example, we can consider the distinction between the Conversive, Antonym, and Contrastive functions, a distinction that could be made through annotations for a quite generic antonym relation. Mel'cuk also considers function refinements, with lexical functions characterized as "wider" or "narrower". Given that JDM is intended for users who are "simple Internet users" and not necessarily experts in linguistics, such functions could be wrongly interpreted. Furthermore, some of these functions are clearly too poorly lexicalized, that is, very few terms feature occurrences of such relations. This is, for example, the case of the functions of 'Metaphor' or 'Functioning with difficulty'.

JDM has a predefined list of around 80 relation types, and players cannot define new types by themselves. These types of relations fall into several categories:

• Lexical relations: synonymy, antonymy, expression, lexical family. These types of relations are about vocabulary and lexicalization.

• Ontological relations: generic (hyperonymy), specific (hyponymy), part of (meronymy), whole of (holonymy), mater/substance, instances (named entities), typical location, characteristics and relevant properties.

• Associative relations: free associations, associated feelings, meanings, similar objects, more and less intense (Magn and anti-Magn). These relations are rather about subjective and global knowledge; some of them can be considered phrasal associations.

• Predicative relations: typical agent, typical patient, typical instrument, location where the action takes place, typical manner, typical cause, typical consequence etc. These relations are about types of relations associated with a verb (or action noun) as well as the values of its arguments (in a very wide sense).

Some relation types are specific to some noun classes. For example, for a noun referring to an intellectual piece of work (book, novel, movie, piece of art, etc.), the relation of *author* is defined. In case of a medical entity, targets and symptoms are defined.

Some outgoing relations for the French word *fromage* are shown below:

fromage → r_associated 800 → lait
fromage → r_associated 692 → camembert
fromage → r_associated 671 → chèvre

fromage → r_associated 580 → vache
fromage → r_associated 571 → gruyère
fromage → r_associated 460 → brebis
fromage → r_associated 419 → roquefort
fromage → r_isa 310 → produit laitier
fromage → r_associated 257 → produit laitier
fromage → r_associated 221 → brie
fromage → r_hypo 214 → gruyère
fromage → r_meaning 205 → produit laitier
fromage → r_hypo 204 → brie
fromage → r_associated 201 → dessert
fromage → r_associated 201 → fromage blanc
fromage → r_locution 199 → fromage de brebis
fromage → r_patient-1 199 → manger
fromage → r_locution 195 → fromage de tête
fromage → r_hypo 189 → fromage blanc
fromage → r_isa 189 → aliment
fromage → r_raff_sem 183 → fromage > produit laitier
fromage → r_isa 182 → ingrédient
fromage → r_lieu 182 → pizza
fromage → r_carac 180 → puant
fromage → r_sentiment 177 → envie
fromage → r_consequence 173 → puer du bec
fromage → r_holo 171 → pizza
fromage → r_associated 168 → laitage
fromage → r_hypo 167 → fromage de vache
fromage → r_hypo 163 → fromage double crème
fromage → r_hypo 163 → fromage à pâte pressée cuite
fromage → r_part_of 163 → lipide
fromage → r_part_of 161 → croûte
fromage → r_lieu :160 → plateau à fromage
fromage → r_carac 160 → odorant
fromage → r_associated#0:154 → raclette
fromage → r_locution :154 → dommage fromage
fromage → r_associated 149 → cancoillotte
fromage → r_locution 148 → faire tout un fromage
fromage → r_locution :148 → fromage analogue
fromage → r_locution :148 → fromage de synthèse

fromage → r_hypo 148 → fromage à pâte dure
fromage → r_similar 148 → substitut de fromage
fromage → r_hypo#8:147 → emmental
…

2.4 *Refinements*

Word senses (or usages) of a given term T are represented as standard nodes T > glose$_1$, T > glose$_2$, ..., T > glose$_n$ which are linked with REFINE(ment) relations. Glosses are terms that help the reader to identify the proper meanings of the term T. For example, consider the French term *frégate (Eng. frigate)*:

- *frégate*→REFINE→*frégate > navire*
  - *frégate > navire* →REFINE→ *frégate > navire > ancient*
  - *frégate > navire* →REFINE→ *frégate > navire > modern*
- *frégate*→REFINE→*frégate > bird*

A frigate can be a ship or a bird (both English and French have the same ambiguity for this word), and as a ship it can either be an ancient ship (with sails) or a modern one (with missiles and such). As can be seen in the above example, word refinements are organized as a decision tree, which can have some advantages over a flat list of word meanings for lexical disambiguation.

A given word sense is treated as any standard term; it can be played regularly. The general polysemous term contains (in principle) the union set of all possible relations given by the senses. In practice, we proceed the other way around, trying to distribute relations from the appropriate term to the proper senses.

2.5 *Negative relations*

A given relation is weighted, and the weight could be negative. A negative weight is only the result of some contributive process (i.e. it is never an outcome of the games) where volunteers add information to the lexical network. The purpose of negative weights is to give some foundation to the inhibitory process that allows us to reject (instead of select) some given meaning during a Word Sense Disambiguation task.

- *frégate > navire* →REFINE→ *coque (Eng. hull)*
- *frégate > bird* →REFINE$_{<0}$→ *coque*

Consider the sentence (in English): *The frigate had her hull breached.* Obviously, the negative relations immediately forbid the frigate from being a bird in this sentence. Hence, negative relations are of primary interest for representing contrastive phenomena among the various senses of a given term.

2.6 *Aggregate nodes*

The JDM lexical network also contains aggregate nodes that are inferred from the set of relations produced by players and contributors. An aggregate (node) is a node that encompasses either:

- a predicate (a verb) + one argument, like for example:
  lion [AGENT] *eat*,
  eat [PATIENT] *salad*.

- a noun + one feature, like for example:
  cat [CARAC] *black*,
  cat [LOCATION] *sofa*,
  rabbit [MADE-OF] *chocolate*.

Aggregates can be combined recursively, for example (parentheses are given for for the purpose of readability):

A :: (*cat [CARAC] black*) [AGENT] *eat*
B :: (*cat [CARAC] black*) [AGENT] (*eat [PATIENT] mouse*)

The motive of such aggregate nodes is to associate information (through relations) with some contextualized items:

- *A* →PATIENT→ *bird*
- *B* →LOCATION→ *garden*

The choice of aggregate node depends on the weight of the relations in the lexical network. An automated process will randomly select some relations and propose them as the aggregate to the players. Those which are selected for playing are dubbed as interesting and reified (instantiated as node) in the lexical network. For example, the relation:

soldier →AGENT→ *kill*
it could lead to the aggregated node:
soldier [AGENT] *kill*

can be proposed to player with various relation types to fill, as PATIENT, LOCATION, MANNER, INSTRUMENT, etc.

2.7 *Some figures*

By February 2017, the JDM lexical network contained roughly 67 million relations between more than 1 million nodes. Around 24 000 terms are refined into 65 000 word senses (word usages). More than 800 000 relations are negative and can be used as inhibitory items. The generic 'associated ideas' relations represent around 25% of the relation total. Annotations (see below) represent around 4.5% of the total. Informational relations (like part-of-speech, some conceptual values like HUMAN, ALIVE, PLACE, SUBSTANCE, ARTIFACT, etc.) stand for 20%.

3 INFERRING AND ANNOTATING RELATION

Inference is the process of proposing new relations on the basis of the actual contents of the network. Simple procedures tend to provide correct but mostly irrelevant results. In Sajous *et al.* (2013) an endogenous enrichment of Wiktionary is done with the use of a crowdsourcing tool. A similar approach of using crowdsourcing has been considering by (Zeichner *et al.* (2012)) for evaluating inference rules that are discovered from texts.

In what follows, we describe two endogenous inference mechanisms which assist the annotation spreading, although other schemas are running in the inference engine, producing new relations and deriving benefit from the produced annotations (Zarrouk 2015).

3.1 *Inference*

In order to increase the number of relations inside the JDM network, an inference engine proposes relations to be validated by other human contributors (or experts in the case of specialized knowledge). The core ideas about inferences in our system are the following:

- as far as the engine is concerned, inferring is deriving candidate conclusions (in the form of relations between terms) from previously known ones (existing relations);
- candidate inferences may be logically blocked regarding the presence or absence of some other relations;

- candidate inferences can be filtered out on the basis of a strength evaluation.

3.1.1 Deductive scheme

The first type of inference we are describing is the deduction or top-down scheme, which is based on the transitivity of the ontological relation *is-a* (hypernym). If a term A is a kind of B and B holds a relation R with C, then we can expect that A holds the same relation with C. The scheme can be formally written as follows:

(1) $$A \xrightarrow{is-a} B \quad \wedge \quad B \xrightarrow{R} C \quad \Rightarrow \quad A \xrightarrow{R} C$$

If we consider a term T with a set of weighted hypernyms, for each hypernym, the inference engine deduces a set of inferences. Those inference sets are not disjoint in the general case, and the weight of a proposed inference in several sets is the incremental geometric mean of each occurrence.

Of course, the scheme above is far too naive, especially considering the resource we are using. Indeed, B may be, possibly, a polysemous term and ways to block inferences that are certainly wrong can be devised. If there are two different meanings of term B that hold between the first and the second relation (Figure 3), then the inference is most likely wrong.

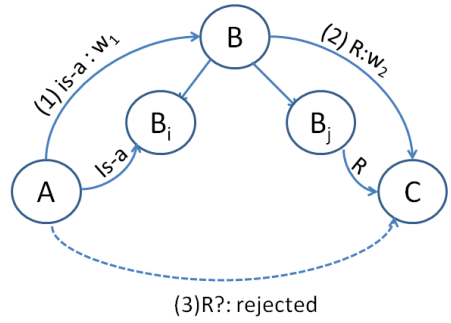


Figure 3: Triangular inference scheme with logical blocking based on the polysemy of B

Moreover, if one of the premises is tagged as *true but irrelevant*, then the inference is blocked. It is possible to assess a confidence level for each produced inference in a way that dubious inferences can be filtered out. The weight *w* of an inferred relation is the geometric mean of the weight of premises. If the second premise has a negative value,

the weight is not a number and the proposal is discarded. As the geometric mean is less tolerant of small values than the arithmetic mean, inferences which are not based on two valid relations (premises) are unlikely to go through.

3.1.2 Induction scheme

As for the deductive inference, induction exploits the transitivity of the relation *is-a*. If a term B is a hypernym of A and A holds a relation R with C, then we might expect that B could hold the same type of relation with C.

(2) $$A \xrightarrow{is-a} B \quad \wedge \quad A \xrightarrow{R} C \quad \Rightarrow \quad B \xrightarrow{R} C$$

This schema is a generalization inference. The global processing is similar to the one applied to the deduction scheme and similarly some logical and statistical filtering may be undertaken. The term joining the two premises is possibly polysemous. If the term A presents two distinct meanings which hold respectively of the premises (Figure 4), then the inference done from that term may be probably wrong.

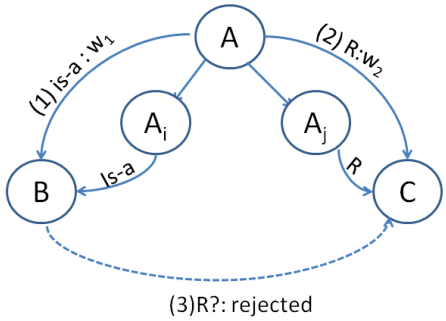


Figure 4:
Induction scheme. Central Term A may be polysemous with meanings holding premises, thus inducing a probably wrong relation

3.2 *Relation annotations*

JDM is a combined lexical semantic network (i.e one containing both general knowledge but also specialist domain knowledge). Besides being typed, relations are weighted and directed. In general, and especially in cases of specialized knowledge, the correlation between the weight of the relation and its importance is not strict. This is why it seems interesting to introduce annotations for some relations as these can be of great help in such areas as medicine, for instance.

In information retrieval, this annotation can be helpful to the users. For instance, in the field of medicine, practitioners may want to know if the characteristic of a given pathology is rare or frequent. For example, the relation between *measles* and *children* is *frequent* and as such will be available in the network.

3.3 *Annotation values*

These annotations will have a filter function in the inference scheme. The types of annotations are of various kinds (mostly frequency and relevance information). The different main annotation labels are:

- frequency annotations: very rare, rare, possible, frequent, always true;
- usage annotations: often believed true, language misuse;
- quantifier: any number like 1, 2, 4, etc. or many, few;
- qualitative: pertinent, irrelevant, inferable, potential, preferred.

Concerning language misuse, a doctor can use the term flu (illness) instead of virus of influenza: it is a misuse of language as the doctor makes use of a "language shortcut". The annotation often believed true is applied to a wrong relation. This is very often considered true, for instance, *spider* (is-a/often believed true) *insect*. This kind of annotation could be used to block the inference scheme. Qualitative annotation relates to the inferable status of a relation, especially concerning inference. The pertinent annotation refers to a proper ontological level for a given relation. For instance: *living being* (charac/pertinent) *alive* or *living being* (can/pertinent) *die*. Another case concerns synonyms: in this case, it may be relevant to choose a preferred synonym, as in the case of hepatocellular *carcinoma* (preferred), *HCC*, *malignant hepatoma*.

The annotation **inferable** is used when a relation is inferable (or has been inferred) from an already existing relation. For example: *cat* (charac/inferable) alive because *cat* (is-a) *living being*.

The annotation **potential** may be used for terms above the pertinent ones in the ontological hierarchy, for example: *bird* (has-part/always true) *wings* and *animal* (has-part/potential) *wings*.

Finally, the annotation **irrelevant** is used for a valid relation that is considered as too far below the pertinent level, for example, *animal* (has-part/irrelevant) *atoms*.

The annotation **quantifier** represents the number of parts of an object. Each human has two lungs so the quantifier relation there is 2. This kind of annotation is not necessarily a numeral, but can be more or less a subjective value, e.g. *few, many*, etc.

The annotation **frequency** can be of five different types: *always true, frequent, possible, rare and exceptional*. There are also two qualitative types (*pertinent and irrelevant*).

The first annotations have been introduced manually, but with the help of the inference scheme, they will spread through the network. We assign empirical values to each annotation's label: 4 to always true, 3 to frequent, 2 to possible, 1 to rare and 0 to the rest of the annotations. These allow us to select annotations to facilitate or block an inference scheme.

The annotation **possible** is a special case. Depending of the configuration of the system, it may block (stricter approach) or not block (lenient approach) the inference mechanism. If a system is lenient, we may obtain many inference proposals that might be wrong (high recall, low precision). On the other hand, if a system is strict, we reduce the risk of wrong proposals, but at the cost of missing adequate ones (low recall, high precision).

4 FROM JDM TO MTTS

In this section, we show how we can exploit the richness of the lexico-semantic information found in JDM, in order to decide on the typing ontology and assign types to objects in a compositional semantics framework that is richly typed. But before we get into this discussion, a very brief intro to MTT semantics.

4.1 *A gentle and brief intro to MTT semantics*

We use the term Modern Type Theory (MTT) to refer to a variant of a class of type theories as studied by Martin-Löf (1975); Martin-Löf (1984) and others, which have dependent types, inductive types and other powerful and expressive typing constructions. In this paper, we are going to employ one of these variants, namely the Unified Theory of dependent Types (UTT) complemented with the coercive subtyping mechanism (Luo 1994, 1999; Luo *et al.* 2012). Given the different typing constructions found in MTTs, various interpretations of linguistic

semantics might be different than what we usually find in traditional Montagovian formal semantics based on simple type theory.

4.1.1 Common nouns as types and subtyping

A key difference between MTT-semantics and Montague semantics (MS) lies in the interpretation of common nouns (CNs). In Montague (1974), the underlying logic, i.e. Church's simple type theory (Church 1940), is 'single-sorted' in the sense that there is only one type, e, of all entities. The other types such as the type of truth values, i.e. t, and the function types generated from types e and t do not stand for types of entities. Thus, no fine-grained distinctions between the elements of type e exist, and as such all individuals are interpreted using the same type. For example, *John* and *Mary* have the same type in simple type theory, i.e. the type e of individuals. An MTT, on the other hand, can be regarded as a 'many-sorted' logical system in that it contains many types. In this respect, MTTs can make fine-grained distinctions between individuals and use those different types to interpret subclasses of individuals. For example, one can have *John* : *man* and *Mary* : *woman*, where *man* and *woman* are different types. Another very basic difference between MS and MTTs is that common nouns in MTTs (CNs) are interpreted as *types* (Ranta 1994) rather than sets or predicates (i.e., objects of type $e \rightarrow t$) as in MS. The CNs *man, human, table* and *book* are interpreted as types *man, human, table* and *book*, respectively. Then, individuals are interpreted as being of one of the types used to interpret CNs.

This many-sortedness has the welcome result that a number of semantically infelicitous sentences, which are however syntactically well-formed, like e.g. *the ham sandwich walks* can be explained easily. This is because a verb like *walks* will be specified as being of type *Animal* → *Prop* while the type for *ham sandwich* will be *food* or *sandwich*:[6]

[6] This is of course based on the assumption that the definite NP is of a lower type and not a Generalized Quantifier. Furthermore, the idea that common nouns should be interpreted as types rather than predicates has been argued in Luo (2012) on philosophical grounds as well. There, Luo argues that the observation found in Geach (1962) according to which common nouns, in contrast to other linguistic categories, have criteria of identity that enable them to be compared, counted or quantified, has an interesting link with the constructive notion of

(3) *the ham sandwich* : *food*

(4) *walk* : *human* → *Prop*

Interpreting CNs as types rather than predicates has also a significant methodological implication: compatibility with subtyping. For instance, one may introduce various subtyping relations by postulating a collection of subtypes (physical objects, informational objects, eventualities, etc.) of the type *Entity* (Asher 2012). It is a well-known fact that if CNs are interpreted as predicates as in the traditional Montagovian setting, introducing such subtyping relations would cause problems given that the contravariance of function types would predict that if $A < B$, then $B \rightarrow Prop < A \rightarrow Prop$ would be the case. Substituting A with type *man* and B with type *human*, we come to understand why interpreting CNs as predicates is not a good idea if we want to add a coercive subtyping mechanism.

The subtyping mechanism used in the MTT endorsed in this paper is that of coercive subtyping (Luo 1999; Luo *et al.* 2012). Coercive subtyping can be seen as an abbreviation mechanism: A is a (proper) subtype of B ($A < B$) if there is a unique implicit coercion c from type A to type B and, if so, an object a of type A can be used in any context $\mathfrak{C}_B[\_]$ that expects an object of type B: $\mathfrak{C}_B[a]$ to be legal (well-typed) and equal to $\mathfrak{C}_B[c(a)]$.

To give an example: assume that both *man* and *human* are base types. One may then introduce the following as a basic subtyping relation:

(5) *man* < *human*

4.1.2 Σ-types, Π-types and universes

In this subsection, the dependent types Σ and Π. as well as universes are briefly introduced.

Dependent Σ-types. One of the basic features of MTTs is the use of Dependent Types. A dependent type is a family of types that depend

set/type: in constructive mathematics, sets (types) are not constructed only by specifying their objects but they additionally involve an equality relation. The argument is then that the interpretation of CNs as types in MTTs is explained and justified to a certain extent. Extensions and further theoretical advances using the CNs as types approach can be found in Chatzikyriakidis and Luo (2017b).

on some values. The constructor/operator Σ is a generalization of the Cartesian product of two sets that allows the second set to depend on the values of the first. For instance, if *human* is a type and *male* : *human* $\rightarrow$ *Prop*, then the Σ-type Σh : *human*. *male*(h) is intuitively the type of humans who are male.

More formally, if A is a type and B is an A-indexed family of types, then $\Sigma(A,B)$, or sometimes written as $\Sigma x{:}A.B(x)$, is a type, consisting of pairs (a, b) such that a is of type A and b is of type $B(a)$. When $B(x)$ is a constant type (i.e., always the same type no matter what x is), the Σ-type degenerates into the product type $A \times B$ of non-dependent pairs. Σ-types (and product types) are associated projection operations π_1 and π_2 so that $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$, for every (a, b) of type $\Sigma(A,B)$ or $A \times B$.

The linguistic relevance of Σ-types can be directly appreciated once we understand that in their dependent case Σ-types can be used to interpret linguistic phenomena of central importance, like adjectival modification (see for example Ranta 1994). To give an example, *handsome man* is interpreted as Σ-type (6), the type of handsome men (or more precisely, of those men together with proofs that they are handsome):

(6) Σm : *man*. *handsome*(m)

where *handsome*(m) is a family of propositions/types that depends on the man m.

Dependent Π*-types.* The other basic constructor for dependent types is Π. Π-types can be seen as a generalization of the normal function space where the second type is a family of types that might be dependent on the values of the first. A Π-type degenerates to the function type $A \rightarrow B$ in the non-dependent case. In more detail, when A is a type and P is a predicate over A, $\Pi x{:}A.P(x)$ is the dependent function type that, in the embedded logic, stands for the universally quantified proposition $\forall x{:}A.P(x)$. For example, the following sentence (7) is interpreted as (8):

(7) Every man walks.

(8) Πx : *man.walk*(x)

Π-types are very useful in formulating the typings for a number of linguistic categories like VP adverbs or quantifiers. The idea is that

adverbs and quantifiers range over the universe of (the interpretations of) CNs and as such we need a way to represent this fact. In this case, Π-types can be used, universally quantifying over the universe CN. Example (9) is the type for VP adverbs[7] while (10) is the type for quantifiers:

(9) $\Pi A : \mathrm{CN}.\ (A \to \mathit{Prop}) \to (A \to \mathit{Prop})$

(10) $\Pi A : \mathrm{CN}.\ (A \to \mathit{Prop}) \to \mathit{Prop}$

Further explanations of the above types are given after we have introduced the concept of type universe below.

Type Universes. An advanced feature of MTTs, which will be shown to be very relevant in interpreting NL semantics in general, is that of universes. Informally, a universe is a collection of (the names of) types put into a type (Martin-Löf 1984).[8] For example, one may want to collect all the names of the types that interpret common nouns into a universe CN : **Type**. The idea is that for each type A that interprets a common noun, there is a name $\overline{A}$ in CN. For example,

$$\overline{[\![man]\!]} : \mathrm{CN} \quad \text{and} \quad T_{\mathrm{CN}}(\overline{[\![man]\!]}) = [\![man]\!].$$

In practice, we do not distinguish a type in CN and its name by omitting the overlines and the operator T_{CN} by simply writing, for instance, $[\![man]\!] : \mathrm{CN}$.

Having introduced the universe CN, it is now possible to explain (9) and (10). The type in (10) says that for all elements A of type CN, we get a function type $(A \to \mathit{Prop}) \to \mathit{Prop}$. The idea is that the element A is now the type used. To illustrate how this works let us imagine the case of the quantifier *some* which has the typing in (10). The first argument we need has to be of type CN. Thus *some human* is

[7] This was proposed for the first time in Luo (2011).

[8] There is quite a long discussion on what properties these universes should have. In particular, the debate is largely concentrated on whether a universe should be predicative or impredicative. A strongly impredicative universe U of all types (with $U : U$ and Π-types) has been shown by Girard (1971) to be paradoxical, and as such logically inconsistent. The theory UTT we use here has only one impredicative universe Prop (representing the world of logical formulas) together with infinitely many predicative universes which as such avoids Girard's paradox (Luo 1994).

of type (*human* → *Prop*) → *Prop* given that the *A* here is *human* : CN (*A* becomes the type *human* in (*human* → *Prop*) → *Prop*). Then given a predicate like *walk* : *human* → *Prop*, we can apply *some human* to get *some human* : *Prop*.

4.2 *Getting MTT typings from JDM*

In this section, we will show how we can define a translation procedure between JDM and MTTs, in order to base our typing judgments and other related lexico-semantic information in JDM. We show some basic examples in which this can be done.

4.2.1 Base types and instances of base types

MTTs, as already said, are many-sorted systems in that they involve a multitude of types rather than just one monolithic type *e* domain of entities. In the accounts proposed by Luo (2011, 2012), every common noun is associated with a base type. What this idea amounts to, among other things, is that in this approach, CNs are base types and as such, are clearly separated in terms of their formal status with either adjectives or intransitive verbs. The type of CNs, like *Man*, *Human* and *Animal* is CN, the universe of common nouns.

The idea is then to extract these base types from common nouns in JDM (terms in JDM). POS tagging of JDM will provide information about which words are the common nouns. What we further have to do in getting the base types, is to exclude instances of terms (for example *John* as an instance of *Man*) in order to distinguish between instances of terms and the terms themselves (CNs).[9] This can be done by excluding named entities (NEs). The second part of the conjunction takes care of that by not allowing *A* to be an instance, i.e. an NE:[10]

(11) $\forall A.POS(N,A) \wedge \neg(Ins(A)) \Rightarrow A{:}\text{CN}$.

[9] This does not mean that we are not interested in instances. On the contrary. What we are saying here is that this rule distinguishes between CNs and instances of these CNs (the difference between a type like Man and an instance of this type, e.g. $John$). There will be a separate rule to derive instances.

[10] Note that modified CNs are also going to be of type CN. To give an example, consider the analysis of adjectival modification. In MTTs, this would be a Σ type, where the first component would be an element *A* of type CN and the second projection a predicate over *A*. The first projection is defined as a coercion, and thus the modified CN can be used as being of type CN. For more information on this, please refer to (Chatzikyriakidis and Luo 2013, 2017a) for more information.

Hyponym and hypernym (noted as *isa* in JDM) relations naturally correspond to subtypes and supertypes. We only use the subtype relation in order to provide a translation procedure:

(12) $\forall A, B.Hyp(A, B) \Rightarrow A < B{:}\mathrm{CN}$.

(13) $\forall A, B.Hyper(A, B) \Rightarrow B < A{:}\mathrm{CN}$.

This basically means that as soon as you have, let us say, a hyponym relation, e.g. $Hyp(A, B)$, this will be translated into a type-theoretic judgment of the following form:

(14) $A < B{:}\mathrm{CN}$

If we want to be more meticulous, we first have to judge A and B as being of type CN and then we can further add the subtype relation.

Moving on to synonyms, these can be defined using equality:[11]

(15) $\forall A, B.Syn(A, B) \Rightarrow A = B{:}\mathrm{CN}$.

Synonymicity is not only relevant for CNs but for other linguistic categories. We can encode this intuition as follows:

(16) $\forall A, B.Syn(A, B) \Rightarrow A = B{:}C(CLType)$

The above rule can declare synonymous words that have the same type via the equality relation. The type itself belongs in the universe *LType*. *LType* can be seen as a universe of linguistic types. The main intuition is that it includes the types instantiated in linguistic semantics (CN, adjectival and verbal types, types for quantifiers etc.). The interested reader is directed to Chatzikyriakidis and Luo (2012) for more details as well as some of the introduction rules for *LType*).

For instances of terms, such as proper names, we define the following:[12]

(17) $\forall A.\exists B.Ins(A, B) \Rightarrow A{:}B$

This means that if A is an instance of B, then A is of type B. For example, if *Einstein* is an instance of *person*, then what we get is

[11] Of course, this will treat A and B as perfect synonyms. We make this simple-minded assumption in this paper, even though perfect synonyms do not really exist in natural language.

[12] Note that here we overload the notation and sometimes treat *Ins* as an one place predicate and sometimes like a two place predicate.

Einstein:*person* with *person*:CN. In more detail, the procedure is as follows: given an instance A of a term B, first you declare B:CN and then judge the instance A to be of that type, i.e. B. This is the easy straightforward case and assumes that every instance will be an instance of one term. However, things are more complicated in practice. Given that JDM is a very elaborate lexical network, proper names will be instances of many terms (and thus, in MTT terms, types). To give an example: in the case of a proper name like *Einstein*, what we get is a number of terms from JDM that Einstein is an instance of: *physicist*, *scientist*, *human individual*. The question is which one do we choose. This is not an easy question to answer. One option would be to go for the term that is the most specific. But how do we define this? One way to do this, and given the discussion on relations of hyponymy, is to define it by saying that the term chosen should not have any subtypes in the given entry. For example *individual* will have subtypes (*scientist*, and also *physicist*) *scientist* (*physicist*). In this case, we are left with *physicist*. This is one way to do it. Note that given subtyping, we do get that *Einstein* is also an instance of the supertypes. This is a viable solution provided that all the terms are somehow connected in terms of subtyping. But there might be discontinuous relations. For example, imagine the case of the term *man*. Let us assume that *Einstein* is an instance of this term (surprisingly the term does not arise in JDM). Now, *physicist* and *scientist* are not subtypes of *man*. In this case, it seems that one has to make a decision about the type of *Einstein* based on those two types. It seems to us that in principle one should be able to make use of both types depending on the context. How one disambiguates is another issue however. Another way to do this is to assume that such instances are complex types, and treat them as disjoint union types in type-theoretical terms. Doing so will mean that *Einstein* will be of the following complex type *physicistman*:

(18) $physicistman = physicist + man$

(19) $Einstein{:}physicist + man$

Now, in this situation one can have such a complex case without actually resorting to context. The correct type will be disambiguated according to what is needed. In case a term of type *man* is needed like in *Einstein was a bachelor*, then the type *man* is going to be used. In cases like *Einstein was a well-known physicist*, the type *physicist* is to be

used. Note that this relies on the assumption that a subtyping mechanism is at play which will provide us with the following subtyping relations:

(20) $physicistman < man$

(21) $physicistman < physicist$

In this context, a more general way of translating these cases into MTTs would be as follows:

$$\forall A, D.\exists B, C.Ins(A,B) \Rightarrow \begin{cases} \text{A:B} & \text{iff } Ins(A,D) \wedge \text{B} < \text{D} \\ \text{A:B} + \text{C}, & \text{iff } Ins(A,C) \wedge Ins(A,D) \rightarrow \neg\, (\text{B} < \text{D} \vee \text{C} < \text{D}) \end{cases}$$

The first case is trivial. The second case says that each A that is an instance of type B has the type $B + C$, in case it is also an instance of C and for every other type D that A is an instance of, it is not the case that either B or C are subtypes of D.

4.2.2 Predicates and world knowledge information

The next question is, how can one extract information on the type of predicates, like for example verbs. JDM provides loads of information with every word, for example characteristics, synonyms, antonyms, collocations. For verbs, *agent*, *patient* and thematic relations in general are defined. This is particularly helpful for a rich type theory like the one used here, since predicates also make use of type many-sortedness. Thus, *walk* will be defined as being a function from type *animal* to propositions, *black* from type *object* to propositions and so on:

(22) $walk{:}animal \rightarrow Prop$

(23) $black{:}object \rightarrow Prop$

The information in JDM is enough to provide MTT typings for predicates as well. In JDM, as already said, one can look at semantic relations like *action*, *patient*, dubbed as predicative relations in the classification given in the previous section, and various other such relations. For example, *man* appears as the agent of a number of verbs that express actions, e.g. *question*. But, the most helpful relation is the inverse agent/theme/patient relation, $agent^{-1}$. This relation returns a list of terms (and instances of terms) that can function as the agent for the action denoted by the verb. For example, the verb *question* will

involve among others *teacher, mother, child, daugther, person, human*. How can we make sense in order to provide typings in MTTs? There is a straightforward way to do this. What we need is to find the most general term, i.e. the term of which all the other terms are hyponyms. Instances of terms are not needed in this process.[13]

(24) $\forall A, B.\exists C.Ag(A,B) \wedge (Hyp(A,C) \vee (A = C)) \Rightarrow B{:}C \rightarrow Prop$

However, things may be less straightforward than that simply because there exists no term in the *agent*$^{-1}$ relation that is a supertype of all the others. In this case, we see two plausible options: a) introduce a supertype or b) split the refinements into different classes. For example in case we have refinements *human*, *man*, *pilot*, *vehicle*, *car*, *bike*, we can split this into class A $=$ *pilot, man* $<$ *human* and class $B =$ *bike, car* $<$ *vehicle* and propose an overloaded polysemous type for the verb in question, with two different typings, *human* $\rightarrow$ *Prop* and *vehicle* $\rightarrow$ *Prop*. As far as the supertype is concerned, the suggestion is that we go for a default supertype, which will be the supertype of all types. For example in the work of Chatzikyriakidis and Luo (2014a), this type is *object*. We can think of such a type, no matter whether we agree that this should be type *object* or not (denoted as *Toptype* below). Then, with these considerations in mind, we may want to update the previous correspondence:

$$\forall A, B.Ag(A,B) \Rightarrow \begin{cases} \text{B:C} \rightarrow Prop \text{ iff } \exists \text{ C.}((Hyp(\text{A,C}) \wedge Ag(\text{C,B})) \vee (\text{A} = \text{C})) \\ \text{B:}D \rightarrow Prop \text{ iff } \neg\exists C.(Hyp(A,C) \wedge D = TopType) \end{cases}$$

The first condition says that if A is the agent of predicate B (expressed by an adjective here), then in case there is a type C that is also the agent of B, it is a supertype of A as well as a supertype of all other types that are agents of B,[14] or if C is actually A, then the type for the adjective will just be a predicate over C. In case there is no hypernym of A, we choose as our type a predicate over the Toptype (the type of which all other types are subtypes).

[13] The formula reads as follows: for all A and B, where A is an agent of B (so B is a predicate), if there exists a C such than all A are either hyponyms of C or are equal to C, the predicate $C \rightarrow Prop$ is returned.

[14] This last bit is not actually encoded in the rule for formatting reasons. The following condition is implicit: $\forall$ E.*Ag*(E,B)$\rightarrow$*Hyp*(E,C).

Moving on to adjectives, similar processes can be defined. However, this time we look at another relation, called *carac* (characteristic) that denotes a characteristic of a term. For example, for $grand$, we find the characteristics *chose* and *homme*, 'object' and 'man' respectively among others. There are two ways to assign types for adjectives here: a) propose a type using the same reasoning for predicates above or b) propose a polymorphic type extending over a universe which includes the most general type found satisfying the characteristic in question (e.g. blackness, bigness, etc.), along with its subtypes:[15]

(25) $\forall A,B.\exists C.Car(A,B) \wedge (Hyp(A,C) \vee (A = C)) \Rightarrow C \rightarrow Prop$

(26) $\forall A,B.\exists C.Car(A,B) \wedge (Hyp(A,C) \vee (A = C)) \Rightarrow \Pi U{:}\mathrm{CN}_C.U \rightarrow Prop$

Using a polymorphic universe in terms of inference will suffice in order to take care of the class of adjectives known as subsective (e.g. *skilful*), while for intersective adjectives (e.g. *black*) a non-polymorphic type is needed. This, along with the use of Σ types for modified, by adjectives, CNs (e.g. *black man*, *skillful surgeon* etc.) will suffice to take care of the basic inferential properties of the two classes of adjectives Chatzikyriakidis and Luo (2013, 2017a) for more details on this analysis). However, as in the case of verbs, more should be said in order to take care of the complications already discussed for verbs previously. Taking these issues into consideration, the updated rule is as follows:

$$\forall A,B.Car(A,B) \Rightarrow \begin{cases} \mathrm{B{:}C} \rightarrow Prop & \text{iff } \exists\ \mathrm{C}.((Hyp(\mathrm{A,C}) \wedge Ag(\mathrm{C,B})) \vee (\mathrm{A = C})) \\ \mathrm{B{:}}D \rightarrow Prop & \text{iff } \neg\exists C.(Hyp(A,C) \wedge D = TopType) \end{cases}$$

$$\forall A,B.Car(A,B) \Rightarrow \begin{cases} \mathrm{B{:}}\Pi U{:}CN_C\ U \rightarrow Prop & \text{iff } \exists\ \mathrm{C}.((Hyp(\mathrm{A,C}) \wedge Ag(\mathrm{C,B})) \vee (\mathrm{A = C})) \\ \mathrm{B{:}}\Pi U{:}CN_D\ U \rightarrow \mathrm{Prop} & \text{iff } \neg\exists C.(Hyp(A,C) \wedge D = TopType) \end{cases}$$

The above two rules are for intersective and subsective adjectives. Hyponymy relations between adjectives can be encoded as meaning postulates.

(27) $\forall A,B.POS(adj,A) \wedge POS(Adj,B) \wedge Hyp(A,B) \Rightarrow \forall x{:}C.A(x) \rightarrow B(x)$
where $A,B{:}C \rightarrow Prop$

[15] For example the universe $U{:}\mathrm{CN}_C$ will contain the type C along with its subtypes.

Due to the abundance of information that JDM has to offer, one can further encode different sorts of information in the form of axioms or definitions. For example the *has_part* relation, in effect a mereological relation, can be translated as a *part of* relation with *part_of*:*object* → *object* → *Prop* and follow a translation procedure along standard assumptions for mereology for formal semantics[16]. There are many more interesting relations in JDM, like for example the collocation relation, (*locution* in JDM) or the magnifying and its inverse anti-magnifying relation, *magn* and *anti-magn* respectively. Now, there is no clear way of what we can do with these relations. One can of course just encode a similar relation in the type-theoretic language used, but the question is what do we gain in terms of reasoning for example, by doing so. For instance, looking at the entry for *homme* 'man', we see a number of collocations like *homme grande* and *homme libre*. The collocations that involve adjectival modification most of the times give rise to subsective inferences. For example a great man is a man and a free man is also a man. It would be tempting in this respect to treat these cases as subtypes of the term. In this case, we allow some non-compositionality and treat collocations as involving one word. Of course, this will not give us the correct results in all cases. For example, think of the term *objet* 'object'. Among the collocations in the category discussed, e.g. *object du désir* or *objet de curiosité*, there are also collocations like *programmation orientée objet* 'object oriented programming' that will give us the wrong results if we go the subtyping route. One can however decide on whether to take such a stance based on the amount of collocations that can be correctly captured in going the subtype route in relation to those that are not. This is a complex issue with which we will not deal in this paper.

4.2.3 Polysemy

The next issue we want to look at is polysemy, more specifically the translation process in case of polysemous terms. First of all, we have to note here that JDM does not distinguish between homophony and polysemy in the sense these are usualy understood in the literature on formal semantics (e.g. *bank* as homophonous and *book* as polysemous). For JDM, there is only one term to refer to both homophony

[16] For an overview see Champollion and Krifka (2016).

and polysemy, and this is polysemy. This is what we are going to use here as well, a single notion for all cases where different meanings associated with a given word are found. For JDM, there is this first level where words with more than one meaning (irrespective of whether the meanings are related or not) are dubbed as polysemous, and then additional levels of refinement are provided. In MTTs, as in formal semantics in general, there are different treatments with respect to cases of homophony and cases of polysemy. For example, in Luo (2011), homophony is treated via local coercions (local subtyping relations), while logical polysemy (cases like *book*) via introducing dot-types, types that encode two senses that do not share any components (Luo 2010). It is a difficult task to be able to translate from a polysemous term identified in JDM to the correct mapping in MTTs. However, there are some preliminary thoughts on how this can be achieved. First of all, let us look at some cases of polysemy identified in JDM that would not be considered such cases in mainstream formal semantics. For example the term *individual* is marked as polysemous in JDM. The reason for this is that JDM goes into more detail than what most formal semantics theories do. For example, JDM distinguishes different meanings of *individual* with respect to its domain of appearance, e.g. a different notion of individual is found in the domain of statistics, a different one in the domain of biology, and so on. This level of fine-grainedness is usually not found in formal semantics. However, there is no reason why we should not go into this level of detail in MTTs. In order to encode domains, we use type theoretic contexts as these have been used by Ranta (1994); Chatzikyriakidis and Luo (2014c), among others. The idea is that a relation can appear in different domains. If this is the case, then different relations might be at play depending on the domain. For example, different refinements of a term might be possible in a domain A than in a domain B.[17]

(28) $\forall A, C.\mathit{Domain}(C) \wedge \mathit{POS}(N, A)\ \mathit{in}\ C \wedge \neg(\mathit{Ins}(A)) \Rightarrow A{:}\mathrm{CN}$ in Γ_C

[17] An anonymous reviewer asks how does the equation help us in using this information. The idea is that as soon as the conditions are satisfied, i.e. there is a relevant domain for a given type declaration, then this declaration is made inside the relevant type theoretic context, e.g. the context of zoology, philosophy, etc. In the case of Coq, this can be done by introducing local sections.

The above example identifies a noun, which is not an instance, in a domain C and declares this to be of type CN in context Γ_C. All this information such as POS, domain and instance status is part of the JDM network. To give an example, take the term French term *fracture* (fracture) in JDM. This is associated with a number of different domains, let us mention two here, *géologie* and *médicine*. This will basically add the term *fracture* into two different contexts where the relations between *fracture* and other terms in the given context might differ in the different contexts. For example, one might have a term B being a subtype of *fracture* in one domain but not in the other:

What about other cases of polysemy like *book* or *bank*? One way to look at the translation process in these cases is the following: in case a term is dubbed polysemous in JDM, we look at the semantic refinements and introduce all these refinements as subtypes of the initial term:

(29) $\forall A, C.\mathit{POS}(N,A) \wedge \neg(\mathit{Ins}(A)) \wedge \mathit{Ref}(A,C) \Rightarrow A < C{:}\mathrm{CN}$

Now in order to decide whether we are going to use local coercions or dot-types we proceed as follows: the types that participate in dot-types are limited and enumerable:[18] some of these include *phy*, *info*, *event*, *inst* among others. We can thus create such a set of refinements that can be senses of a dot-type. Call this set *dot refinements*, DR. Now, in case the refinements happen to be members of this set then we can form a dot-type out of the individual refinements:

(30) $\forall A, B, C.POS(N,A) \wedge \mathit{Ref}(A(B,C)) \in DR \Rightarrow A{:}CN < B \bullet C$

Other cases of polysemy that should be taken into consideration involve cases where the two meanings are associated with different types (e.g. cases like *run*). In this case, we have at least two verbal meanings with a different verbal arity as well as a different CN argument. An easy way to do this is to just overload the types to take care of situations like these. For example, in Luo (2011), the polysemy of

[18] An anonymous reviewer asks how these types are chosen. This is not an easy question. For the needs of this paper, and given that dot-types have specific properties compared to other polysemous terms, the types comprising the dot-types are limited. We enumerate these types based on existing theoretical work on co-predication by Pustejovsky (1995) and Asher (2008), among others.

run is assumed to be captured using a Unit type which allows us to overload the type with the two different typings:

(31) $run_1 : human \rightarrow Prop$

(32) $run_2 : human \rightarrow institution \rightarrow Prop$

With this last note, we will move on to look how information from JDM can be used in order to perform reasoning tasks. What we are going to do is to look at simple cases of lexical semantics information extraction from JeuxdeMots, their direct translation to MTT semantics feeding the proof-assistant Coq. Reasoning is then performed using the assistant.

5 JDM, MTTS AND REASONING USING PROOF-ASSISTANTS

Coq is an interactive theorem prover (proof-assistant). The idea behind it and proof-assistants in general is simple and can be roughly summarized as follows: one uses Coq in order to check whether propositions based on statements previously pre-defined or user defined (definitions, parameters, variables) can be proven or not. Coq is a dependently typed proof-assistant implementing the calculus of Inductive Constructions (CiC, see Coq 2007). This means that the language used for expressing these various propositions is an MTT. To give a very short example of how Coq operates, let us say we want to prove the following propositional tautology in Coq:

(33) $((P \vee Q) \wedge (P \rightarrow R) \wedge (Q \rightarrow R)) \rightarrow R$

Given Coq's typed nature we have to introduce the variables P, Q, R as being of type *Prop* (P, Q, R:*Prop*). To get Coq into proof mode, we have to use the command *Theorem*, followed by the name we give to this theorem, followed by the theorem we want to prove:

(34) *Theorem A*: $((P \vee Q) \wedge (P \rightarrow R) \wedge (Q \rightarrow R)) \rightarrow R$

This will put Coq into proof mode:

```
Theorem A:((P\/Q)/\(P -> R)/\(Q->R))->R.
1 subgoal
============================
(P\/Q)/\(P -> R)/\(Q -> R)->R
```

Now, we have to guide the prover to a proof using its pre-defined proof tactics (or we can define our own). For the case under consideration, we first introduce the antecedent as an assumption using *intro*:[19]

```
A < intro.
1 subgoal
H :(P \/ Q)/\(P -> R)/\(Q -> R)
============================
R
```

We split the hypothesis into individual hypothesis using *destruct*:[20]

```
destruct H. destruct H0.
1 subgoal
H : P \/ Q
H0 : P -> R
H1 : Q -> R
============================
R
```

Now, we can apply the elimination rule for disjunction which will basically result in two subgoals:

```
elim H.
2 subgoals
H : P \/ Q
H0 : P -> R
H1 : Q -> R
============================
P -> R
subgoal 2 is:
Q -> R
```

The two subgoals are already in the hypotheses. We can use the *assumption* tactic that matches the goal in case an identical premise exists, and the proof is completed:

[19] This tactic moves the antecedent of the goal into the proof context as a hypothesis.

[20] After destructing H, we get $H0$ as $H0{:}(P \rightarrow R) \wedge (Q \wedge R)$.

```
assumption. assumption.
1 subgoal
H : P \/ Q
H0 : P -> R
H1 : Q -> R
============================
Q -> R
Proof completed.
```

Now, as we have already said, Coq implements an MTT. In this respect, Coq 'speaks' an MTT so to say. It is also a powerful reasoner, i.e. it can perform elaborate reasoning tasks. These two facts open up the possibility of using Coq for reasoning with NL using MTT semantics. Indeed, earlier work has shown that Coq can be used to perform very elaborate reasoning tasks with very high precision (Mineshima *et al.*; Bernardy and Chatzikyriakidis 2017). To give an example, consider the case of the existential quantifier *some*. Quantifiers in MTTs are given the following type, where A extends over the CN (this is reminiscent of the type used for VP adverbs):

(35) $\Pi A : \text{CN}.\ (A \rightarrow \textit{Prop}) \rightarrow (A \rightarrow \textit{Prop})$

We provide a definition based on this type, giving rather standard semantics for the existential quantifier (in Coq notation):

```
Definition some:=fun(A:CN)(P:A->Prop)=>exists x:A,P x.
```

This says that given an A of type CN and a predicate over A, there is an x:A such that P holds of x. Imagine, now, that we want to see the consequences of this definition. For example we may want to check whether *John walks* implies that *some man walks* or that *some man walks* implies that *some human walks*. We define, following our theoretical assumptions about CNs, man and $human$ to be of type CN and declare the subtyping relation $man < human$. The subtyping relations in Coq are declared by first introducing them as axioms and then coercing them:

```
Parameter man human: CN
Axiom mh: man -> human. Coercion mh: man >-> human.
```

This is all we need to get the above inferences. These assumptions suffice to prove these inferences in Coq. We formulate the theorem and put Coq into proof mode:

```
Theorem EX: walk John-> (some man) walk.
```

Unfold the definition for *some* and use *intro*

```
EX < intro.
1 subgoal
H : walk John
============================
exists x : man, walk x
```

Using the *exists* tactic to substitute x for $John$. Using *assumption* the theorem is proven. Now, what we want to show is that we can actually use JDM to extract lexical and typing information, translate this information into MTT semantics in the form of Coq code and then perform reasoning tasks. Let us look at the following example:

(36) John Fitzgerald Kennedy ate some gruyère

Suppose, now, that we further want to check whether the following is true:

(37) John Fitzgerald Kennedy ate some gruyère ⇒ John Fitzgerald Kennedy ate some cheese

Let us see whether we can extract this information from JDM. We use the JDM XML version, and further use simple Python code to extract the relevant information and turn it into Coq code. We first extract all the synonyms and subtypes of cheese and translate them to MTT semantics (in Coq code). The result is more than 200 subtypes for cheese (*fromage* in French), among them the type for *gruyère*. What the code does is that it first declares all subtypes to be of type CN and then further declares them to be subtypes of the CN in question (cheese in our case. The result is something like this (we use the first 5 subtypes to illustrate this):

```
Parameter gruyere:CN.
Parameter brie:CN.
Parameter kiri:CN.
```

```
Parameter camembert:CN.
Axiom  Gruyere:gruyere -> fromage.
Coercion Gruyere:gruyere>-> fromage.
Axiom Brie:brie->fromage.Coercion Brie:brie>->fromage.
Axiom Kiri:kiri->fromage.Coercion Kiri:kiri>->fromage.
Axiom Camembert:camembert->fromage.
Coercion Camembert:camembert>-> fromage.
```

The next step is to extract information about *John Fitzgerald Kennedy*. The only thing needed here is to extract the information for the instances of the type *man* (*homme* in French). Simple coding in Python can extract all the subtypes for *man* as well as its instances, declaring them as being of type *man*. What we get in doing so is the following (we only show the information relevant to our example):

```
Parameter man: CN.
Parameter John Fitzgerald Kennedy: man.
```

The next step is extracting the information for the verb *eat* (*manger* in French). Here we use a more simple and less elegant way of extracting the function types than we have described in the previous section. We first chose 6 very basic types, *woman, man, human, animal, food, object*. If any of these types is present as an agent argument (starting hierarchically from type *object* and all the way down to the other types), it is added as an argument to the function type. Thus, in case of a predicate which has an object agent, the type *object* → *Prop* is returned. The other types, even if present, are neglected. If *object* is not present, the next type is checked and so on. Doing so, we end up with the type *Animal* → *food* → *Prop* for *eat*. *Cheese* is of course a subtype of *food* (we get this from the hyponyms of *food*), and *human* of *animal*. So, the only thing left is a definition of the quantifier. Quantifiers and related elements can perhaps be assumed to belong to a closed set of words that can be given their semantics manually. This is what we do here by manually providing a definition for *some*. With this in place, what we get is the following information for Coq (only the relevant code to the example is shown):

```
Definition CN:= Set.
Definition some:= fun(A:CN)(P:A->Prop)=>exists x:A,P x.
Parameters man woman human animal food object: CN.
```

```
Axiom  Man:man->human. Coercion Man:man>->human.
Axiom  Human: human->animal.Coercion Human:human>->animal.
Axiom  Animal: animal->object.
Coercion Animal: animal>->object.
Axiom  Food: food->object.Coercion food:food>->object.
Axiom  Woman: woman->human. Coercion Woman:woman>->human.
Parameter gruyere: CN.
Axiom  Gruyere: gruyere->fromage.
Coercion Gruyere:gruyere>-> fromage.
Parameter John_Fitzgerald_Kennedy: human.
```

This is in fact enough to work through the inference we are interested in. Of course, this is not a very elaborate example, but it is a nice way to exemplify how information from a lexical network can be used in a compositional semantics framework to perform reasoning tasks. Note, that a number of other inferences also follow from the previous example:

(38) John Kennedy ate some gruyère ⇒ some man ate some gruyère.

(39) John Kennedy ate some gruyère ⇒ John Kennedy ate some food.

Let us look at another example:

(40) The frigate had its hull breached.

In this example, what we need to predict is that the *bird* sense cannot be used. On the contrary, we should predict that the *ship* sense is required. First of all, the way this is achieved in JDM is via using negative weights as we have mentioned in chapter 2. We will now see that compositional semantics can further help us in this task. We start with the assumption that frigate is not yet refined or can be either a *bird* or a *ship*. The next thing we have is an NP with a possessive pronoun. Following Ranta (1994), we assume a pronominalization and a genitive rule. The two rules are shown below (adapted from Ranta (1994):[21]

$$\frac{A{:}\text{CN} \quad a{:}A}{\begin{cases}\text{PRON(A,a):A} \\ \text{PRON(A,a) = a:A}\end{cases}} \qquad \frac{A{:}\text{CN} \quad B{:}\text{CN} \quad C(x{:}A, y{:}B) \quad a{:}A \quad b{:}B \quad c{:}C(a,b)}{\begin{cases}\text{Gen(A,B(x,y),C(x,y),a,b,c):B} \\ \text{Gen(A,B(x,y),C(x,y),a,b,c = b):B}\end{cases}}$$

[21] Ranta (1994) uses type *Set* instead of CN that we are using.

The result of sugaring in English will be *A's B*. The pronominalization rule will depend on the type that A will take. For example $\textit{Pron}(\textit{man}, a)$ will return *he*, $\textit{Pron}(\textit{woman}, a)$ *she* and $\textit{Pron}(\textit{object}, a)$ *it*. Returning to our example, the possessive *its* is a combination of the two rules we have presented, i.e. *Pron* and *Gen*. As we have said, the semantics for pronouns will depend on the value for CN. This is also the case obviously for *its*. Let us assume that A takes the value *frigate*. This will give us:

(41) $\textit{Gen}(\textit{frigate}, \textit{hull}(x, y), C(x, y), \textit{Pron}(\textit{frigate}, a), b\ \textit{hull}, c)\text{:}\textit{hull}.$

The C relation is an underspecified relation, since it can take different values, given the semantic polysemy of the genitive. Assuming that the relation involved in our example is one of meronymy, what we get is an elaboration of $C(a, b)$ to $\textit{has\_part}(a, b)$. Now, notice that JDM provides meronymy relation refinement between two objects, of which one is a ship and the other a hull, but not between a bird and a hull. Specifically, supplies us with the following information (translated into MTT semantics):

(42) $\forall a\text{:}\textit{ship}.\exists b\text{:}\textit{hull}.\textit{has\_part}(a)(b)$

But not:

(43) $\forall a\text{:}\textit{bird}.\exists b\text{:}\textit{hull}.\ \textit{has\_part}(a)(b)$

Parsing *The frigate had its hull breached*, what we get is the following (simplified):

(44) $\textit{breached}(\textit{the}(\textit{ship}, a))(\textit{Gen}(\textit{ship}, \textit{hull}(x, y), \textit{has\_part}(x, y),$
$\textit{Pron}(\textit{ship}, a), b\text{:}\textit{hull}, c))$

Now, we can assume that the negative weight amounts to the negation of the has_part relation:

(45) $\forall a\text{:}\textit{bird}\neg(\exists b\text{:}\textit{hull}.(\textit{has\_part}(a)(b)))$

If now, we substitute with bird and given the information associated with hull as a refinement of bird, what we will get is a contradiction:

(46) $\textit{breached}(\textit{the}(\textit{ship}, a))(\textit{Gen}(\textit{bird}, \textit{hull}(x, y), \textit{has\_part}(x, y),$
$\textit{Pron}(\textit{ship}, a), b\text{:}\textit{hull}, c)) \wedge \forall a\text{:}\textit{bird}.\neg(\exists b\text{:}\textit{hull}.(\textit{has\_part}(a)(b)))$

If we have a system that can spot contradictions between information derived from lexical semantics (in our case the negative weight translating into a meaning postulate) and information derived for semantic compositionality, we might use this in order to disambiguate word senses as well. For example, one can define a ranking algorithm that will rank the senses of a given word in a sentence depending on whether they give rise to contradictions between lexical semantics information and information derived for semantic compositionality. In this manner, one can seek to define a combined strategy to disambiguate using insights from both the lexical network itself as well as the formal system in which this information is encoded.

5.1 *Reasoning with missing premises: enthymematic reasoning*

It is a well-known fact that natural language inference (NLI) is not only about logical inference. Better put, logical inference is only part of NLI. Other kinds of non-logical inferencing is going in NLI, e.g. implicatures, presuppositions, or enthymematic reasoning to name a few. The latter form of reasoning is particularly important for the scope of this article, since enthymematic reasoning is basically deriving conclusions from missing premises or implicit premises. Consider the following classic case of an enthymeme:

(47) Socrates is human, therefore he is mortal

In this example, there is an implicit premise at play, namely that all humans are mortal. This is not given however. It is somehow presupposed as part of the general world knowledge. What would be interesting to see is to check whether such implicit arguments can be retrieved via the richness of a network like JDM. Indeed, this can be done. In particular, the entry for *mortal* in JDM, specifies *human* as one of its hyponyms. So, extracting lexical relations for *human* will also extract the synonym relation. Thus, it is easy to get the inference we are interested in. The same kind of information that can lead to retrieving the implicit argument in further examples like the ones shown below can be found using the richness of a network like JDM:[22]

(48) He coughs. He is ill.

[22] We are rather simplifying here, given that in JDM there is more than one relation between *human* and *mortal*. One finds the hyponym relation, the synonym relation as well as the characteristic relation (mortality as a characteristic

(49) She has a child, thus she has given birth.

Of course, it would be naive to think that enthymematic inference can be dealt with in full via using only information present in a lexical network, no matter how rich that network is. We are not suggesting anything like this. The interested reader that wants to have a deeper look at the issue of enthymemes and reasoning with them in a type-theoretic framework is directed to Breitholtz (2014). For the needs of this paper, it is enough to mention that at least some cases of enthymemes can be captured via a combination of lexical semantics information taken from a rich lexical network like JDM and their feeding to a richly typed logical system like the one we are endorsing in this paper.

6 CONCLUSION

In this paper we have looked at the way one can use information from a rich GWAP lexical network in order to construct typing ontologies for NL in rich type theories. Rich or modern type theories offer us elaborate typing structures and type many-sortedness, where the monolithic domain of individuals is substituted by a multitude of types. The problem that is created however, given this context, concerns which types need to be represented and which not, as well as the criteria that one uses in order to reach to such a decision. In this paper, we have proposed that one does not have to take such a decision but rather leave this information flow from a lexical network, in our case a rich GWAP network, JDM. We have proposed an initial way of doing this, namely extracting information from JDM w.r.t. to base types for common nouns as well as the types for other categories like verbs and adjectives. We have also proposed to use MTTs for several other types of information obtained from such a rich network. Lastly, we have initiated a discussion on how one can further use this

of humans). From a logical point of view, it cannot be the case that two terms are both synonyms and hyponyms. Furthermore, as one of the reviewer's notes, the synonym relation in JDM seems to be assymetrical, otherwise one would expect things like *pandas are human* to be inferred. This raises a more general issue, i.e. handling potentially contradictory information in the network. This is something that we will definitely explore in the future.

richness of information, especially common knowledge information, in order to deal with aspects of inference. On the one hand you have a wealth of lexical-semantic relations and on the other, a very rich and expressive compositional framework with powerful reasoning mechanisms. The result one would aim at, given this situation, is a combination of these two aspects in order to perform reasoning tasks with NLI. We have discussed some simple reasoning examples using the Coq proof-assistant, a proof-assistant that implements an MTT. Information is extracted from JDM and then translated into Coq code (thus into an MTT variant). The results are promising, showing a potential to deal with important aspects of NL reasoning. Furthermore, some easy cases of reasoning under implicit premises, i.e. enthymematic inference, were also shown to be captured via retrieving the implicit premises as lexical information associated with words appearing in the explicit premises. It is our hope that this work will initiate a more active discussion on the need for more fine grained frameworks for formal semantics as well as an active dialogue between people working on lexical networks and type theoretical (or logical in general) semantics from both a theoretical and an implementational point of view.

REFERENCES

Nicholas Asher (2008), A type driven theory of predication with complex types, *Fundamenta Informaticae*, 84(2):151–183.

Nicholas Asher (2012), *Lexical Meaning in Context: a Web of Words*, Cambridge University Press.

Christoph Benzmüller, Frank Theiss, and Arnaud Fietzke (2007), The LEO-II Project, in *Automated Reasoning Workshop*.

Jean-Philippe Bernardy and Stergios Chatzikyriakidis (2017), A Type-Theoretical system for the FraCaS test suite: Grammatical Framework meets Coq, ms, University of Gothenburg. `http://www.stergioschatzikyriakidis.com/uploads/1/0/3/6/10363759/iwcs_bercha.pdf`.

Ellen Breitholtz (2014), *Enthymemes in Dialogue: A micro-rhetorical approach*, Ph.D. thesis, University of Gothenburg.

Lucas Champollion and Manfred Krifka (2016), Mereology, in Paul Dekker and Maria Aloni, editors, *Cambridge Handbook of Semantics*, pp. 369–388, Cambridge University Press.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2012), An Account of Natural Language Coordination in Type Theory with Coercive Subtyping, in Y. PARMENTIER and D. DUCHIER, editors, *proceedings of Constraint Solving and Language Processing (CSLP12). LNCS 8114*, pp. 31–51, Orleans.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2013), Adjectives in a modern type-theoretical setting, in G. MORRILL and J.M NEDERHOF, editors, *Proceedings of Formal Grammar 2013. LNCS 8036*, pp. 159–174.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2014a), Natural Language Inference in Coq, *Journal of Logic, Language and Information.*, 23(4):441–480.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2014b), Natural Language Reasoning Using proof-assistant technology: Rich Typing and beyond, in *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pp. 37–45.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2014c), Using Signatures in Type Theory to Represent Situations, *Logic and Engineering of Natural Language Semantics 11. Tokyo*.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2017a), Adjectival and Adverbial Modification: The View from Modern Type Theories, *Journal of Logic, Language and Information*, 26(1):45–88.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2017b), *On the Interpretation of Common Nouns: Types Versus Predicates*, pp. 43–70, Springer International Publishing.

Alonzo CHURCH (1940), A Formulation of the Simple Theory of Types, *J. Symbolic Logic*, 5(1).

Allan M COLLINS and M Ross QUILLIAN (1969), Retrieval time from semantic memory, *Journal of verbal learning and verbal behavior*, 8(2):240–247.

Robin COOPER, Simon DOBNIK, Shalom LAPPIN, and Staffan LARSSON (2014), A probabilistic rich type theory for semantic interpretation, in *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pp. 72–79.

Coq 2007 (2007), *The Coq Proof Assistant Reference Manual (Version 8.1), INRIA*, The Coq Development Team.

Christiane FELLBAUM (1998), *WordNet: An Electronic Lexical Database*, MIT press.

Bruno GAUME, Karine DUVIGNAU, and Martine VANHOVE (2007), Semantic associations and confluences in paradigmatic networks, in Martine VANHOVE, editor, *Typologie des rapprochements sémantiques*, p. (on line), John Benjamins Publishing Company.

Peter GEACH (1962), *Reference and Generality: An examination of some Medieval and Modern Theories*, Cornell University Press.

Jean-Yves GIRARD (1971), Une extension de l'interpretation fonctionelle de Gödel à l'analyse et son application à l'élimination des coupures dans et la thèorie des types, *in proceedings of the 2nd Scandinavian Logic Symposium.* North-Holland, Amsterdam, pp. 63–92.

Mathieu LAFOURCADE (2007a), Making people play for Lexical Acquisition., in *SNLP 2007, 7th Symposium on Natural Language Processing. Pattaya, Thailande, 13-15 December 2007*.

Mathieu LAFOURCADE (2007b), Making people play for Lexical Acquisition with the JeuxDeMots prototype, in *SNLP'07: 7th international symposium on natural language processing*, p. 7.

Mathieu LAFOURCADE, Alain JOUBERT, and Nathalie. LE BRUN (2015), *Games with a Purpose (GWAPS)*, Focus Series in Cognitive Science and Knowledge Management, Wiley, ISBN 9781848218031.

Henry LIEBERMAN, Dustin SMITH, and Alea TEETERS (2007), Common Consensus: a web-based game for collecting commonsense goals., in *Workshop on Common Sense for Intelligent Interfaces, ACM Conferences for Intelligent User Interfaces (IUI 2007), Honolulu.*

Zhaohui LUO (1994), *Computation and Reasoning: A Type Theory for Computer Science*, Oxford University Press.

Zhaohui LUO (1999), Coercive subtyping, *Journal of Logic and Computation*, 9(1):105–130.

Zhaohui LUO (2010), Type-Theoretical Semantics with Coercive Subtyping, *Semantics and Linguistic Theory 20 (SALT20), Vancouver*.

Zhaohui LUO (2011), Contextual analysis of word meanings in type-theoretical semantics, *Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736*.

Zhaohui LUO (2012), Common Nouns as Types, in *LACL'2012, LNCS 7351*.

Zhaohui LUO, Sergei SOLOVIEV, and Tao XUE (2012), Coercive subtyping: theory and implementation, *Information and Computation*, 223:18–42.

Andrea MARCHETTI, Maurizio TESCONI, Francesco RONZANO, Marco ROSELLA, and Salvatore MINUTOLI (2007), SemKey: A Semantic Collaborative Tagging System, in *Tagging and Metadata for Social Information Organization Workshop, WWW07*.

Per MARTIN-LÖF (1975), An Intuitionistic Theory of Types: predicative part, in H.ROSE and J.C.SHEPHERDSON, editors, *Logic Colloquium'73*.

Per MARTIN-LÖF (1984), *Intuitionistic Type Theory*, Bibliopolis.

Igor MEL'CUK and Andrei ZHOLKOVSKY (1988), The Explanatory Combinatorial Dictionary, in Martha Walton EVENS, editor, *Relational Models of the Lexicon: Representing Knowledge in Semantic Networks*, pp. 41–74, Cambridge University Press, Cambridge.

Rada MIHALCEA and Timothy CHKLOVSKI (2003), Building sense tagged corpora with volunteer contributions over the Web, in *RANLP*, volume 260 of *Current Issues in Linguistic Theory (CILT)*, pp. 357–366, John Benjamins, Amsterdam/Philadelphia.

George A. MILLER (1995), WordNet: A Lexical Database for English, *Commun. ACM*, 38(11):39–41.

Koji MINESHIMA, Yusuke MIYAO, and Daisuke BEKKI (), Higher-order logical inference with compositional semantics, in *Proceedings of EMNLP15*, pp. 2055–2061.

Richard MONTAGUE (1974), *Formal Philosophy*, Yale University Press, collected papers edited by R. Thomason.

Alain POLGUÈRE (2003), Collocations et fonctions lexicales : pour un modèle d'apprentissage., *Revue Française de Linguistique Appliquée*, E(1):117—133.

Alain POLGUÈRE (2014), From Writing Dictionaries to Weaving Lexical Networks, *International Journal of Lexicography*, 27(4):396—418.

James PUSTEJOVSKY (1995), *The Generative Lexicon*, MIT.

Aarne RANTA (1994), *Type-Theoretical Grammar*, Oxford University Press.

Christian RETORÉ (2014), The Montagovian Generative Lexicon Lambda Ty: a Type Theoretical Framework for Natural Language Semantics, in Ralph MATTHES and Aleksy SCHUBERT, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 202–229, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, ISBN 978-3-939897-72-9, ISSN 1868-8969, doi:http://dx.doi.org/10.4230/LIPIcs.TYPES.2013.202, `http://drops.dagstuhl.de/opus/volltexte/2014/4633`.

Franck SAJOUS, Emmanuel NAVARRO, Bruno GAUME, Laurent PRÉVOT, and Yannick CHUDY (2013), Semi-automatic enrichment of crowdsourced synonymy networks: the WISIGOTH system applied to Wiktionary, *Language Resources and Evaluation*, 47(1):63–96.

Katharina SIORPAES and Martin HEPP (2008), Games with a Purpose for the Semantic Web, 23:50–60, ISSN 1541-1672, doi:10.1109/MIS.2008.45, `http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4525142`.

John SOWA and John ZACHMAN (1992), Extending and Formalizing the Framework for Information Systems Architecture, *IBM Systems Journal*, 31(3):590–616.

Luis VON AHN and Laura DABBISH (2004), Labeling Images with a Computer Game, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pp. 319–326, ACM, New York, NY, USA.

Luis VON AHN and Laura DABBISH (2008), Designing games with a purpose, *Commun. ACM*, 51(8):58–67.

Luis VON AHN, Mihir KEDIA, and Manuel BLUM (2006), Verbosity: a game for collecting common-sense facts, in *CHI*, pp. 75–78, ACM.

Manel ZARROUK (2015), *Endogeneous Consolidation of Lexical Semantic Networks*, Theses, Université de Montpellier.

Naomi ZEICHNER, Jonathan BERANT, and Ido DAGAN (2012), Crowdsourcing inference-rule evaluation, in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 156–160, Association for Computational Linguistics.

Michael ZOCK and Slaven BILAC (2004), Word lookup on the basis of associations: from an idea to a roadmap., in *Proceedings of the Workshop on Enhancing and Using Electronic Dictionaries, Association for Computational Linguistics.*, pp. 29–35.

Michael ZOCK and Didier SCHWAB (2008), Lexical Access Based on Underspecified Input, in *Proceedings of the Workshop on Cognitive Aspects of the Lexicon*, COGALEX '08, pp. 9–17, Association for Computational Linguistics, Stroudsburg, PA, USA.

Interfacing language, spatial perception and cognition in Type Theory with Records

Simon Dobnik and Robin Cooper
Department of Philosophy, Linguistics & Theory of Science (FLOV)
Centre for Linguistic Theory and Studies in Probability (CLASP)
University of Gothenburg, Sweden

ABSTRACT

We argue that computational modelling of perception, action, language, and cognition introduces several requirements of a formal semantic theory and its practical implementations in situated dialogue agents. Using examples of semantic representations of spatial descriptions we show how Type Theory with Records (TTR) satisfies these requirements and provides a promising knowledge representation system for situated agents.

Keywords: spatial language, Type Theory with Records (TTR), computational framework

1 INTRODUCTION

In this paper, we consider the treatment of spatial language from the perspective of a robot learning spatial concepts and classifying situations according to the spatial relations holding between objects while interacting with a human conversational partner. We start from our experience of building such agents and a conclusion that there is a need for a unified knowledge representation system that connects theories of meaning from formal semantics to practical implementations. We suggest that the type-theoretic notion of *judgement* is important and that a type theory such as TTR (Type Theory with Records) is advantageous because it can be used to model both the low level perceptual judgements of the robot as well as the conceptual judgements associated with spatial relations. This is distinct from previous approaches

(discussed in Section 3) where the low level perceptual processing is carried out in an entirely different system to that used for semantic processing. An advantage we claim for our approach is that it facilitates the construction of types which have components relating to both low level and high level processing.

In Section 2, we give an overview of the problem area before describing some of the approaches that have been taken in Section 3. We then give a brief intuitive account of the tools we are using from TTR in Section 4 and give some examples of how this relates to understanding spatial descriptions (as our focus is on knowledge representation) in Section 5. Finally, in Section 6, we offer some conclusions and perspectives for future work. An implementation of examples in this paper is available on `https://github.com/GU-CLASP/pyttr/blob/master/lspc.ipynb`.

2 COMPUTATIONAL MODELLING OF SPATIAL LANGUAGE

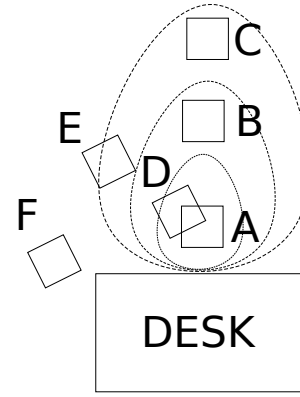
We approach the study of spatial descriptions from the perspective of building computational models for situated agents which we have implemented so far, the typical problems and the ad-hoc solutions taken when representing multi-sourced information. Spatial language is central for situated agents as these must resolve their meaning and reference to visual scenes when being involved in conversations with humans. In such conversations humans would use locational information to identify objects (*the chair to the left of the table*), describe directed action (*pick up the red cube near the green one*) or give route instructions (*go down this corridor nearly towards its end and then take the second door to your right*). However, interfacing language and perception is not only the domain of applications that involve language-based interaction with humans. There is an emerging trend in robotics where information represented in language is used as assistance to *visual search* (Sjöö 2011; Kunze *et al.* 2014). Robots are typically equipped with several sensors that allow creation of perceptual representations at different levels of abstraction. Creating and classifying for all representations all the time is therefore a computationally expensive task. In the domain of visual object recognition, a system would have to employ all image classifiers on every observation it makes even if most

of these classifiers would not yield a match in these situations. For example, the robot is in a corridor and is applying classifiers that would recognise objects found in a kitchen. Having background knowledge about the likely distribution of objects would allow it to prioritise certain classifications. The ontology capturing this knowledge may be static or dynamically built through interaction (Dobnik and Kelleher 2016). In the latter case humans programme the robot through language (Lauria *et al.* 2002).

Cross-disciplinary research has shown that spatial language is dependent on several contextual factors that are part of an agent's interaction with the environment through perception and other agents through dialogue, for example geometrical arrangement of the scene (Regier and Carlson 2001), the type of objects referred to and their interaction (Coventry *et al.* 2001; Dobnik and Kelleher 2013, 2014), visual and discourse salience of objects (Kelleher *et al.* 2005), alignment in dialogue (Watson *et al.* 2004; Dobnik *et al.* 2015), and gesture (Tutton 2013) among others.

The geometrical arrangement of scenes is captured in *spatial templates* or *potential fields*. These can be captured experimentally by placing the target object in various locations around the landmark object and asking participants for judgements whether a particular spatial relation holds (Logan and Sadler 1996; Dobnik and Åstbom 2017). The semantics of spatial templates may be approximated to functions (Gapp 1994a,b) or expressed as a general function with trainable parameters as in the case of the *Attentional Vector Sum (AVS)* model (Regier and Carlson 2001). Figure 1 shows a spatial template for the description *in front of* relating a table and a chair. Spatial templates capture gradience of semantics of spatial descriptions in terms of angles and distances from the location and the orientation of the landmark object. There are regions where native speakers would judge the relation holds to a high degree, for example for the placement of chairs A and D, and regions where the relation holds to a lesser degree, the placement of chairs C and E, or does not hold at all, the placement of chair F. A particular scene may be matched by several spatial descriptions. Spatial templates are far from being fixed or universally applicable. In addition to angle and distance, several contextual parameters can be incorporated, for example the presence of distractor objects (Costello and Kelleher 2006), object occlusion (Kelleher *et al.* 2011),

Figure 1:
The chair is in front of the desk



or the function itself can be learned from a dataset of perceptual observations and descriptions as a classifier (Roy 2002; Dobnik 2009).

Scene geometry is not the only meaning component of spatial descriptions. Spatial relations are also expressing other non-geometric aspects of how we view the relation between the landmark and the target objects. For example, a description such as *Alex is at her desk* might not only mean that Alex is proximal to her desk. Instead, we might interpret the description that she is sitting in her chair facing a computer screen and working. In literature, such aspects of meaning are known as *functional aspects* (Coventry and Garrod 2005) because they are dependent on the function of interacting objects: what are they used for, how do they interact with each other, and how they can be manipulated? In order to understand the interaction of objects, one needs to observe what will happen to scenes. Coventry *et al.* (2005) model functional aspects of meaning as *dynamic-kinematic routines* captured by several stacked recurrent neural networks that take both visual and language input data. Modelling different takes on the scene and the relations into which the target and the landmark objects enter leads to the development of qualitative spatial ontologies (Bateman *et al.* 2010) and logics such as (Zwarts and Winter 2000; Cohn and Renz 2008) which are similar to Allen's interval algebra for temporal reasoning (Allen 1983).

Spatial descriptions are also sensitive to changing linguistic context that arises in linguistic interaction. One such example is the coordination of referring expressions (Garrod and Doherty 1994). Projective spatial descriptions such as *to the left of* and *behind* require setting

Katie: Please tell me, where is the darker box?

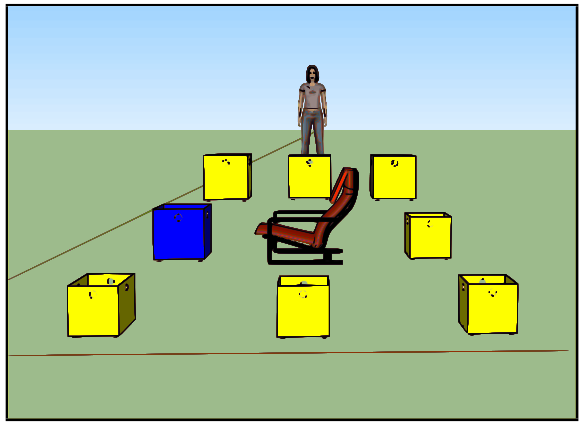


Figure 2:
Assignment of FoR in dialogue

a perspective or the frame of reference (FoR) which can be modelled as a set of three orthogonal axes fixed at some point (the location of the landmark object) and oriented in a direction determined by the viewpoint (Maillat 2003). The viewpoint can be any conversational participant or object in the scene (for this reason such FoR assignment is known as *relative FoR*[1]) that has an identifiable front and back which introduces considerable referential ambiguity of projective spatial descriptions. Alternatively, a scene can also be described from a global bird's eye perspective, e.g. *North of*, in which case we talk about *extrinsic FoR* assignment. The FoR may be specified overtly such as *from your point of view* but frequently it is omitted and its resolution is relied upon the dynamics of conversation, among other things.

Figure 2 shows a virtual scene involving a conversational partner, Katie, facing us at the opposite side of the room. What FoR would we use to continue the conversation? How would the FoR be assigned over several utterances and conversational role changes? Would conversational partners align with a particular FoR or would they tend to change it frequently – and what are the conditions licensing such change? What other factors in addition to linguistic conversation contribute to the assignment of FoR? Can a system learn from human assignments of FoR and successfully demonstrate its knowledge in a new

[1] We do not distinguish *intrinsic* FoR as this is *relative* FoR where the viewpoint is the landmark object.

conversation with a human? We investigate the strategies of FoR assignment in dialogue, both restricted and free, in (Dobnik *et al.* 2014) and (Dobnik *et al.* 2015) respectively.

The preceding discussion demonstrates that the semantics of spatial descriptions involves meaning representations at three distinct levels none of which have been so far captured in a single representational framework which could be employed with situated conversational agents. (i) Geometric representations involve grounding symbols in perceptual observations (Harnad 1990), (ii) integrating of functional knowledge involves lexical and compositional semantics, and (iii) FoR assignment involves both of the previous steps and pragmatics of conversation. Modelling the semantics of spatial descriptions thus raises several open questions. How is an agent able to determine the *sense* and *reference* (Frege 1948)[2] of spatial descriptions? The former relates to what components of lexical meaning are involved and the latter relates to how expressions relate to contextual features arising from perceptual and discourse contexts. A model of *grounding* is required: how are perceptual and conceptual domains bridged (reference) and how is information from contextual features fused into bundles of meaning representations (sense)? The resulting framework should possess sufficient *formal accuracy* and *expressiveness* of representations for modelling human language and reasoning to capture notions such as logical entailment, scoping properties, underspecification, hierarchical organisation of meaning and structure, compositionality of structure for words, sentences and utterances, recursion, feature unification, and others. The framework should also include a *learning theory* concerning how an agent is able to adapt or learn its representations in new physical and conversational contexts (Cooper *et al.* 2015; Dobnik and Kelleher 2016).

3 COMPUTATIONAL FRAMEWORKS

3.1 *A classical view of vision and language*

Figure 3 shows a typical approach to modelling language and vision. We start by building a model of the perceptual scene which captures its geometrical representation. In this example, the robot starts with a

[2] The paper was first published in 1892.

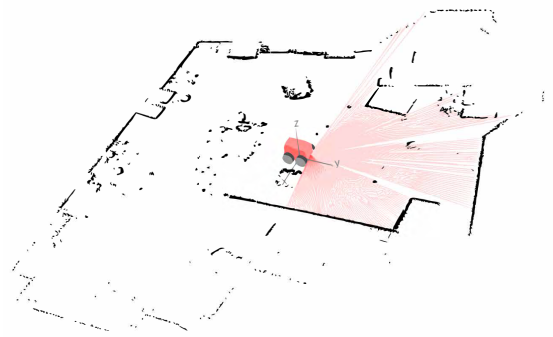
1. 

Figure 3: Grounding language in perception

2. $\forall x \forall y [\text{supports}(y,x) \wedge \text{contiguous}(\text{surface}(x), \text{surface}(y)) \rightarrow \text{on}_1(x,y)]$

3. *The newspaper is on the table*

SLAM map (Dissanayake *et al.* 2001) which contains clouds of points in 3-dimensional coordinate space. The perceptual model is then connected to a formal conceptual representation of the scene which in this example is expressed in first-order logic. An important and challenging issue here is to find a mapping between a reasonably accurate geometric representation of a scene with continuous parameters (locations in the coordinate space and angles of orientation) to cognitive categories that are reflected in language. The mapping between two such domains thus results in vagueness. The formal representation is then mapped to the linguistic expression. The mapping between the layers is typically learned from datasets of collected observations with machine learning. For example, in (Dobnik 2009) we learn classifiers that map representations from SLAM maps to words, thus skipping an intermediate representational layer. Matuszek *et al.* (2012a) present a method where also the intermediate semantic representation is included: linguistic expressions are grounded in compositional semantic forms which are grounded in perception. Finally, natural language does not only need to be grounded in perception but also in the robotic control language (Matuszek *et al.* 2012b).

3.2 *Model-theoretic Montague semantics*

Classical model-theoretic or Montague semantics uses higher order logic (Montague 1974; Dowty *et al.* 1981; Blackburn and Bos 2005; Bird *et al.* 2009) which provides the required and desired formal accuracy and expressiveness of a representation system. It accounts for how meaning representations of words are composed in the form of higher

order functions to form meaning representations of sentences. The functional composition of constituents allows us to translate between sentence constituent structure and its logical representation as shown in Figure 4. The final logical forms of spatial prepositions are slightly more complicated than presented in this example and due to their context dependency a single description or utterance (surface form) may resolve to several representations as discussed in (Miller and Johnson-Laird 1976; Herskovits 1986), for example $\text{on}(x,y)_1$: $\text{object}(x) \wedge \text{object}(y) \wedge \text{supports}(y,x) \wedge \text{contiguous}(\text{surface}(x),\text{surface}(y))$ and $\text{on}(x,y)_2$: $\text{object}(x) \wedge \text{object}(y) \wedge \text{contiguous}(\text{boundary}(x),y)$. However, dealing with these two issues separately, we are able to derive their compositional representation along the same lines as in Figure 4.

In model-theoretic semantics the expression's reference is determined by an assignment, a valuation function between linguistic strings and entities (or sets of tuples of entities) in a model. The model is agent external and fixed. The valuation returns true if an entity or a relation between entities denoted by an expression can be found in the model, otherwise it returns false. While it would be possible to represent the referential semantics of *on* in a model by listing a set of all coordinates of locations where this spatial description applies, this referential representation of meaning is cumbersome as the model would have to include an assignment for every scale, for every spatial relation, for every pair of objects. Since angles and distances in a coordinate system are continuous measures this means that such sets would be infinite. The model also does not straightforwardly represent gradience and vagueness of spatial descriptions. In order to do that one would have to resort to the notion of possible worlds (Lassiter 2011) which introduces further computational complexity (for discussion see (Cooper *et al.* 2015, Section 1.1, p.3ff)).

As discussed earlier both vagueness and gradience of spatial language are captured in computational models as spatial templates or potential fields. While spatial templates can be thought of as referential overlays of regions induced experimentally (as a set of points where participants consider a particular spatial relation to apply), potential fields capture the notion that such regions can be generalised as functions. However, as argued in (Lappin 2013) these functions do not represent objects in a model (or extensions or the referential meaning of these descriptions) but rather they capture their *sense* or

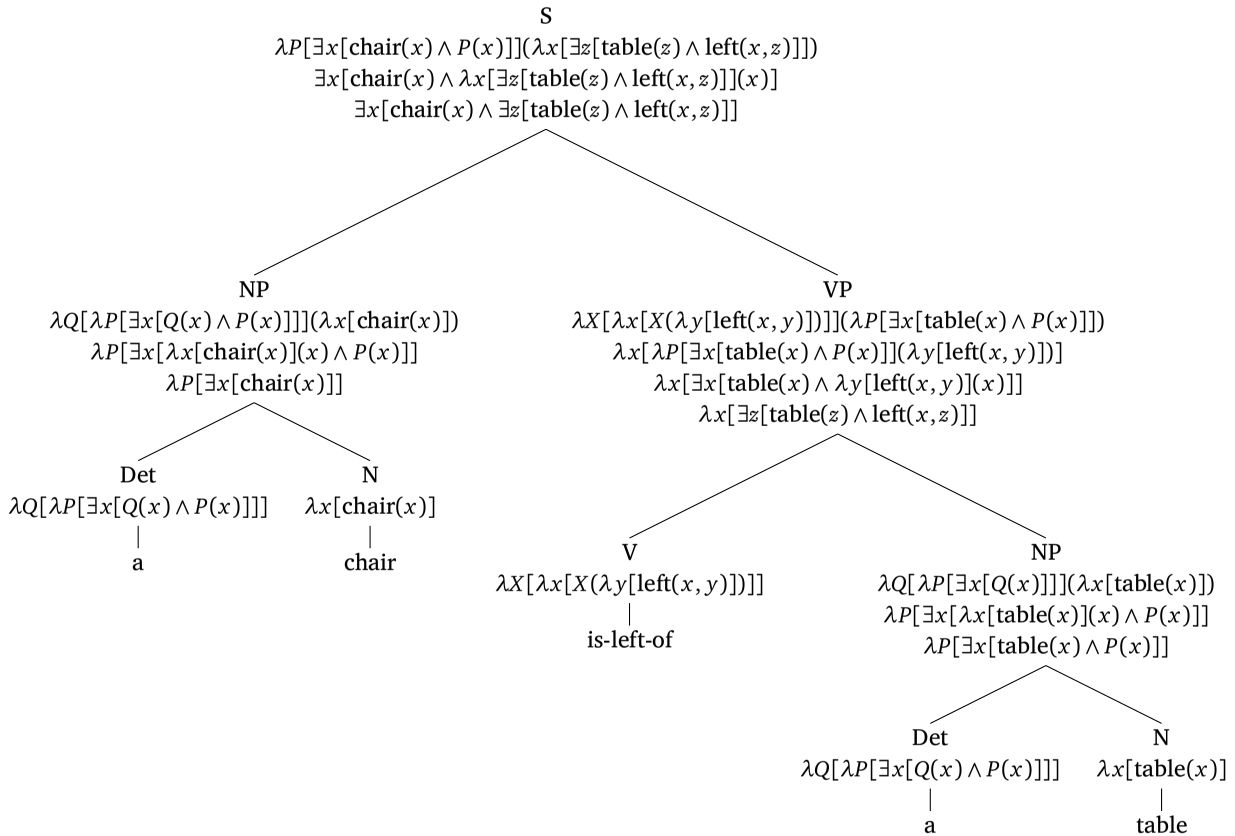


Figure 4: A chair is left of a table

intension specifying in what ways a description relates to perceptual observations. Knowing this function, we can check whether a particular spatial relation associated with the function applies to a particular pair of objects and to what degree. The notion of applying a function from perceptual observations to words (or the other way around) representing the meaning of words is also known as *grounding* these words in perception (Harnad 1990).

The model-theoretic approach to semantics assumes that a model is derived through some external process and therefore pre-given, that it is complete and represents a state of affairs at a particular temporal snapshot (Fagin *et al.* 1995). In practice, however, complete models may be rarely observable and we must deal with partial models. We must also account for the fact that we may incrementally observe more and more of the world and we have to update the model with new observations, sometimes even correct representations that we have already built in the light of new evidence. Finally, the world is not static itself as new objects and events continually come into existence. Imagine a robot (and indeed such robots were used in the early days of robotics) with a pre-programmed static model of the world. Every minute change in the world would render it useless as there would be a discrepancy between its representation of the world and the actual world. Modern robotic models used in localisation and map building are incrementally learned or updated over time by taking into account robot's perceptional observations and motion and errors associated with both (Dissanayake *et al.* 2001). An important consequence of this is that the model of the world a robot builds is individual to a particular robot's life-span and experience. Two robots experiencing the same world will have slightly different models. Of course, the more they experience the world, the more similar their models will be. It is conceivable that humans learn meanings in the same way. However, doing so they are equipped with yet another tool to overcome individual inconsistencies in their model. They can use linguistic dialogue interaction to resolve such inconsistencies in the form of repair (Pickering and Garrod 2004). In robotics, several models that explore learning language through interaction have been built which include (Steels and Belpaeme 2005; Skočaj *et al.* 2011; Ivaldi *et al.* 2014), also related to spatial cognition (Steels and Loetzsch 2009). We describe a system for the mod-

elling of semantic concept learning through dialogue interaction in (Dobnik and de Graaf 2017).

3.3 *Models used in robotics*

In building situated conversational agents, several systems have been proposed but none of them capture all of the requirements discussed in Section 2. For example, *semiotic schemas* (Roy 2005) represent the lexical meaning of words as directed graphs composed of nodes that, in turn, represent sequences of perceptual observations and classification events as shown in Figure 5. The meaning/sense of an object is defined in terms of what can be experienced with the sensors and actuators of a robot. The reference is determined by embedding a semiotic schema with the actual sensory readings. For example, a cup can be experienced and classified either through visual or haptic modalities. The location of the sensory readings determines the location of the object. Semiotic schemas represent a very attractive model of grounded lexical semantics of words, but how such semiotic schemas compose to form larger linguistic structures is left unaccounted for.

Quite frequently, grounded representations are arranged into layers. This is related to the fact that in practical applications several distinctive sub-systems are used that are stacked into a pipeline. For example, in the layered approach of (Kruijff *et al.* 2007), here summarised in Figure 6, the lowest level consists of a feature map which directly relates to laser sensors. Here, features are sets of points which can be connected to lines which represent walls. The next level is a navigation graph. As the robot moves around space, it creates nodes.

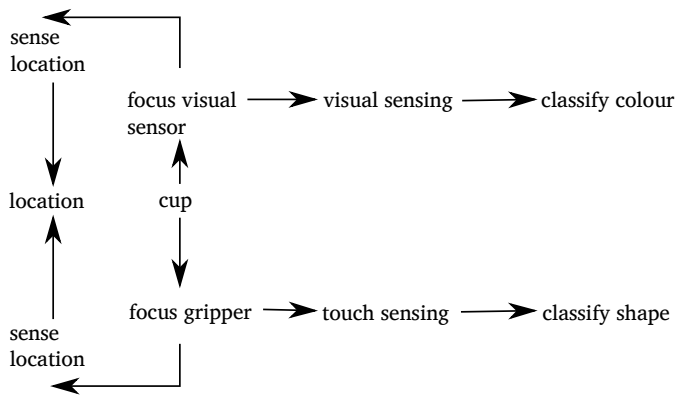


Figure 5:
A simplified
semiotic schema

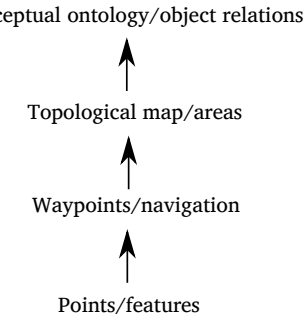


Figure 6:
A layered approach

If the robot can move directly between two nodes, a connection is made and, on the basis of several such connections, a navigation graph is created. Groups of nodes may be identified whereby two groups are only connected through a single node in each group. Such nodes are gateway nodes and indicate passages between different areas or doors. From such a topology of nodes, a topological map can be hypothesised such that it identifies enclosed spaces, corridors, kitchens, and rooms. The information about the spaces can be further augmented with linguistic information from the ontology, for example what objects are found in kitchens. In this approach one needs to design interfaces between representational levels in the pipeline. Most frequently, representations and operations at each level are distinct from each other. A question we would like to explore is whether representations at different levels can be generalised by taking inspiration from the way humans assign, learn, and reason with meaning. A unified meaning representation would allow interactions between modalities that are required in modelling human cognition but are difficult to implement in a layered pipeline architecture.

4 TYPE THEORY WITH RECORDS (TTR)

Type Theory with Records (TTR) (Cooper 2012) builds on the tradition of classical formal semantics (and therefore captures the notion of compositionality) but at the same time, drawing on insights from situation semantics, addresses the outstanding questions related to perception discussed in the preceding paragraphs. It starts from the idea that information is founded on our ability to perceive and classify the world, that is to perceive or *judge* objects and situations as being of

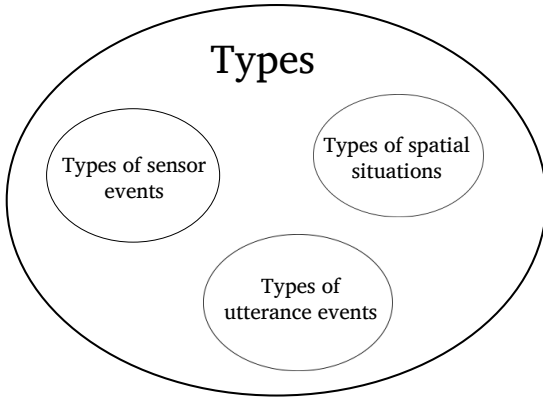


Figure 7:
A unified view: types all over the place

types. All information can be represented as types (Figure 7) which makes type assignment an abstract theory of cognition and perception. Having a single representational layer allows information fusion between perception, conceptual knowledge and linguistic communication which is an important requirement for modelling spatial descriptions.

Types are intensional – that is, there can be distinct types which have identical extensions. For example, the type of situations in which an object, a, is to the left of another object, b, in symbols *left(a,b)*, can have exactly the same witnesses as the type of situations in which b is to the right of a, *right(b,a)*, without requiring that the two types be identical. For some more discussion of the intensional nature of types in TTR see (Cooper 2017). This allows us to relate linguistic propositions to types, the so-called *propositions as types* dictum which is standard in type theories deriving from the original work of Martin-Löf (Martin-Löf 1984; Nordström *et al.* 1990). The notion of truth is linked to judgements that an object a is of type T ($a : T$). As in standard Martin-Löf type theories, a type is *true* just in case it has some witness. Thus, the type of situations *left(a,b)* is *true* just in case there is some situation where a is to the left of b.

We can furthermore seek to operationalise the types as computable functions (Lappin 2013) or classifiers (Larsson 2015), rather than associating them with sets of witnesses as in the standard definition of TTR (Cooper in prep, 2012). Under this view, we can consider an agent to have access to a particular type inventory as a resource. Different agents can have access to different type resources which can

be dynamically revised, both in terms of learning new types and in modifying the witness conditions in terms of classifiers, which can change as the result of the agent's experience of new situations (Dobnik *et al.* 2013; Larsson 2015). In order for communication between agents to be possible, they must converge on sufficiently similar type resources. This convergence is in part enabled by the fact that the agents exist in similar environments and have similar perceptual apparatus to classify features in the environment. But in addition it is important that the agents be able to use language to communicate with each other about their classification of features in the environment. For example, an agent may receive linguistic information which provides a classification which is at variance from that given by its perceptual apparatus or, in linguistic communication between agents, corrective feedback might be used to express a variance in judgement by two agents.

This is perhaps a novel view in linguistic semantics and computational linguistics but it relates to a standard view in mobile robotics (Dissanayake *et al.* 2001) where a map of an environment is constructed dynamically as a robot moves around in it and features are constructed on the basis of clouds of points in 3D space where the robot's sensors indicate that something is present. In our terms, this would correspond to recognising the physical presence of an object and assigning a particular type to it.

In such a learning scenario, it is natural to consider the role of probabilistic judgements, that is, the judgement that an object a is of type T with probability p instead of the standard categorical judgements to be found in type theory. For a proposal of how this might be incorporated into TTR see (Cooper *et al.* 2015). This means that an agent can determine a degree of belief that a particular situation is of a particular type. For example, the probability that a situation is to be classified as one where an umbrella is *over* a person may vary with respect to both geometric configuration and the degree to which the umbrella is protecting the person from rain (Coventry *et al.* 2001).

In contrast to the classical Montagovian semantic framework which employs a variant of the simple theory of types, TTR introduces an extended set of basic types (for example *Ind* and *Real* that correspond to the basic conceptual categories *individuals* and *real numbers*. However, it is also *a rich type system* which, in addition to basic types,

contains complex types constructed from types and other objects, among them *ptypes* constructed from predicates and their arguments, such as *left(a,b)*, and *record types*, such as,

$$\begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{y} & : & \textit{Ind} \\ \text{e} & : & \text{left(x,y)} \end{bmatrix}$$

whose witnesses would be any record with three fields labelled by x, y and e, respectively (and possibly more fields with other labels) such that the x-field contains an object a of type *Ind*, the y-field contains an object b of type *Ind* and the e-field contains an object of type *left(a,b)*. For a detailed characterisation of record types in TTR see (Cooper in prep, 2012). Record types in TTR are used to model, among other things, lexical content and dialogue information states. For our present purposes, the structured nature of record types allows us to combine in a single object the kind of multi-source information needed for robotics and the modelling of spatial descriptions representing a bridge between what might be thought of in other approaches as the sub-symbolic domain of perception and the symbolic domain of high level conceptual analysis.

The structured nature of record types in TTR allows representation of several kinds of formal structural relations which has implications for inference of representations containing multi-sourced information. Record types (and the corresponding records) can be compared with each other. Consider the following example. If

$$\textit{Relation} = \begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{y} & : & \textit{Ind} \\ \text{c}_1 & : & \text{target(x)} \\ \text{c}_2 & : & \text{landmark(y)} \end{bmatrix}$$

and

$$\textit{Left} = \begin{bmatrix} \text{x} & : & \textit{Ind} \\ \text{y} & : & \textit{Ind} \\ \text{c}_1 & : & \text{target(x)} \\ \text{c}_2 & : & \text{landmark(y)} \\ \text{c}_3 & : & \text{left(x,y)} \end{bmatrix}$$

then *Left* $\sqsubseteq$ *Relation* where $\sqsubseteq$ denotes the *subtype* relation (Cooper 2012, p.295). Similarly, record types allow identification of depen-

dencies using *dependent types*. The notation like target(x) within the context of the record type above is an abbreviation for a tuple of objects $\langle \lambda v{:}Ind \ .\ \text{target}(v), \langle \text{x} \rangle \rangle$ where the first element is a dependent type, a function mapping objects to a type, and the second element is a sequence of paths to the arguments of this function within a record type. Finally, both *Ind* and target(x) are *component types* of record types *Relation* and *Left* which means that the latter types are representations of thematic relations between individuals and properties found in language (Lin and Murphy 2001; Estes *et al.* 2011).

5 TYPES OF SPATIAL DESCRIPTIONS

In the remainder of the paper, we discuss how our empirical investigations of learning geometric meanings of spatial descriptions with situated robots (Dobnik 2009; Dobnik and de Graaf 2017), learning functional meanings of prepositions from collections of image descriptions (Dobnik and Kelleher 2013, 2014), and modelling of reference frame assignment in conversation (Dobnik *et al.* 2014, 2015) can be captured in the TTR framework.

The idea is that TTR can be seen as an abstract model of cognition and perception (Cooper 2012, in prep) which can be used to model both the linguistic behaviour of humans as well as perception based on sensor readings in artificial agents. It is important to note that robots have different perceptual apparatus than humans, both in the number and the nature of sensors. It follows that their sensors will give rise to different types of information at the lowest sensory level. However, these sensory types can be related to types corresponding to concepts which are similar enough to conceptual types internalised by humans to allow communication between the two. Nevertheless, the type system an agent can acquire is constrained by the agent's perceptual apparatus. We cannot, for example, expect an agent incapable of colour perception to successfully make judgements about the colour concepts available to a human, however much we may talk to the agent or train it on objects of different colours. It simply does not have the required sensors and classifiers to distinguish the appropriate situations.

There are two main aspects of theoretical interest with the approach we suggest:

1. The notion of *judgement* from type theory can be used to model both the kind of low level perceptual discrimination carried out by classifiers in robotic systems and the high level conceptual classification including the truth of propositions which are important for linguistic semantics. Thus, it offers the possibility of a unified approach to both.
2. Given the kind of structured types that are proposed in a system like TTR it is not only possible to express relations between the low level and high level types but even to have a low level perceptual type and a high level conceptual type as components within a single type and even to have one type depend on the other. This gives a very different perspective on the cognitive makeup of situated agents than that given by the kind of layered approaches discussed in Section 3, where the different layers involve entirely different systems.

In the next section we will give examples which illustrate this.

5.1 *Types of objects*

Figure 8 shows an example of bridging between perceptual and conceptual domains for object recognition. Step 1 shows a record of type *PointMap* which is produced by SLAM (for details see (Dobnik *et al.* 2013)). The type *PointMap* is a subtype of a type that represents a list of records containing three real numbers modelling points in three-dimensional space. A point map is a list (or a set) of points that a robot is tracking in space. TTR allows function types one of which is exemplified in the object detection function in Step 2. This function maps an object of type *Pointmap* to a type that represents a set of records specifying (1) the reg(ion) occupied by the object (a sub-pointmap) and (2) a property which is modelled as a *pfun* which maps an individual to a type, in this case a ptype or a predicate type. The purpose of this function type is to associate a perceptual object and some property, thus to pair two kinds of information. The property functions take objects of type *Ind*ividuals to types of individuals having some property. The target record type of the main function type does not yet constrain any individuals that this property could be assigned to nor does this record type correspond to a situation. In Step 3 we introduce an individuation function which takes records of associated perceptual

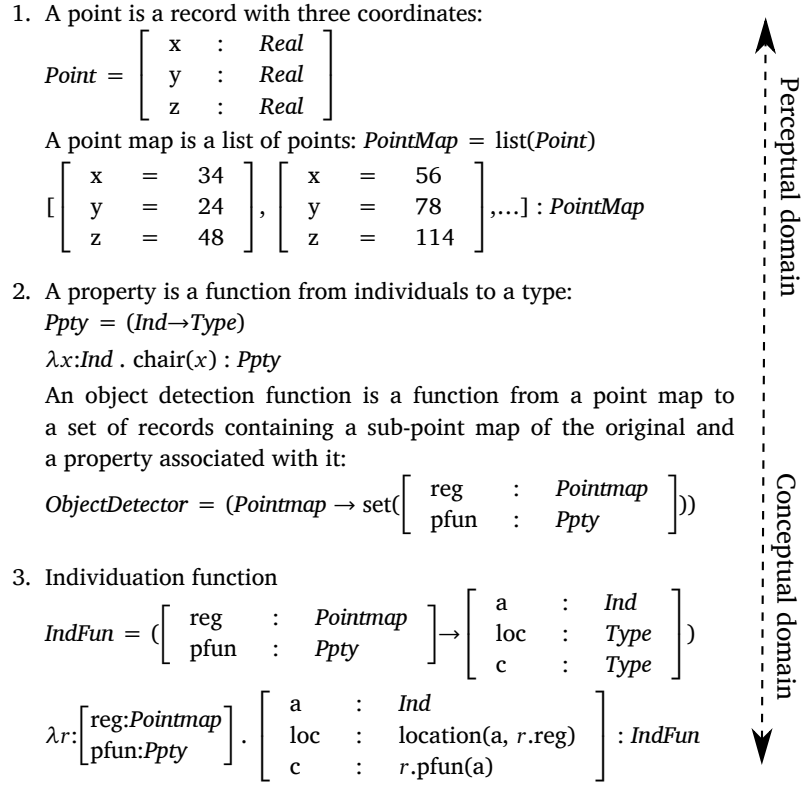


Figure 8: From perceptional to conceptional domain

objects and properties and yields a type of situation involving an individual located at a certain location and having this property. This type therefore represents a cognitive take on a situation.

In this example, the mappings between the types are modelled with functions but in practice (some) associations would be learned. For example, Harnad (1990) argues that grounding, associating perceptual and conceptual domains, can only be accomplished through classification. In (Dobnik 2009), decision tree and Naïve Bayes classifiers are learned to classify between point clouds and spatial descriptions. Here, the associating function that the classifier has learned is in the domain of the hypothesis space of each learning algorithm and is therefore quite complex. Larsson (2015) introduces a perceptron model to TTR and Cooper *et al.* (2015) give the type system, including function types, a Bayesian interpretation. The latter allows direct propagation of Bayesian probabilistic beliefs between the types while

the observed type probabilities can be trained based on the agent's observations.

5.2 *Types of spatial situations*

Spatial descriptions, e.g. *over* and *above* are sensitive to classes of interacting objects and the contribution of such functional world-knowledge versus geometric knowledge for the semantics is different from one spatial preposition to another (Coventry *et al.* 2001; Coventry and Garrod 2005; Coventry *et al.* 2005). While previous work attempted to determine the contribution of each modality experimentally, Dobnik and Kelleher (2013, 2014) extract functional information from a large corpus of text describing images. Image descriptions are constrained by the properties of the visual scene shown in the image, both perceptual (geometric arrangement of the scene) and functional (the nature and interaction of objects shown there). Both kinds of information will be reflected in the text describing the image, in a particular choice of descriptions that annotators used. Building lexical models of word co-occurrence thus allows us to capture functional interactions between prepositions and targets and landmarks. In (Dobnik and Kelleher 2013) we capture the strength of association between a preposition and different target-landmark pairs with *log-likelihood ratio*. In (Dobnik and Kelleher 2014), we generalise the types of targets and landmarks of a particular spatial preposition by ascending in a WordNet hierarchy (Fellbaum 1998). This allows us to generate patterns of prepositional use such as the following: *person.n.01 under tree.n.01*, *shirt.n.01 under sweater.n.01*, and *person.n.01 under body of water.n.01*. Labels such as *person.n.01* indicate the labels given to the generalised synsets in the WordNet hierarchy. The patterns indicate types of spatial situations that the *under* relation applies to. Importantly, each of these patterns corresponds to quite a different arrangement of target and landmark objects and without such functional knowledge it would be difficult to capture a single spatial template that would not over-generate. The functional knowledge represented in these types thus constrains sub-sets of spatial situations for which individual spatial templates can be learned.

Figure 9(a) shows a TTR function that maps ontological knowledge from one ontological category to another. This is a similar function to *pfun* in the object detection function shown in Figure 8. It as-

Figure 9: Representing functional knowledge

(a) $\lambda r{:}\begin{bmatrix}\text{a:}\mathit{Ind}\\ \text{c:person(a)}\end{bmatrix}.\text{organism}(r.\text{a})$

(b) If $s:\begin{bmatrix}\text{a} & : & \mathit{Ind}\\ \text{loc} & : & \text{location(a},\pi)\\ \text{c} & : & \text{person(a)}\end{bmatrix}$
then $\exists s'\ [s' : \text{organism}(s.\text{a})]$

signs the individual of the type in the domain of the function a particular property $\lambda r.\text{organism}(r)$. Figure 9(b) shows how associative reasoning is captured in TTR. Having a meaning postulate in Figure 9(a) an agent can make a conclusion that a situation s of the first type (the left hand side of the *If-then* rule) requires that there is also a situation of the second type (the right hand side of the same rule).

Each type of situation representing a spatial pattern involves a different interplay of geometric and conceptual knowledge spanning the domain of point clouds and "logical" individuals. Figure 10 shows the conceptual constraints on the target and landmark objects limiting top-down a subset of spatial situations over which individual types of spatial relations are built. Hence, the resulting spatial template spatial-template$_{under_1}$ is a distinct pytpe classifier from spatial-template$_{under_2}$. In the generation step, the function in Figure 10 takes account of conceptual properties of objects that could be obtained by computing relevant hypernyms such as *person* and *furniture* and an associated spatial template that relates the point clouds associated with them. It then

Figure 10: Spatial templates sensitive to object function

$$\lambda r{:}\begin{bmatrix}\text{o}_1 & : & \begin{bmatrix}\text{a} & : & \mathit{Ind}\\ \text{reg} & : & \mathit{Pointmap}\\ \text{c} & : & \text{person(a)}\end{bmatrix}\\ \text{o}_2 & : & \begin{bmatrix}\text{a} & : & \mathit{Ind}\\ \text{reg} & : & \mathit{Pointmap}\\ \text{c} & : & \text{artefact(a)}\end{bmatrix}\\ \text{st} & : & \text{spatial-template}_{under_1}(o_1.\text{reg},o_2.\text{reg})\end{bmatrix}.\text{under}_1(\text{r}.o_1.\text{a},\text{r}.o_2.\text{a})$$

$$\lambda r{:}\begin{bmatrix}\text{o}_1 & : & \begin{bmatrix}\text{a} & : & \mathit{Ind}\\ \text{reg} & : & \mathit{Pointmap}\\ \text{c} & : & \text{person(a)}\end{bmatrix}\\ \text{o}_2 & : & \begin{bmatrix}\text{a} & : & \mathit{Ind}\\ \text{reg} & : & \mathit{Pointmap}\\ \text{c} & : & \text{body-of-water(a)}\end{bmatrix}\\ \text{st} & : & \text{spatial-template}_{under_2}(o_1.\text{reg},o_2.\text{reg})\end{bmatrix}.\text{under}_2(\text{r}.o_1.\text{a},\text{r}.o_2.\text{a})$$

generates a type of situation which involves a conceptual spatial relation between individuals.

5.3 *Types of dialogue information states*

TTR can also be used to model dialogue by representing types of information states (IS). Agents in conversation align with the primed frame of reference (FoR) and continue to use it (Dobnik *et al.* 2014). However, such alignment is only local and depends on the nature of the dialogue that agents are engaged in and other contextual factors of the conversation such as the perceptual properties of the scene or the task that agents are performing (Dobnik *et al.* 2015). Dobnik *et al.* (2014) study the properties of local FoR alignment over several turns of conversation in the constrained environment (Figure 2). The experiment captures participants' understanding of the agreed FoR and therefore alignment. In Game 1, a virtual conversational partner generates an unambiguous description that refers only to one of the objects. The participant must then click on that object. Here, the system primes the participant for a particular FoR. In Game 2, the system generates an ambiguous description which may refer to several objects. Again, the participant must click on one of the target objects but this time they must decide on a particular FoR assignment. Will this be aligned with the previous turn pair or will they assume a new strategy? Game 3 is identical to Game 2 and it tests if the priming from Game 1 is persistent over several games. In Game 4, the speaker-hearer roles reverse: the system selects an object and the participant must describe it using a particular FoR assignment. The role of this game is to test whether priming will persist if the conversational roles change.

The preceding interaction is formalised as a probabilistic model of FoR assignment over several local turns of conversation. This model is then applied in a generation experiment. Here, the system is making assumptions about the human conversational partner and is trying to align with them to the extent captured in the previous experiment. In Game1, the system chooses an object and a human primes the system by generating an unambiguous description. In Game 2, a human selects a box and the system generates a description using its FoR model. The human then confirms if the description is a good one. Game 3 is identical to Game 2. In Game 4 a human chooses a box. The system asks the user to describe it and also generates a description for itself.

A match between the human description and the system-generated description is compared. The results show a good agreement between humans and the system ($\geq$ 82.76% for Game 4).

The model of FoR assignment predicts, for example, that speakers initiating conversation tend to be egocentric. Figure 11 shows two types of information states (ISs). When Alex is planning the utterance *The chair is to the left of the table* her information state would be of the type shown in (a). Information states represent information that is private to the agent, and information that the agent believes is a part of the common ground with another conversational participant or shared. In the shared part of the IS in (a), there is a pointer to the object in focus. The object is stored in the private part of the IS as each agent builds its own objects. Σ_i is a type returned by an individuation function on the basis of the pointmap that the agent has constructed. The agent also has a private belief that they are one of the objects and a belief that two particular objects are in the left relation. Crucially, at this stage, the FoR origin is assigned to the object corresponding to the

(a) s_0^{Alex} :

$$\begin{bmatrix} \text{priv} : \begin{bmatrix} \text{objs:} \begin{bmatrix} o_0{:}\Sigma_0 \\ o_1{:}\Sigma_1 \\ o_2{:}\Sigma_2 \\ o_3{:}\Sigma_3 \end{bmatrix} \\ \text{bel:} \begin{bmatrix} c_{me} = [\text{c:me}(\Uparrow^2\text{objs.}o_0\text{.a})]{:}\textit{Type} \\ c_{left} = [\text{c:left}(\Uparrow^2\text{objs.}o_2\text{.a},\ \Uparrow^2\text{objs.}o_3\text{.a})]{:}\textit{Type} \end{bmatrix} \\ \text{for-origin} = \textbf{objs.o}_0\textbf{.a}{:}\textit{Ind} \\ \text{agenda} = [\begin{bmatrix} \text{move:}\textit{Assertion} \\ \text{cont} = \Uparrow^2\text{bel.}c_{left}{:}\textit{Type} \end{bmatrix}]{:}\text{list}(\textit{DMove}) \end{bmatrix} \\ \text{shared:}[c_{in\text{-}focus}{:}\Uparrow\text{priv.objs.}o_2\text{.a}] \end{bmatrix}$$

(b) s_1^{Sam} :

$$\begin{bmatrix} \text{priv} : \begin{bmatrix} \text{objs:} \begin{bmatrix} o_0{:}\Sigma_0 \\ o_1{:}\Sigma_1 \\ o_2{:}\Sigma_2 \\ o_3{:}\Sigma_3 \end{bmatrix} \\ \text{bel} : [c_{me} = [\text{c:me}(\Uparrow^2\text{objs.}o_1\text{.a})]{:}\textit{Type}\ ] \end{bmatrix} \\ \text{shared:} \begin{bmatrix} \text{speaker} = \Uparrow\text{priv.objs.}o_0\text{.a}{:}\textit{Ind} \\ c_{in\text{-}focus}{:}\Uparrow\text{priv.objs.}o_2 \\ \text{latest-move:} \begin{bmatrix} \text{speaker} = \Uparrow^2\text{priv.objs.}o_0\text{.a}{:}\textit{Ind} \\ \text{cont} = [\text{c:left}(\Uparrow^3\text{priv.objs.}o_2\text{.a},\ \Uparrow^3\text{priv.objs.}o_3\text{.a})]{:}\textit{Type} \end{bmatrix} \\ \text{for-origin} = \textbf{speaker}{:}\textit{Ind} \end{bmatrix} \end{bmatrix}$$

Figure 11: Types of dialogue information states

individual having this IS. A double arrow $\Uparrow^2$ indicates that the path refers to the container-type which the current type is a dependent type of, the superscript indicates the depth of embedding. Notation such as *label* = *value :* Type as in *for-origin* = $objs.o_0.a$ *: Ind* represents singleton types where the *value* stands for a manifest field.

The model of FoR assignment also predicts that hearers assume that speakers are egocentric. Figure 11(b) shows Sam's IS accommodating Alex's utterance. After Alex has made an utterance, the shared part of the IS is expanded through accommodation. There is information about the latest move: the speaker and the content of the move. Since Sam is a hearer of the utterance, he assumes that the FoR is identical to the speaker of the previous utterance as predicted by our probabilistic model. In this example, we assume that agents use identical labels for objects. However, it is not necessary or indeed possible that they have identified the same objects. In future work, we plan to investigate how agents resolve such differences using language, in particular what mechanisms of clarification and repair are used in such cases (Purver *et al.* 2003).

6 CONCLUSION

In this paper, we outlined an application of type theory to natural language semantics in the framework called Type Theory with Records or TTR which allows to relate semantics to action, perception, and cognition. We used TTR to represent different components of analysis of spatial descriptions. TTR is naturally suited for this task as it treats meaning being based on perception and interaction. Perception and conceptual reasoning can be related within one unified approach. The framework also points to similarities between linguistic and non-linguistic learning. We will be testing practical implementations of TTR with situated agents in our forthcoming work based on the framework described in (Dobnik and de Graaf 2017). The expressiveness of the type theoretic framework is associated with high computational cost. In order to make the framework computationally more tractable, we are investigating mechanisms of attention from psychological research which allow us to contextually restrict the type judgements a situated agent has to make (Dobnik and Kelleher 2016).

One aspect of spatial meaning which we have not discussed in this paper is the gradability of types like *left(a,b)*. For example, a would be judged to be left of b with a high probability if the two objects were close to each other. However, the probability of this judgement would decrease if a is much closer to the observer than b. This suggests exploring the use of probabilistic judgements in TTR as described in (Cooper et al. 2015) in our future work.

ACKNOWLEDGEMENTS

This paper was supported in part by the project Networks and Types (Vetenskapsrådet/Swedish Research Council project VR 2013-4873).

REFERENCES

James F ALLEN (1983), Maintaining knowledge about temporal intervals, *Communications of the ACM*, 26(11):832–843.

John A. BATEMAN, Joana HOIS, Robert ROSS, and Thora TENBRINK (2010), A linguistic ontology of space for natural language processing, *Artificial Intelligence*, 174(14):1027–1071.

Steven BIRD, Ewan KLEIN, and Edward LOPER (2009), *Natural language processing with Python*, O'Reilly, `http://nltk.org/book/`.

Patrick BLACKBURN and Johan BOS (2005), *Representation and inference for natural language. A first course in computational semantics*, CSLI Publications.

Anthony G. COHN and Jochen RENZ (2008), Qualitative Spatial Representation and Reasoning, in Vladimir Lifschitz FRANK VAN HARMELEN and Bruce PORTER, editors, *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, chapter 13, pp. 551–596, Elsevier.

Robin COOPER (2012), Type theory and semantics in flux, in Ruth KEMPSON, Nicholas ASHER, and Tim FERNANDO, editors, *Handbook of the Philosophy of Science*, volume 14 of *General editors: Dov M Gabbay, Paul Thagard and John Woods*, Elsevier BV.

Robin COOPER (2017), Adapting Type Theory with Records for Natural Language Semantics, in Stergios CHATZIKYRIAKIDIS and Zhaohui LUO, editors, *Modern Perspectives in Type-Theoretical Semantics*, number 98 in Studies in Linguistics and Philosophy, pp. 71–94, Springer.

Robin COOPER (in prep), Type theory and language: from perception to linguistic communication, `https://sites.google.com/site/typetheorywithrecords/drafts`, draft of book chapters.

Robin COOPER, Simon DOBNIK, Shalom LAPPIN, and Staffan LARSSON (2015), Probabilistic Type Theory and Natural Language Semantics, *Linguistic Issues in Language Technology — LiLT*, 10(4):1–43.

Fintan J. COSTELLO and John D. KELLEHER (2006), Spatial prepositions in context: the semantics of near in the presence of distractor objects, in *Proceedings of the Third ACL-SIGSEM Workshop on Prepositions*, Prepositions '06, pp. 1–8, Association for Computational Linguistics, Stroudsburg, PA, USA.

Kenny R. COVENTRY, Angelo CANGELOSI, Rohanna RAJAPAKSE, Alison BACON, Stephen NEWSTEAD, Dan JOYCE, and Lynn V. RICHARDS (2005), Spatial Prepositions and Vague Quantifiers: Implementing the Functional Geometric Framework, in Christian FREKSA, Markus KNAUFF, Bernd KRIEG-BRÜCKNER, Bernhard NEBEL, and Thomas BARKOWSKY, editors, *Spatial Cognition IV. Reasoning, Action, Interaction*, volume 3343 of *Lecture Notes in Computer Science*, pp. 98–110, Springer Berlin Heidelberg.

Kenny R. COVENTRY and Simon C. GARROD (2005), Spatial prepositions and the functional geometric framework. Towards a classification of extra-geometric influences, *Functional features in language and space: Insights from perception, categorisation and development*, pp. 163–173.

Kenny R. COVENTRY, Mercè PRAT-SALA, and Lynn RICHARDS (2001), The interplay between geometry and function in the apprehension of Over, Under, Above and Below, *Journal of Memory and Language*, 44(3):376–398.

M. W. M. G DISSANAYAKE, P. M. NEWMAN, H. F. DURRANT-WHYTE, S. CLARK, and M. CSORBA (2001), A solution to the simultaneous localization and map building (SLAM) problem, *IEEE Transactions on Robotic and Automation*, 17(3):229–241.

Simon DOBNIK (2009), *Teaching mobile robots to use spatial words*, Ph.D. thesis, University of Oxford: Faculty of Linguistics, Philology and Phonetics and The Queen's College, Oxford, United Kingdom, `http://www.dobnik.net/simon/documents/thesis.pdf`.

Simon DOBNIK and Amelie ÅSTBOM (2017), (Perceptual) grounding as interaction, in Volha PETUKHOVA and Ye TIAN, editors, *Proceedings of Saardial – Semdial 2017: The 21st Workshop on the Semantics and Pragmatics of Dialogue*, pp. 1–9, Saarbrücken, Germany.

Simon DOBNIK, Robin COOPER, and Staffan LARSSON (2013), Modelling Language, Action, and Perception in Type Theory with Records, in Denys DUCHIER and Yannick PARMENTIER, editors, *Constraint Solving and Language Processing: 7th International Workshop, CSLP 2012, Orléans, France, September 13–14, 2012, Revised Selected Papers*, volume 8114 of *Lecture Notes in Computer Science*, pp. 70–91, Springer Berlin Heidelberg.

Simon DOBNIK and Erik DE GRAAF (2017), KILLE: a Framework for Situated Agents for Learning Language Through Interaction, in Jörg TIEDEMANN, editor,

Proceedings of the 21st Nordic Conference on Computational Linguistics (NoDaLiDa), volume 131 of *Linköping Electronic Conference Proceedings and NEALT Proceedings Series Vol. 29*, pp. 1–10, Northern European Association for Language Technology (NEALT), Linköping University Electronic Press, Gothenburg, Sweden.

Simon DOBNIK, Christine HOWES, and John D. KELLEHER (2015), Changing perspective: Local alignment of reference frames in dialogue, in Christine HOWES and Staffan LARSSON, editors, *Proceedings of goDIAL – Semdial 2015: The 19th Workshop on the Semantics and Pragmatics of Dialogue*, pp. 24–32, Gothenburg, Sweden.

Simon DOBNIK and John D. KELLEHER (2013), Towards an automatic identification of functional and geometric spatial prepositions, in *Proceedings of PRE-CogSsci 2013: Production of referring expressions – bridging the gap between cognitive and computational approaches to reference*, pp. 1–6, Berlin, Germany.

Simon DOBNIK and John D. KELLEHER (2014), Exploration of functional semantics of prepositions from corpora of descriptions of visual scenes, in *Proceedings of the Third V&L Net Workshop on Vision and Language*, pp. 33–37, Dublin City University and the Association for Computational Linguistics, Dublin, Ireland.

Simon DOBNIK and John D. KELLEHER (2016), A Model for Attention-Driven Judgements in Type Theory with Records, in Julie HUNTER, Mandy SIMONS, and Matthew STONE, editors, *JerSem: The 20th Workshop on the Semantics and Pragmatics of Dialogue*, volume 20, pp. 25–34, New Brunswick, NJ USA.

Simon DOBNIK, John D. KELLEHER, and Christos KONIARIS (2014), Priming and Alignment of Frame of Reference in Situated Conversation, in Verena RIESER and Philippe MULLER, editors, *Proceedings of DialWatt – Semdial 2014: The 18th Workshop on the Semantics and Pragmatics of Dialogue*, pp. 43–52, Edinburgh.

David R DOWTY, Robert Eugene WALL, and Stanley PETERS (1981), *Introduction to Montague semantics*, D. Reidel Pub. Co., Dordrecht, Holland.

Zachary ESTES, Sabrina GOLONKA, and Lara L JONES (2011), Thematic Thinking: The Apprehension and Consequences of Thematic Relations, in Brian ROSS, editor, *The Psychology of Learning and Motivation*, volume 54, pp. 249–294, Burlington: Academic Press.

Ronald FAGIN, Joseph Y. HALPERN, Yoram MOSES, and Moshe Y. VARDI (1995), *Reasoning about knowledge*, MIT Press, Cambridge, Mass.

Christiane FELLBAUM (1998), *WordNet: an electronic lexical database*, MIT Press, Cambridge, Mass.

Gottlob FREGE (1948), Sense and Reference, *The Philosophical Review*, 57(3):209–230.

Klaus-Peter GAPP (1994a), Basic Meanings of Spatial Relations: Computation and Evaluation in 3D Space, in Barbara HAYES-ROTH and Richard E. KORF, editors, *AAAI*, pp. 1393–1398, AAAI Press/The MIT Press.

Klaus-Peter GAPP (1994b), A computational model of the basic meanings of graded composite spatial relations in 3D space, in *Advanced geographic data modelling. Spatial data modelling and query languages for 2D and 3D applications (Proceedings of the AGDM'94)*, Publications on Geodesy 40, pp. 66–79, Netherlands Geodetic Commission.

Simon GARROD and Gwyneth DOHERTY (1994), Conversation, co-ordination and convention: An empirical investigation of how groups establish linguistic conventions, *Cognition*, 53(3):181–215.

Stevan HARNAD (1990), The symbol grounding problem, *Physica D*, 42(1–3):335–346.

Annette HERSKOVITS (1986), *Language and spatial cognition: an interdisciplinary study of the prepositions in English*, Cambridge University Press, Cambridge.

Serena IVALDI, Sao Mai NGUYEN, Natalia LYUBOVA, Alain DRONIOU, Vincent PADOIS, David FILLIAT, Pierre-Yves OUDEYER, and Sigaud OLIVIER (2014), Object Learning Through Active Exploration, *IEEE Transactions on Autonomous Mental Development*, 6(1):56–72.

John D. KELLEHER, Fintan J. COSTELLO, and Josef VAN GENABITH (2005), Dynamically Structuring Updating and Interrelating Representations of Visual and Linguistic Discourse, *Artificial Intelligence*, 167:62–102.

John D. KELLEHER, Robert J. ROSS, Colm SLOAN, and Brian MAC NAMEE (2011), The effect of occlusion on the semantics of projective spatial terms: a case study in grounding language in perception, *Cognitive Processing*, 12(1):95–108.

Geert-Jan M. KRUIJFF, Hendrik ZENDER, Patric JENSFELT, and Henrik I. CHRISTENSEN (2007), Situated dialogue and spatial organization: what, where... and why?, *International Journal of Advanced Robotic Systems*, 4(1):125–138, special issue on human and robot interactive communication.

Lars KUNZE, Chris BURBRIDGE, and Nick HAWES (2014), Bootstrapping Probabilistic Models of Qualitative Spatial Relations for Active Visual Object Search, in *AAAI Spring Symposium 2014 on Qualitative Representations for Robots*, Stanford University in Palo Alto, California, US.

Shalom LAPPIN (2013), Intensions as Computable Functions, *Linguistic Issues in Language Technology*, 9:1–12.

Staffan LARSSON (2015), Formal semantics for perceptual classification, *Journal of Logic and Computation*, 25(2):335–369.

Daniel LASSITER (2011), Vagueness as probabilistic linguistic knowledge, in *Proceedings of the international conference on vagueness in communication (ViC'09)*, pp. 127–150, Springer-Verlag, Berlin, Heidelberg.

Stanislao LAURIA, Guido BUGMANN, Theocharis KYRIACOU, and Ewan KLEIN (2002), Mobile robot programming using natural language, *Robotics and Autonomous Systems*, 38(3–4):171–181.

Emilie L. LIN and Gregory L. MURPHY (2001), Thematic relations in adults' concepts, *Journal of experimental psychology: General*, 130(1):3–28.

Gordon D. LOGAN and Daniel D. SADLER (1996), A computational analysis of the apprehension of spatial relations, in Paul BLOOM, Mary A. PETERSON, Lynn NADEL, and Merrill F. GARRETT, editors, *Language and Space*, pp. 493–530, MIT Press, Cambridge, MA.

Didier MAILLAT (2003), *The semantics and pragmatics of directionals: a case study in English and French*, Ph.D. thesis, University of Oxford: Committee for Comparative Philology and General Linguistics, Oxford, United Kingdom.

Per MARTIN-LÖF (1984), *Intuitionistic Type Theory*, Bibliopolis, Naples.

Cynthia MATUSZEK, Nicholas FITZGERALD, Luke ZETTLEMOYER, Liefeng BO, and Dieter FOX (2012a), A joint model of language and perception for grounded attribute learning, in John LANGFORD and Joelle PINEAU, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)*, Edinburgh, Scotland.

Cynthia MATUSZEK, Evan HERBST, Luke ZETTLEMOYER, and Dieter FOX (2012b), Learning to Parse Natural Language Commands to a Robot Control System, in *Proceedings of the 13th International Symposium on Experimental Robotics (ISER)*.

George A. MILLER and Philip N. JOHNSON-LAIRD (1976), *Language and perception*, Cambridge University Press, Cambridge.

Richard MONTAGUE (1974), *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, New Haven, ed. and with an introduction by Richmond H. Thomason.

Bengt NORDSTRÖM, Kent PETERSSON, and Jan M. SMITH (1990), *Programming in Martin-Löf's Type Theory*, volume 7 of *International Series of Monographs on Computer Science*, Clarendon Press, Oxford.

Martin J. PICKERING and Simon GARROD (2004), Toward a mechanistic psychology of dialogue, *Behavioral and Brain Sciences*, 27(2):169–190.

Matthew PURVER, Jonathan GINZBURG, and Patrick HEALEY (2003), On the means for clarification in dialogue, in *Current and new directions in discourse and dialogue*, pp. 235–255, Springer.

Terry REGIER and Laura A. CARLSON (2001), Grounding spatial language in perception: an empirical and computational investigation, *Journal of Experimental Psychology: General*, 130(2):273–298.

Deb ROY (2002), Learning visually-grounded words and syntax for a scene description task, *Computer speech and language*, 16(3):353–385.

Deb ROY (2005), Semiotic schemas: a framework for grounding language in action and perception, *Artificial Intelligence*, 167(1-2):170–205.

Kristoffer SJÖÖ (2011), *Functional understanding of space: Representing spatial knowledge using concepts grounded in an agent's purpose*, Ph.D. thesis, KTH, Computer Vision and Active Perception (CVAP), Centre for Autonomous Systems (CAS), Stockholm, Sweden.

Danijel SKOČAJ, Matej KRISTAN, Alen VREČKO, Marko MAHNIČ, Miroslav JANÍČEK, Geert-Jan M. KRUIJFF, Marc HANHEIDE, Nick HAWES, Thomas KELLER, Michael ZILLICH, and Kai ZHOU (2011), A system for interactive learning in dialogue with a tutor, in *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2011*, San Francisco, CA, USA.

Luc STEELS and Tony BELPAEME (2005), Coordinating Perceptually Grounded Categories Through Language: A Case Study For Colour, *Behavioral and Brain Sciences*, 28(4):469–489.

Luc STEELS and Martin LOETZSCH (2009), Perspective Alignment in Spatial Language, in Kenny R. COVENTRY, Thora TENBRINK, and John. A. BATEMAN, editors, *Spatial Language and Dialogue*, Oxford University Press.

Mark TUTTON (2013), A new approach to analysing static locative expressions, *Language and Cognition*, 5:25–60.

Matthew E. WATSON, Martin J. PICKERING, and Holly P. BRANIGAN (2004), Alignment of reference frames in dialogue, in *Proceedings of the 26th Annual Conference of the Cognitive Science Society*, Chicago, USA.

Joost ZWARTS and Yoad WINTER (2000), Vector Space Semantics: A Model-Theoretic Analysis of Locative Prepositions, *Journal of Logic, Language and Information*, 9:169–211.

Individuation, reliability, and the mass/count distinction*

Peter R. Sutton and Hana Filip
Heinrich Heine University Düsseldorf

ABSTRACT

Counting in natural language presupposes that we can successfully identify what counts as *one*, which, as we argue, relies on how and whether one can balance two pressures on learning nominal predicates, which we formalise in probabilistic and information theoretic terms: INDIVIDUATION (establishing a schema for judging what counts as one with respect to a predicate); and RELIABILITY (establishing a reliable criterion for applying a predicate). This hypothesis has two main consequences. First, the mass/count distinction in natural language is a complex phenomenon that is partly grounded in a theory of individuation, which we contend must integrate particular qualitative properties of entities, among which a key role is played by those that rely on our spatial perception. Second, it allows us to predict when we can expect the puzzling variation in mass/count lexicalization, cross- and intralinguistically: namely, exactly when the two learning pressures of INDIVIDUATION and RELIABILITY conflict.

Keywords: mass/count distinction, probabilistic semantics, individuation, reliability, semantic learning, information theory, context-sensitivity, Type Theory with Records

1 INTRODUCTION

This paper attempts to combine state of the art research on the mass/count distinction in formal semantics with the cutting edge research in Type Theory with Records that provides a unified representation of cognitive, perceptual, and linguistic information. This allows

*This research was funded by the German Research Association (DFG), CRC 991, project C09. We would like to thank the attendees of the TYTLES workshop at ESSLLI 2015 and the CLASP research seminar. In particular, Robin Cooper, Simon Dobnik, and Shalom Lappin for many helpful discussions.

us not only to unify two largely separate strands of research and enrich both with our novel contributions, but also, and most importantly, to further our understanding of the concept of *individuation* (what counts as one) relative to a predicate, which, as we argue, is the fundamental concept in countability research. The account proposed here covers a number of the complex and puzzling data that pertain to cross- and intralinguistic mass/count variation, which resist an adequate account within the extant theories of the mass/count distinction in formal semantics, to the best of our knowledge.

The outline of this paper is as follows. In Section 2, we introduce the basis for our semantic formalism: Type Theory with Records (TTR, Cooper 2012) and probabilistic type theory with records (prob-TTR Cooper *et al.* 2015). In Section 3, we outline some of the most influential recent theories of the mass/count distinction in formal mereological semantics, which are largely driven by the concept of individuation (what counts as one). Sections 4–7 focus on our new proposal. In Section 4, we enrich prob-TTR with mereological assumptions (probM-TTR). We then show how this formalism can represent, in detail, both qualitative and quantitative criteria for the application of nominal predicates (inspired by Krifka (1989)).

Novelly, building on Dobnik *et al.* (2012), we model how representations of spatial perceptual information in a given context can inform and affect judgements about what counts as an individual (as one) relative to a predicate. In Section 5, we relate the quantitative and qualitative criteria to probabilistic learning and argue that the ability to successfully individuate entities relative to a predicate, and thereby establish a basis for counting, is essentially tied to how one balances two learning pressures. The first is to establish a disjoint individuation schema for a predicate (INDIVIDUATION), the second is to establish reliable criteria for applying a predicate (RELIABILITY). In Section 6, we give a schema for the lexical entries of concrete nouns.

In Section 7, we show how mass and count encoding arises from the balancing of individuation and reliability with respect to nominal predicate learning. One of the advantages of our proposal over merely mereological ones is that the interaction of these two learning pressures also allows us to delimit the range of cases where one should expect to find cross- and intralinguistic mass/count variation in natural language.

2 BACKGROUND: PROBABILISTIC TYPE THEORY WITH RECORDS

2.1 *Type Theory with Records*

TTR integrates rich lexical semantic frame-based representations in the sense of Fillmore (1975, 1976) (and elsewhere) with a compositional semantics in the Frege-Montague tradition. As such, it is an ideal theoretical framework to investigate the lexical semantics of nouns as well as compositional (formal) properties of complex expressions. TTR also integrates the insights of situation semantics insofar as situation types (RECORD TYPES) are taken to be true of situations (RECORDS), rather than being true at possible worlds. The idea is that an agent judges whether a situation *s*, is of type *T*. Such judgements correspond to type theoretic objects, namely *Austinian propositions* (Barwise and Etchemendy 1987) inspired by Austin's (1950/1979) idea that to say something true (in indicative, non-generic cases at least) is to refer to a particular situation with one's utterance that is of the type expressed by the sentence one uses. Sentences can be used to express situation types, and utterances can refer to particular situations.

In the following, we present TTR as presented in Cooper (2012). Record types, which are displayed in tabular format (1), are sets of fields, i.e., ordered pairs, whose first member is a LABEL (to the left of the colon in (1)) and whose second member is a TYPE (to the right of the colon in (1)):

(1)
$$\begin{bmatrix} x & : & T_1 \\ \dots & & \\ y & : & T_n \end{bmatrix}$$

Records, which are displayed in tabular format (2), are sets of fields, i.e., ordered pairs, whose first member is a LABEL (to the left of the '=' in (2)) and whose second member is a VALUE for this label (to the right of the colon in (2)):

(2)
$$\begin{bmatrix} x & = & v_1 \\ \dots & & \\ y & = & v_n \end{bmatrix}$$

An example of a record type is given in (3), which represents the type of situation in which a cat purrs.

(3)
$$\begin{bmatrix} x & : & Ind \\ s_{cat} & : & \langle \lambda v.\text{cat}(v), \langle x \rangle \rangle \\ s_{purr} & : & \langle \lambda v.\text{purr}(v), \langle x \rangle \rangle \end{bmatrix}$$

In the record type in (3), the first field contains a label x and a basic type *Ind*. In TTR, there is a set of basic types, such as *Ind* for *individual*, *Time*, and *Loc* (location). Predicates are functions. This can be seen in the second field. The label s_{cat} is a label for a situation, and the type contains a predicate which is a function from a value v to the type of situation in which this value is a cat. It is important to note that these predicates do *not* take labels as arguments, but rather the *values* of labels. Labels function as pointers to values. We will use an abbreviated conventional notation in this work, as illustrated in (4) (Cooper 2012, p.11):

(4)
$$\begin{bmatrix} x & : & Ind \\ s_{cat} & : & \text{cat}(x) \\ s_{purr} & : & \text{purr}(x) \end{bmatrix}$$

Records are the entities of which record types are true or false ('proofs' of propositions in type-theoretic terminology, 'witnesses' in a natural language setting). For example, (5) specifies a situation in which there is an individual, ***felix***, and two witnesses p and q. Witnesses can be thought of as situations or parts of the world that make type judgements true or false.

(5)
$$\begin{bmatrix} x & = & \text{felix} \\ s_{cat} & = & p \\ s_{purr} & = & q \end{bmatrix}$$

Individuals in the domain are typed. The record in (5) will make the proposition in (3)/(4) true iff:

felix is of type *Ind*,
p is a witness of cat(felix),
q is a witness of purr(felix).

An important aspect of TTR, however, is that it is a semantics that reserves a role for judgements made by agents (see Section 2.2). If the

type in (3)/(4) is T_1 and the situation represented by (5) is s, an agent may judge s to be of type T_1, $(s : T_1)$. This judgement will be true iff the above conditions hold.

Finally, natural language predicates denote properties of type $[x : Ind] \rightarrow RecType$, a function from records containing individuals, to a record type. For example, a simplified representation of *cat* would be as in (6):

(6) $$\lambda r : [x : Ind].(\begin{bmatrix} r.x & : & Ind \\ s_{cat} & : & \text{cat}(r.x) \end{bmatrix})$$

In (6), $r.x$ means that the value to be supplemented is the value of x in r. Hence, if provided with a record $[x = \text{felix}]$, (6) would, via β-conversion, yield the proposition that *felix is a cat* in (7):

(7) $$\begin{bmatrix} s_{cat} & : & \text{cat(felix)} \end{bmatrix}$$

2.2 *Probabilistic Type Theory with Records*

The following two subsections summarise some of the key details of probabilistic TTR from Cooper *et al.* (2014, 2015). The principal difference between prob-TTR and TTR is that judgements are graded as opposed to categorical. Instead of a judgement $s : T$, which is true or false, judgements hold with a probability $p(s : T) = k \in [0, 1]$, or the probability that an agent will judge a situation s to be of type T. The reason for introducing probabilities is to be able to model the inherent gradience in the ways in which we classify parts of the world when we apply predicates of natural language to them. This type of gradedness is directly represented by the inherent gradedness in metalinguistic uncertainty within the range $[0, 1]$.

A major advantage of prob-TTR is that one can model how probability values can be assigned, in a cognitively plausible way, and how probability distributions can be updated via observation and semantic learning. Witnessing how language is used provides localised information that, over time, helps language learners build a probability distribution that guides them how to use language and that approximates the 'true' distribution which underpins how competent speakers use language. This is in contrast to top-down probabilistic approaches that assign (global) probability distributions over sets of possible worlds. This issue is discussed in more depth in Cooper *et al.* (2015).

Meet types and join types in prob-TTR respect the Kolmogorov axioms for probability (Kolmogorov 1950):

(8) $p(a : T_1 \wedge T_2) = p(a : T_1) \times p(a : T_2 | a : T_1)$

(9) $p(a : T_1 \vee T_2) = p(a : T_1) + p(a : T_2) - p(a : T_1 \wedge a : T_2)$

2.3 *Bayesian learning in prob-TTR*

Assuming that agent–learners continually receive new evidence with respect to how to correctly apply types to aspects of the world, this can be modelled, in line with Bayesian approaches to cognition, as continuous updates of the probability distributions in the light of new evidence. The probability distributions of learners will gradually come to be close to those of competent speakers. The way this is modelled in prob-TTR is that agents maintain judgement sets.

In simple and intuitive terms, when an agent makes a judgement about a given situation, an entry in the agent's judgement set is made. Entries in judgement sets record the probability that the encountered situation is of some type. Members of judgement sets are what Cooper *et al.* (2015) refer to as *probabilistic Austinian propositions*.[1] For example, the probabilistic Austinian proposition involving a cat purring, judged with a probability of 0.9 would be:

(10)
$$\left[\begin{array}{lcl} \text{sit} & = & s_1 \\ \text{sit-type} & = & \left[\begin{array}{lcl} x & : & \mathit{Ind} \\ s_{cat} & : & \text{cat}(x) \\ s_{purr} & : & \text{purr}(x) \end{array}\right] \\ \text{prob} & = & 0.9 \end{array}\right]$$

Proposition (10) records a situation, a type and a probability value for the judgement that the situation is of that type. The set of such judgements, i.e., the set of probabilistic Austinian propositions, for

[1] Cooper *et al.* (2015) present a naive Bayesian learning model, a picture that is highly simplified. (They state an intention to develop a more sophisticated learning model.) On the simple model, agents are presented with discrete situations, probabilistic judgements are made and the situation, situation type and judgement value are recorded in the judgement set. A more plausible model would have to incorporate the dynamic development of situations and how judgements will very often be implicit.

an agent A will be the set $\mathfrak{J}$. For a type T, $\mathfrak{J}_T$ is the set of Austinian propositions j such that j.sit-type $\sqsubseteq T$:

(11) $$\mathfrak{J}_T = \{j | j \in \mathfrak{J}, j.\text{sit-type} = T\}$$

The sum of probabilities associated with that type T in $\mathfrak{J}$ is:

(12) $$||T||_{\mathfrak{J}} = \sum_{j \in \mathfrak{J}_T} j.\text{prob}$$

Priors are calculated from sums over entries in the judgement set:

(13) $$\text{prior}_{\mathfrak{J}}(T) = \frac{||T||_{\mathfrak{J}}}{\sum(\mathfrak{J})} \qquad \text{if } \sum(\mathfrak{J}) > 0 \text{ and } 0 \text{ otherwise.}$$

It is worth contrasting this approach with more top-down models in terms of possible worlds. What this system provides, in contrast to more top-down models, is an explanation of how priors are set. In probabilistic possible worlds-based approaches such as that of Eijck and Lappin (2012), one must assume a set of priors over possible worlds from which priors for particular sentences/propositions are calculated. On top of issues of computational tractability (discussed in Cooper *et al.* (2015)), this approach leaves it unexplained on what basis the priors for possible worlds are calculated.

The judgement set also provides a simple way to estimate conditional probabilities such as likelihoods and posteriors. For example, suppose $\mathfrak{J}$ were to contain the record in (10) and the record in (14):

(14) $$\begin{bmatrix} \text{sit} & = & s_2 \\ \text{sit-type} & = & \begin{bmatrix} x & : & \mathit{Ind} \\ s_{cat} & : & \text{cat}(x) \end{bmatrix} \\ \text{prob} & = & 0.9 \end{bmatrix}$$

We could then calculate the probability of there being a situation in which something purrs given it is a cat. $p_{\mathfrak{J},A}$ is a probability function with respect to a judgement set $\mathfrak{J}$, and an agent A. Conditional probabilities are calculated in terms of a type theoretic version of Bayes' Rule:

(15) $$p_{\mathfrak{J},A}(s : T_1 | s : T_2) = \frac{||T_1 \wedge T_2||_{\mathfrak{J}}}{||T_2||_{\mathfrak{J}}}$$

For the case in hand, this yields:

$$(16)\quad p_{\mathfrak{J},A}(s : \begin{bmatrix} x & : Ind \\ s_{cat} & : \text{cat}(x) \\ s_{purr} & : \text{purr}(x) \end{bmatrix} \mid s : \begin{bmatrix} x & : Ind \\ s_{cat} & : \text{cat}(x) \end{bmatrix}) = \frac{0.9}{0.9+0.9} = 0.5$$

which follows given that:

$$\begin{bmatrix} x & : Ind \\ s_{cat} & : \text{cat}(x) \\ s_{purr} & : \text{purr}(x) \end{bmatrix} \sqsubseteq \begin{bmatrix} x & : Ind \\ s_{cat} & : \text{cat}(x) \end{bmatrix}$$

and that if $T \sqsubseteq T'$, then $p(a : T \wedge T') = p(a : T')$. Intuitively, if two situations involving cats are observed with equal probability, but only one is a situation involving purring, then the probability of a situation involving a purring cat, given it involves a cat should be 0.5.

3 BACKGROUND: VAGUENESS, OVERLAP, MASS/COUNT VARIATION AND INDIVIDUATION

In this section, we briefly introduce some of the state of the art semantic accounts of the mass/count distinction in concrete nouns.[2] The accounts we discuss here are all based on enriching formal semantics with mereology (first proposed by Link (1983)). In mereological semantics, domains of entities form a Boolean semilattice closed under sum ($\sqcup$). That is to say: domains of entities are populated not just with individuals (a, b, c), but also with sums of entities ($a, \sqcup b, a \sqcup b \sqcup c$ etc.) which are of the same semantic type. The use of mereology has, since Link (1983), proved highly fruitful in analysing both plurality and the count/mass distinction.

We highlight two key factors that have been proposed to play a role in the mass/count distinction as part of a mereological semantics: (i) vagueness (understood in terms of a kind of context-sensitivity) (Chierchia 2010); (ii) disjointness vs. overlap at a context in a noun's denotation (Rothstein 2010; Landman 2011).

[2] For a more thorough critical analysis, see Sutton and Filip (2016a,b).

3.1 Context sensitivity: vagueness and overlap

Chierchia's (2010) main claim is that mass nouns are *vague* in a way that count nouns are not. Mass nouns are uncountable, because they lack *stable atoms*. Stable atoms are the entities in the denotation of a noun that are atoms in every context relative to a *ground context*. Ground contexts determine the entities that are denotations of a noun at every context (but are not necessarily atoms). If a noun lacks stable atoms (has only unstable individuals), then there is no entity that is an atom in the denotation of the predicate at all contexts.

Nouns such as *rice* are vague in so far as the minimal entities in the denotation of *rice* may vary depending on context. Sometimes they are sums of a few grains, sometimes single grains, half grains, or even rice flour dust. Thus these quantities of rice are in the vagueness band of *rice*. Chierchia (2010) models this vagueness with a supervaluationist semantics. At some total precisifications of a ground context, c, single grains are rice atoms. Where $c \propto c'$ means that c' precisifies c; then at some c' such that $c \propto c'$, half grains are rice atoms. At some c'' such that $c' \propto c''$, rice dust particles are rice atoms. There is, therefore, no entity that is a rice atom at every total precisification of rice. The denotation of *rice* lacks stable atoms, but counting is counting stable atoms, and so *rice* is mass.

Rothstein (2010) focuses on providing a formal model of how nouns such as *fence* and *wall* – which fail to denote entities with "natural units" in the sense of Krifka (1989) – nonetheless behave like ordinary count nouns.[3] Rothstein (2010) coins the term *counting context*, and defines count nouns as typally distinct from mass nouns. Mass nouns are of type $\langle e, t\rangle$. Count nouns, which are of type $\langle e \times k, t\rangle$, denote sets of entity-context pairs (the entity denoted and the context in which it counts as one). To take Rothstein's example, suppose that a square field is encircled by fencing. The question 'How many fences are there?' has no determinate answer, but rather the answer depends on what counts as one disjoint fence in a given context. In some con-

[3] However, Filip and Sutton (2017) argue that nouns such as *fence* are not bona fide count nouns, since they are felicitous as bare singulars in some measure constructions, for example, *"You will need about 150 yards of fence per acre" (BNC)*. Bona fide count nouns are much less felicitous in such constructions. For example, *#I read 10cm of book*.

texts, it would be natural to answer 'four': namely, one for each side of the field. In other contexts, it would be more natural to answer 'one': namely, one fence encircling the whole field. By indexing count nouns to contexts, Rothstein can explain how there can be one single answer to the question of how many fences there are in any particular context, despite *fence* lacking natural atoms in its denotation, i.e., atoms that are independent of counting-contexts. Countability, in Rothstein's view, is a matter of what might be dubbed *disjointness at a counting ccontext.*

In Landman (2011), counting is a matter of *non-overlap in a given context*. He defines a set of generators which contains "the things that we would want to count as one" (Landman 2011, p. 26) relative to a context. Formally, generator sets generate a noun's whole denotation under sum. If the elements in the generator set are non-overlapping, as in the case of count nouns, then counting is sanctioned: counting is counting elements in the generator set and there is only one way to count. However, if generators overlap, as in the case of mass nouns, counting goes wrong, because it leads to a number of different simultaneous counting results. Formally, this is modelled as maximally disjoint subsets which generate the superset under sum (variants). In the above, "a number of different simultaneous counting results" equates to a variation in cardinality across variants. One of Landman's innovations is to provide a new delimitation of the two cases when this happens: *mess* mass nouns like *mud*, and *neat* mass nouns like *kitchenware* (a.k.a. 'object' or 'fake' mass nouns). A noun is a *mess* mass noun if, at every world, its intension determines a regular generator set whose set of minimal elements is overlapping. A noun is a *neat* mass noun if its intension at every world specifies a regular generator set whose set of minimal elements is non-overlapping.

Landman offers an ingenious solution to the perennial problems posed by mass nouns like *kitchenware, furniture, silverware* and the like. Let us take his paradigm example *kitchenware*:

> "The teapot, the cup, the saucer, and the cup and saucer all count as kitchenware and can all count as one simultaneously in the same context. ... In other words: the denotations of *neat nouns* are sets in which the distinction between *singular individuals* and *plural individuals* is not properly articulated." (Landman 2011, pp. 34–35)

The key idea here is that there are contexts which allow overlap in the denotation of a noun N with respect to what counts as 'one N'. In other words, there are contexts in which, either one simply does not apply an individuation schema, or, alternatively, the individuation schema one applies fails to resolve overlap; in either case, overlap is not made 'irrelevant', and therefore counting goes wrong.

3.2 *The puzzle of mass/count variation*

All three of the aforementioned analyses of the mass/count distinction make significant advances in accommodating the puzzling data that display cross- and intralinguistic mass/count variation. However, each account taken individually cannot accommodate the full range of such data. We take five broad classes of nouns as cases in point. Two of these, the prototypical cases, pose no problems for most accounts of the mass/count distinction:

Prototypical objects: Examples in English are *cat*, *car*, *boy*, *chair*. These nouns show a very strong intra- and crosslinguistic tendency towards being count.[4] They are not vague in Chierchia's sense, not counting context-sensitive in Rothstein's sense, and not overlapping in Landman's sense.

Substances: Examples in English are *mud*, *air*, *blood*, *slime*. These nouns show a very strong intra- and crosslinguistic tendency towards being mass.[5] They are vague in Chierchia's sense, not indexed to counting contexts in Rothstein's semantics, and have overlapping minimal generators (are mess mass) in Landman's sense.

Granulars: Examples in English are *lentils*, *rice*, *oats*, *beans*. These nouns show a significant amount of variation in mass/count encoding such as in (17) and (18):

(17) *lentil-s*$_{+C,PL}$; *linse-n*$_{+C,PL}$ (German); *lešta*$_{-C}$ (Bulgarian); *čočka*$_{-C}$ (Czech).

(18) *oat-s*$_{+C,PL}$, *oatmeal*$_{-C}$; *kaura*$_{-C}$ (Finnish); *kaurahiutale-et*$_{+C,PL}$ (Finnish, lit. oat. flake-s).

[4] There are some languages, such as Brazilian Portuguese, which also license a non-coerced mass reading of many count nouns. See Pires de Oliveira and Rothstein (2011) for discussion.

[5] There are some languages, such as Yudja, which also license a non-coerced count reading of many or even all mass nouns. See Lima (2014) for discussion.

Granulars are vague in the sense of Chierchia (2010), but if vagueness were the only factor in mass/count encoding, these data could not be accommodated.[6] Rothstein's account can introduce a typal difference between, for example, *rice* and *lentils*, but does not have the formal tools to explain why a typal distinction should arise commonly for these nouns, but not for, say, prototypical count nouns. Landman (2011) faces a challenge, given that it is unclear why the English *lentil*, for example, should be count, whereas its Bulgarian counterpart *lešta* ('lentil') would presumably come out as either a neat or a mess mass noun (depending on how Landman's theory is applied to this case). That is, it is unclear – given Landman's account – why granulars should license non-overlapping generators in some languages but overlapping generators in others.

Collective artifacts: These nouns, examples of which in English are *furniture*, *kitchenware*, *footwear*, *equipment*, show a significant amount of variation in mass/count encoding, such as we see in (19) and (20):

(19) *furniture*$_{-C}$; *huonekalu-t*$_{+C,PL}$ (Finnish);
meubel-s$_{+C,PL}$, *meubilair*$_{-C}$ (Dutch).

(20) *kitchenware*$_{-C}$; *Küchengerät-e*$_{+C,PL}$ (German, lit. kitchen device-s).

Collective artifacts are recognised to be exceptions to a vagueness based analysis of the mass/count distinction and as requiring a separate source for their mass/count encoding (Chierchia 2010, pp. 136–139). For Landman, collective artifacts constitute the key data points for developing his theory, and to this goal, he focuses on Dutch examples like (19).

Although Landman's theory is not explicitly intended to account for cross-linguistic variation in mass/count encoding, it could be extended to do this job too. A possible line one could then adopt is that mass/count variation is only licensed for neat nouns which can have overlapping generators 'simultaneously in the same context'. If neat mass nouns have overlapping generators simultaneously in the same context, and overlap means MASS, then we may ask why the count noun counterparts of neat mass nouns are count nouns. In Landman's analysis, they have non-overlapping generator sets. However, presum-

[6] Chierchia (2010) is aware of this problem; however it is only informally addressed (Chierchia 2010, p.140).

ably, at different contexts, exactly what counts as one can vary. For example, in some contexts a vanity counts as one *huonekalu* (the Finnish count noun counterpart of the English neat mass *furniture*); in other contexts, it counts as at least two (the mirror and the table etc.).

If the count noun counterparts of neat mass nouns are context sensitive with respect to what counts as one across contexts, then arguably, the count nouns in (19) and (20) are counting context-sensitive in the sense of Rothstein (2010) (just like *fence*). A possible extension to Rothstein's account is that one tends to find cross- and intralinguistic mass counterparts of count nouns that are counting context sensitive. Indeed, the link between these two classes suggests, to us, that the explanation of why count/mass variation is found within them should have a common explanation. We develop these lines of thought in Section 7 in the light of the formal analysis we develop in Sections 4-6.

Non-bounded objects: Examples in English are *fence*, *wall*. These nouns are usually count in their morphologically simple form, but frequently have derived mass counterparts:

(21) $fence_{+C}$ - $fencing_{-C}$; $wall_{+C}$ - $walling_{-C}$

Chierchia (2010) argues that the count versions of these nouns are not vague (with respect to their minimal countable entities), given that the ground context is fixed (Chierchia 2010, pp. 122–123). As such, the mass counterparts provide a challenge to a vagueness-only based account.

Overlap/non-overlap based accounts may fare better when it comes to *non-bounded objects*. Indeed, Rothstein's and Landman's accounts could be extended in a similar way just outlined for collective artifacts. Namely, the mass counterparts of count non-bounded object nouns are neat mass (licensing a count counterpart), and the count counterparts (*fence*) of mass non-bounded object nouns (*fencing*) are counting context sensitive (licensing a mass counterpart). Indeed, the link between the non-bounded objects and collective artifacts classes suggests, to us, that the explanation of why count/mass variation is found within them should have a common explanation.[7] We pursue this in Section 7.

[7] See Sutton and Filip (2016a) for in-depth discussion.

In summary, two ways in which context is important emerge from these three accounts. First, the extension of a noun may vary across contexts with respect to its atomic elements (Chierchia 2010). Second, the entities that 'count as one' in the denotation of a noun may vary across contexts thereby yielding either a disjoint set of individuated entities (Rothstein 2010), or an overlapping set in which all possible individuated units appear simultaneously (Landman 2011).

3.3 *Individuation and two criteria of applicability for nouns*

Here we briefly review how both qualitative and quantitative criteria for the application of noun predicates have been highlighted as important for the semantics of the mass/count distinction and individuation. Specifying these two criteria originates in the work of Krifka (1989), but echoes of it percolate through his later work and that of others. The majority of responses to Krifka's work have focused on improving his representation of the quantitative criteria for the application of count predicates. We will also detail how these qualitative and quantitative criteria come together to feed into an account of individuation in the form of mereotopological properties (Grimm 2012).

Krifka (1989) proposed that the semantic representation of (concrete) count nouns involves two criteria of applicability: one qualitative, and one quantitative. For example, *one/a cow* has the following semantic representation: $\lambda n.\lambda x.[\mathbf{COW}(x) \wedge \mathbf{NU}(\mathbf{COW})(x) = n]$. Intuitively, the quantitative criterion yields what counts as one 'natural unit' in the denotation of a given predicate, and is represented by means of **NU**, standing for a natural unit measure function. Natural unit functions are instances of extensive measure functions and are used to form quantized predicates from cumulative ones.[8] The qualitative criterion of applicability, which is represented by **COW**, qualitatively distinguishes cows from, say, cats, dogs and other entities. In contrast, the semantic representation of (concrete) mass nouns only contains the qualitative criterion of application. For example, the semantic representation for *water* is: $\lambda x.[\mathbf{WATER}(x)]$. This amounts to the claim that there is a typal distinction between mass and count nouns, such that only count nouns involve **NU** a natural unit function. This is motivated by the fact that singular

[8] $\forall P[QUA(P) \leftrightarrow \forall x, y[P(x) \wedge P(y) \rightarrow \neg x < y]]$

count nouns, Krifka argues, have quantized reference, whereas mass nouns do not.

The main responses to Krifka's proposal (to be detailed below) have focused on criticisms of his **NU** function; however something akin to a distinction between qualitative and quantitative criteria remains in most leading accounts (even if the quantitative criterion is often given at a pretheoretical level). The position we will argue for, in line with Grimm (2012), is that a more satisfactory account of individuation requires specifying mereotopological properties.

Zucchi and White (1996, 2001) criticise Krifka's claim that count nouns are semantically quantized. Take, for example, *fence*, *twig*, *line*. They have entities in their denotation whose proper parts also fall under the denotation of *fence*, *twig*, *line*, hence they fail to be quantized. They have a solution in terms of a "maximal participant" relative to situation and a time. On their *Maximal Participant Approach*, determiners such as *a/an* encode a requirement that the entity bound by the existential quantifier is the largest sum individual in the denotation of the V predicate at the event time. On this view, *Alex broke a twig* translates loosely as, *a breaking event whose patient is maximal among the individuals in the denotation of twig broken at the reference time*. Crucially, this does not require that the maximal twig entity is maximal for other events and reference times. The effect, in simple terms, is to make sure that the denotation of the noun is quantized relative to an event and a time.

Whereas Zucchi and White (1996, 2001) emphasise the maximal participant relative to an event and reference time, Rothstein (2010) emphasises that what counts as one varies with *counting context* (Section 3.1). However, on Rothstein's account, what is 'one' is not defined in terms of maximality. Take, for example, fencing around a square field, where what counts as one *fence* need not be the whole enclosure, in each context. Furthermore, *fence* does not denote natural units, since what counts as one varies with context. However, how exactly the set of entities that can count as one are to be delimited remains at a pretheoretical level.

Similarly, as per Landman's account, as we saw in Section 3.2, the only formal restriction on the set of entities that count as one for a predicate is that this set generates the noun's whole denotation. But this means that the criteria deciding the membership of

the set of entities that count as one also remain at a pretheoretical level.

What matters the most for our proposal is that these accounts converge on one key and valuable insight: namely, there is a non-trivial concept 'what counts as one' that underlies the mass/count distinction, albeit treated as pre-theoretical.[9] This insight in fact takes centre stage in Grimm (2012). Grimm argues that mereology is insufficient to define the notion of individual, and that mereology, therefore, must be enriched with topological notions. Mereotopological properties of concrete objects include their part-whole structure, spatial proximity, size, disjointness, adjacency, and shape. Grimm's *mereotopological* theory uses mereotopological predicates in the lexical entries of nouns. For example, for *dog*:

(22) $[\![dog]\!] := \lambda x_o[R(x_o, Dog) \wedge MSSC(x_o)]$

This states that entities in the denotation of the singular count noun *dog* are *Maximally Strongly Self-Connected* (MSSC). x_o is an object variable (as opposed to a kind variable). MSSC is a mereotopological property. An mereological individual "is Maximally Strongly Self-Connected relative to a property if (i) every (interior) part of the individual is connected to (overlaps) the whole (Strongly Self-Connected) and (ii) anything else which has the same property and overlaps it is once again part of it (Maximality))" (Grimm 2012, p. 135).

Our account takes inspiration from Grimm (2012), but we will connect mereotopological properties more directly to formal accounts of perception. In particular, we will address the problematic data of *granulars* like *rice* and *lentils* and argue that the conceptualisation of mereotopological properties can arise out of more domain general perceptual processes.

Instead, building on the suggestion in Krifka (1989) that the application criteria of nouns consist of both a qualitative and a quantitative criterion, we propose that qualitative criteria involve perceptual

[9] Although, arguably, Chierchia (2010) tries to derive 'counting as one' from his supervaluationist semantics, he still assumes a pre-theoretical setting of the 'ground context' which, among other things, ensures that nouns such as *mountain* and *fence* have stable atoms. On Chierchia's (2010) account, different answers to the question 'How many fences are there?' is attributed to their being different ground contexts (Chierchia 2010, pp. 122–123).

properties of objects, which subsume Grimm's mereotopological properties, and functional properties of objects. This is not to say that there are not other properties relevant to individuation, but perceptual and functional properties form the most salient aspects of entities in the denotation of concrete nouns. Here we focus on perceptual properties that concern the spatial organisation entities in the world, and as a case study we take granulars, since granulars present problems for previous theories (Section 3.2). To model this with probM-TTR, we take as a foundation work done by Dobnik *et al.* (2012), because they link spatial knowledge gained by perception with semantic knowledge in a single TTR representation.

4 PROPOSAL: COUNTABILITY AND PROBABILISTIC MEREOLOGICAL TYPE THEORY WITH RECORDS

4.1 *Probabilistic mereological Type Theory with Records*

Thus far, the structure of objects of basic types has been left unspecified. We assume a domain for physical entities that is structured as a Boolean semi-lattice closed under sum. A part of such a domain is given in Figure 1. As is standard in mereological semantics, we assume the operation $\sqcup$ and the relations $<, \leq$.[10]

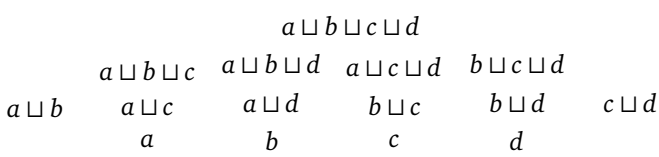


Figure 1: Boolean semi-lattice closed under join

This means that, formally, our enrichment of (prob-)TTR regards the structure of the domain. The principal divergence from TTR and prob-TTR is that we do not assume a basic type *Ind*, but instead only a basic type of *Stuff* for physical entities and individuals; i.e. the type for the whole physical domain. In terms of mereological semantics, this is comparable to adopting the approaches of Krifka (1989) and Landman (2016) who assume a domain unspecified for atomicity (non-atomic). This contrasts with Link's (1983) two domain approach (an atomic

[10] Part relation: $\leq$, where $x \leq y \leftrightarrow x \sqcup y = x$. Proper part relation $<$, where $x < y \leftrightarrow x \leq y \wedge \neg y \leq x$.

domain for count nouns and a non-atomic domain for mass nouns), and also with, for example, Chierchia (2010) and Rothstein (2010) who assume a single atomic domain.

Upward closures of types are defined recursively:

Definition: $^*T$ (The upward closure of a type T under sum)
Where **Type** is the set of types:

1. for any $T \in$ **Type**, $^*T \in$ **Type**
2. for any $T \in$ **Type**, $a : {}^*T$ iff:
   (i) $a : T$
   (ii) or there is some $b, c : {}^*T$ such that $b \sqcup c = a$

For example, if $a, b : T$, then, by (i), $a, b : {}^*T$, and by (ii), $a \sqcup b : {}^*T$.

The advantage of using the tools of probM-TTR is that they will allow us to provide a more nuanced proposal of what it means to be an individual relative to a predicate than those which are found in most mereological approaches to the mass/count distinction. Individuals relative to a predicate are what count as one relative to that predicate (see Section 3.3). TTR provides us with the sufficient tools to combine perceptual, functional, spatial and semantic information within the same representational framework. This will allow us, for example, to show how the same entities can count as a plurality, an aggregate, or even be judged to count as an individual (as one). Such subtle cognitive and perceptual details at the level of our representations allow us to give a formal characterisation of individuation that captures intuitions which are left at the pre-theoretical level in other approaches.

4.2 *Qualitative types*

The qualitative criteria for applying concrete noun concepts will vary greatly from noun to noun not only in values for predicates (like COLOUR, SHAPE, SIZE) used to capture the criteria related to their perceptual properties, or in values for predicates (like USED-FOR-GRINDING) related to their functional properties, but they will also vary with respect to which kinds of criteria are relevant for their application in the first place. Take, for instance, the contrast between natural objects like apples, leaves, trees on the one hand, and artifacts like cars, chairs, buildings, on the other. Whereas *perceptual* properties (from the senses) may be relevant for identifying both natural objects

and artifacts, *functional* properties will play a far bigger role in identifying artifacts. A pile of cushions can count as a chair and a cardboard box can function as a table if, in context, that pile of cushions can aptly function as a chair and the cardboard box can aptly function as a table. In contrast, it is harder to imagine a situation in which some natural object that is not a carrot could count as a carrot even if it fits the same functional role as a carrot does. For instance, even if one uses beetroot or courgette instead of carrot to moisten a cake, one has not, thereby, still made a carrot cake.[11]

We represent such perceptual and functional properties in terms of an all-encompassing type as the one schematised in (23) and exemplified for *rice* in (24). In this respect, we build on a previous proposal in Sutton and Filip (2016b). We assume a basic type *Stuff* that does not distinguish between substances and individuals. Entities of this type may or may not be a clearly demarcated and countable entity, i.e., an individual in our sense.

(23)
$$\begin{bmatrix} x & : & Stuff \\ s_{P_{pptys}} & : & P_{pptys}(x) \end{bmatrix}$$

(24)
$$\begin{bmatrix} x & : & Stuff \\ s_{rice_{pptys}} & : & \mathrm{rice}_{pptys}(x) \end{bmatrix}$$

Instances of the predicate P_{pptys} are placeholders for a wider number of predicates that specify perceptual information such as colour, texture, and, especially for artifacts, functional information (e.g. what activities these items are used for).

Here we wish to expand somewhat on what kind of information the predicate P_{pptys} is a placeholder for, especially with respect to granular nouns such as *rice* and *lentil(s)*. To this end, for the rest of this section we will focus on how qualitative *perceptual* – in particular mereotopological – properties of concrete objects facilitate their classifications under concrete noun predicates (Grimm 2012). We will show how this information can be included in TTR frames, and to this goal, we will also make use of the work done in Dobnik, Cooper and

[11] It is quite plausible that our discussion of perceptual and functional properties mirrors distinctions made in Pustejovsky (1995). For example, Pustejovsky's *constitutive* and *formal* roles seem to approximate our perceptual properties and his *telic* and *agentive* roles seem to approximate our functional properties.

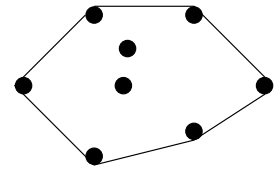
Larsson (Dobnik *et al.* 2012) on the linking of semantic, perceptual and world knowledge in a single TTR representation. Their focus is on the interaction of the inputs from robot perceptual sensors with higher level semantic representations of the robot's environment.

The robot uses a sensor to build a map of points where each point has been classified as being at a particular location in the robot's environment (a point map). Points, in this context, are minimal readings that the robot's sensor makes; the robot builds up a map of its environment by taking point readings. Point maps are represented along the lines of the schema in (25) (Dobnik *et al.* 2012, p. 54).

(25) $$PointMap = \begin{bmatrix} p_1 & : & Point \\ . & & . \\ . & & . \\ . & & . \\ p_n & : & Point \end{bmatrix}$$

Such point maps are then used as the inputs to functions to define bounded regions or volumes which envelop points. These are known as convex hulls, i.e., regions or volumes, and are classified as individuals. The convex hull of a set of points is the smallest convex region containing that set of points. A simple representation of a 2D convex hull of points is given in Figure 2.

Figure 2:
Simple example
of a 2D convex hull of 8 points



Formally, this is represented in (26) (Dobnik *et al.* 2012, p. 55).

(26) $$f : \lambda r : PointMap(\begin{bmatrix} a & : Ind \\ p_1 & : r.Point \\ . & . \\ . & . \\ p_n & : r.Point \\ c_{reg} & : \text{region}(a) \\ c_{inc} & : \text{includes}(a, \langle p_1, ..., p_n \rangle) \\ conv\text{-}hull & : \langle p_i, p_j, p_k \rangle \\ c_{hulled} & : \text{hulled}(\langle p_1, ..., p_n \rangle, conv\text{-}hull) \end{bmatrix})$$

It turns out that the above insights from Dobnik *et al.* (2012) allow us to analyse the problematic data of granulars such as *rice, lentils, peas*, which pose thorny problems for mereological accounts (Section 3).

The basic idea we pursue here is that stuff in the world can be conceptualised in different ways based in part on its perceptual properties. Entities such as, for example, grains of rice can be conceptualised in different ways; this reflects different ways of individuating or otherwise grouping stuff with the relevant rice properties. We highlight three such ways. Granular entities can be (i) individuated in terms of single grains; (ii) grouped in terms of *aggregates* of grains (of some amount or another); (iii) grouped in terms of *bounded aggregates* of grains (portions of grains that form a discrete bounded region or volume in space). Substances such as mud, in contrast, cannot be individuated in terms of anything like grains (mud does not come in clearly perceptible units such as grains). However, stuff like mud can, similar to aggregation, be *amassed* (stuff of some amount with the relevant properties) and conceived of in terms of *bounded amassments* (stuff with the relevant properties that forms some discrete bounded region or volume in space).

We now outline how aggregation and bounded aggregation can be represented in mereological TTR using representations inspired by the work of Dobnik *et al.* (2012). We use rice as a working example. Aggregates with respect to a predicate *rice* involve identifying some plurality of entities each of which has the relevant properties for being grains of rice and judging them to be an aggregate. Unlike Dobnik *et al.*'s convex hulled regions, aggregates need not be grouped into a single discrete region. This is outlined in (27).

(27)
$$f{:}\lambda r{:}\begin{bmatrix} x_1{:}\mathit{Stuff} \\ \vdots \quad \vdots \\ x_n{:}\mathit{Stuff} \end{bmatrix}\left(\begin{bmatrix} c_{\text{rice\_agg}}{:}\,\text{rice}_{\text{agg}}(a) \\ r.x_1 \quad {:}\,\mathit{Stuff} \\ c_{1\_col} \quad {:}\,\text{white}(r.x_1) \\ c_{1\_shape}{:}\,\text{grain\_shaped}(r.x_1) \\ \vdots \qquad \vdots \\ r.x_n \quad {:}\,\mathit{Stuff} \\ c_{n\_col} \quad {:}\,\text{white}(r.x_n) \\ c_{n\_shape}{:}\,\text{grain\_shaped}(r.x_n) \\ c_{agg} \quad {:}\,\text{aggregate}(a) \\ c_{inc} \quad {:}\,\text{includes}(a, \langle r.x_1, \ldots, r.x_n \rangle) \end{bmatrix}\right)$$

The predicate *aggregate* is specified as containing some quantity of entities each of which has some relevant properties (such as white colour or being grain-shaped). This collection is then judged as being a rice aggregate ($rice_{\text{agg}}$). In other words, we can recast mereological sums in terms of an aggregation of, in this case, entities with the requisite rice-grain properties.

Alternatively, we can add extra restrictions on aggregates by requiring that aggregate entities form 'hulled regions'. That is to say that we use the notion of a hulled volume or region as a means of representing mereotopological sum entities. Our novel proposal is that something akin to hulling, namely, carving out chunks or regions out of the parts of the world and judging this region to be a *bounded aggregate* (or alternatively a *bounded amassment* for substance denoting nouns) could model a process of individuation that relies on the spatial (mereotopological) properties of concrete objects: namely, properties having to do with their spatial proximity, disjointness, adjacency, size and shape. This proposal not only capitalises on some insights in Grimm (2012), but is also reminiscent of the longstanding proposals in cognitive semantics (Jackendoff 1991; Talmy 2000) which emphasise spatial notions in the analysis of lexicalization patterns of mass and count nouns.

In the frame in (28), we assume that mereotopological sums of bounded entities may be represented via a similar mechanism to 'region hulling', namely, aggregating identified portions of stuff each of which have certain physical properties.

(28) $$f : \lambda r : \begin{bmatrix} x_1 : \textit{Stuff} \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ x_n : \textit{Stuff} \end{bmatrix} \left(\begin{bmatrix} c_{\text{rice\_agg}} & : \text{rice}_{\text{bounded}}(a) \\ r.x_1 & : \textit{Stuff} \\ c_{1\_col} & : \text{white}(r.x_1) \\ c_{1\_shape} & : \text{grain\_shaped}(r.x_1) \\ \cdot & \quad \cdot \\ \cdot & \quad \cdot \\ r.x_n & : \textit{Stuff} \\ c_{n\_col} & : \text{white}(r.x_n) \\ c_{n\_shape} & : \text{grain\_shaped}(r.x_n) \\ c_{reg} & : \text{region}(a) \\ c_{inc} & : \text{includes}(a, \langle r.x_1, ..., r.x_n \rangle) \\ \textit{conv-agg} & : \langle r.x_i, r.x_j, r.x_k, ... \rangle \\ c_{b\_agged} & : \text{agg}(\langle r.x_1, ..., r.x_n \rangle, \textit{conv-agg}) \end{bmatrix} \right)$$

We can perceptually identify the collection of grains of rice as being comprised of individual grains; however, at the same time this collection can be viewed as a bounded entity in a manner akin to hulling. Importantly, just as with hulling collections of perceptual points to identify entities, there will be restrictions on what kinds of collections of entities will be identified as bounded aggregates. One such restriction will be that the entities that form the aggregate cannot be too dispersed and so must be relatively clustered together. Such intuitions also motivate how some of the mereotopological restrictions on granular and collective aggregates in Grimm (2012) can be represented.[12]

The function in (28) mirrors that in (26); however the function is from a record of a type of having more than one physical entity (or bit of stuff), rather than in terms of perceived points. The function determines a bounded aggregate then yields a new entity judged to be of type $rice_{\text{bounded}}$ that 'collates' the physical entities in this region. The bits of stuff have properties such as colour and shape. c_{inc} labels a function that selects which of these is to be included in the region. The *conv-agg* tuple determines the entities around which the boundaries of the convex aggregate will be 'drawn'. c_{b_agged} is the condition that all the entities in the region are within the bounds of the boundary.

So, similarly to defining a convex hull in terms of perceived points, this function defines a *bounded aggregate* in terms of entities that have already been classified as physical entities.[13] Intuitively, if a situation contains many small entities (such as lentils or grains of rice) that are in close proximity to one another, this function picks them out as a convex aggregate – an aggregate falling within what is judged to be a certain bounded area of space – and then classifies this as $rice_{\text{bounded}}$.

For substance denoting nouns such as *mud* or *blood*, a similar function could be defined. However, instead of aggregating grains into bounded aggregates, it would hull stuff with the relevant properties into a bounded amassment.

[12] Examples of collective aggregates in Grimm (2012) are names for insects (found in swarms or groups) and berries. Examples of granular aggregates are *rice* and *sand*.

[13] It will be that something akin to (26) is also needed to classify what, in a perceptual field, is to be identified as of the type *Stuff*, albeit at a 'lower' level.

4.3 *Quantitative functions relative to a predicate*

Given a representation of the qualitative properties of an entity or some collection of entities, we may assign some quantity value to the entities of that type. This is the role of a quantitative function, which is of the type in (29), a function from record types specifying qualitative criteria for applicability to real numbers.

(29) $$f_{P_{quant}} \quad : \quad \left(\begin{bmatrix} x & : & Stuff \\ s_{P_{pptys}} & : & P_{pptys}(x) \end{bmatrix} \rightarrow \mathbb{R}\right)$$

Quantitative functions are relative to predicates (different functions are defined for different predicates), since the same entity or entities may count as 'one' relative to one predicate, but not another. For example, 52 playing cards could be judged to have a quantity of 1 with respect to a predicate *deck of cards*, but not with respect to the predicate *card* (Link 1983). However, because we are not assuming a pretheoretical notion of individuation, how some stuff will be quantified may depend on what counts as an individual relative to that predicate. Our strategy is to derive individuation from a special case in which a quantitative function outputs 1. Competing schemas for individuation will be represented as competing quantitative functions. These competing quantitative functions differ with respect to what perceptual and functional properties are required to measure 1 (count as one). For example, take the record type in the right hand side of (27) as compared to the one below in (30).

(30) $$\begin{bmatrix} x_1 & : Stuff \\ c_{1\_col} & : \text{white}(x_1) \\ c_{1\_shape} & : \text{grain\_shaped}(x_1) \end{bmatrix}$$

There is more than one possible way to try to individuate the stuff or collections of stuff with rice-like properties (being white, grain shaped etc.). We give three cases by way of example.

Case 1: One possible quantitative function would output a value 1 for the type in (30). This function would individuate single grains. Applied to something like the type in (27), which contains multiple entities with the requisite properties that have been judged to be an aggregate, this function could use, for example, the approximate size of the aggregate to output an approximate quantity value. We do not assume that these functions are mere cardinality functions. In fact, it

would be cognitively implausible to do so. Take the case of some collections of rice grains. For larger collections that have been judged to be aggregates (Section 4.2), we do not assume that the quantity value will reflect the number of rice grains exactly, since we are not in a position to know this without explicitly counting. However, for small numbers of grains, such as an aggregate entity comprised of three grains, this function could return a value where the output number of the function equals the number of grains. Whether the output of the quantitative function reflects the exact number of grains or some approximation, we suggest, could be grounded in the distinction found in psychology between the *approximate number system* (ANS) and the *parallel individuation system* (PI) (Hyde and Spelke 2011, and references therein). In brief, both of these systems are supposed to be developed pre-linguistically. The difference between them is that PI operates accurately in individuating entities, but is severely limited in terms of number. It operates accurately up to about four entities, and is assumed to involve the representation of all entities individually. ANS works on much larger numbers of entities but is assumed to represent entities as collections, not individually. ANS works effectively as a way of discerning differences in number between collections, but not as an accurate representation of cardinality. If the quantitative function is constrained by these systems, its numerical output would be an accurate measure of cardinality of pluralities up to about four entities, but only an approximation of cardinality for larger collections (about 5, about 10, ..., about 50, about 100). What the output of this function is, we suggest, could be modeled in relation to factors such as the size and density of the aggregate identified in the situation. So an output of e.g. 1 would indicate *exactly 1*, but an output of e.g. 10, would indicate *approximately 10*.

Case 2: An alternative quantitative function could individuate, not single grains, but clusters, such that any (sufficiently large) aggregates containing entities that are individually white, grain shaped, etc., would measure 1. This function, applied to the type in (27) would output 1, but applied to the type for a single grain in (30) would measure either a value less than one (or, alternatively, could be undefined). This would allow for the possibility that overlapping collections of rice grains could each be judged to be an aggregate and so each be measured by the quantitative function as one.

Case 3: Another alternative quantitative function could individuate, not single grains or aggregates, but bounded aggregates: any clusters of entities that are individually white, grain shaped etc., but also form a discrete bounded region would measure 1. This function, applied to the type in (28) would output 1. The additional boundedness condition, in effect, treats any discrete regions filled with rice grains as entities to be counted.

These cases represent different ways of individuating rice. The first quantitative function 'finds' individual grains, and if more than one is present approximates a quantity. The second function 'finds' collections of grains and groups them as an aggregate entity. The third function 'finds' bounded regions or clusters of grains and groups them as a bounded aggregate. In Sections 5–7, we will argue that the fact that there can be competing individuation schemas can be used in conjunction with information theoretic requirements, to explain count/mass lexicalization patterns cross- and intralinguistically.

The special case for the application of a quantitative function will therefore be where the output is 1. In the case of *cat* this would indicate a type of individual cats. For granular nouns such as *rice* or *lentils*, this could be the type for individual grains of rice or individual lentils.[14] In this sense, the special case where a quantitative function returns a value of 1 marks the INDIVIDUATION SCHEMA for a predicate. We introduce the following notational convention for the special case to act as both an abbreviation and as an mnemonic for this individuating role:

$$(31)\quad \left[s_{cat\text{-}ind} : \begin{bmatrix} s_{cat_{qual}} & : & \left[\; s_{\mathbf{cat}_{pptys}} \; : \; \mathrm{cat}_{pptys}(x) \; \right] \\ f_{cat\text{-}quant} & : & \left(\left[\; s_{\mathbf{cat}_{pptys}} : \mathrm{cat}_{pptys}(x) \; \right] \rightarrow \mathbb{R} \right) \\ f_{cat\text{-}quant}(s_{cat_{qual}}) & : & \mathbb{R}_1 \end{bmatrix} \right]$$

$$= [s_{cat\text{-}ind} : \mathrm{cat}_{Ind}(x)]$$

In other words, the type of situation in which for some predicate P, a physical entity (or sum) is judged to have a quantity of 'one' is a type of situation in which one judges that thing to be a P-individual. To emphasise, this means that we do not take being an individual as a basic notion, but rather as a classification task. We assume a basic

[14] We discuss other nouns in detail in Section 7.

type of physical entities (and the upward closure of this type), but which of these (collections of) physical entities are individuals is both relative to a predicate and a non-trivial question.

An important restriction for any P_{Ind} predicate associated with a count noun is that the entities of this type are disjoint (do not overlap mereologically). In other words, an entity that is judged to be 'one' P-individual cannot also be judged to be of some larger quantity value with respect to P under the same individuation schema if that individuation schema is to form the basis for grammatical counting.

4.4 *Individuation schemas and learningto apply predicates in context*

It is important to note that there may be cases where individuating, relative to a predicate and a quantitative function, may not always guarantee felicitous application of the predicate. Chierchia (2010) points out that for many mass nouns, whether or not some entity falls under the denotation of that noun can depend on the context. For example, take a collection of around ten grains of rice. In the context of cooking dinner, one can truly say "We have no rice" when the ten grains are all that remains of a once full packet, and a child can felicitously say "I have eaten all the rice" when only ten grains remain on the plate. However, when around ten grains have fallen in the same context, we can felicitously and truthfully state "I spilled some rice on the floor". Stating "There is rice in this dish" is felicitous and truthful, when uttered by someone with a severe rice allergy, for example, even if it contains only about ten grains of rice.

Assuming a quantitative function (labeled $f_{P_{quant}}$) for a predicate P, this kind of context sensitivity can be represented as the calculation of conditional probabilities of the form in (32). The learner-agent A must identify which qualities (specified in the qualitative criterion type) and which quantities of these entities (relative to an individuation schema) maximise the probability of applying the predicate relative to her judgement set $\mathfrak{J}$.

$$(32)\quad p_{\mathfrak{J},A}(r:\begin{bmatrix} x:Stuff \\ s_P:\mathrm{P}(x)\end{bmatrix} \mid r:\begin{bmatrix} s_{P_{qual}} & :\begin{bmatrix}\text{qualitative}\\ \text{criterion type}\end{bmatrix} \\ f_{P_{quant}} & :(\begin{bmatrix}\text{qualitative}\\ \text{criterion type}\end{bmatrix})\rightarrow\mathbb{N} \\ i & :\mathbb{N} \\ f_{P_{quant}}(s_{P_{qual}}) & :\mathbb{N}_i\end{bmatrix})$$

We assume the data for these judgements come from, in part, witnessing competent speakers' judgements with regard to applying predicates. We now give a simple example for *rice*.

Suppose that a learner is exposed to two situations in which an adult speaker provides her with evidence for how to make *rice* judgements. Furthermore, the agent is employing a schema/quantitative function that individuates single grains. In $s_{\text{making dinner}}$, there are, by the agents' estimations, approximately 10 grains (say at the bottom of a packet). The adult speaker may say, looking at the packet, *Oh no! We have no rice left.* This constitutes evidence that the small quantity of around 10 grains is not sufficient to count as *rice*. In s_{allergy}, a similar quantity of grains falls into the soup. The soup is for someone with a rice allergy and the adult speaker says *Oh no! Rice fell into the soup*. This constitutes evidence that the small quantity of around 10 grains is sufficient to count as *rice*.

In terms of learning data, these utterances in context provide conflicting information as to whether or not a collection of around ten grains counts as *rice*. For the case we just informally described, this could result in a judgement set containing the probabilistic Austinian propositions in Figure 3. The figure itself contains judgements for different situations (labelled *sit*). These situations are meant to represent the contexts just described. The idea is that the quantity of grains

Figure 3: Possible (partial) judgement set for around 10 grains of rice.

$$j_1 = \left[\begin{array}{ll} \text{sit} & = s_{\text{making dinner}} \\ \text{sit-type} & = \left[\begin{array}{ll} x & : \mathit{Stuff} \\ s_{rice} & : \neg\text{rice}(x) \\ s_{rice_{qual}} & : \left[\begin{array}{l} s_{rice.col} : \text{white}(x) \\ \ldots \end{array}\right] \\ f_{rice_{quant}}(s_{rice_{qual}}) & : \mathbb{N}_{10} \end{array}\right] \\ \text{prob} & = 0.9 \end{array}\right]$$

$$j_2 = \left[\begin{array}{ll} \text{sit} & = s_{\text{allergy}} \\ \text{sit-type} & = \left[\begin{array}{ll} x & : \mathit{Stuff} \\ s_{rice} & : \text{rice}(x) \\ s_{rice_{qual}} & : \left[\begin{array}{l} s_{rice.col} : \text{white}(x) \\ \ldots \end{array}\right] \\ f_{rice_{quant}}(s_{rice_{qual}}) & : \mathbb{N}_{10} \end{array}\right] \\ \text{prob} & = 0.9 \end{array}\right]$$

to count as *rice* could vary across these contexts. The situation types assign a measure to stuff with the relevant rice properties (such as colour) and a condition that the stuff is rice. The probability values represent the extent to which the situations are of that type.

We assume that the learner has learned these judgements directly from the competent speaker and so attributed a high value (0.9) to all of them.[15] Using the prob-TTR version of Bayes' rule (15), these judgements then allow the calculation of the probability of something being rice, given that it has a quantitative function value (relative to *rice*) of 10. This is shown in (33).[16]

$$
\begin{aligned}
(33)\quad & p_{\mathfrak{J},A}(s : [s_{rice} : \mathrm{rice}(x)] \mid s : \begin{bmatrix} s_{rice_{qual}} & : \begin{bmatrix} s_{rice.col} : \mathrm{white}(x) \\ \dots \end{bmatrix} \\ f_{rice_{quant}}(s_{rice_{qual}}) & : \mathbb{N}_{10} \end{bmatrix} \\
& = \frac{\left\| [s_{rice} : \mathrm{rice}(x)] \wedge \begin{bmatrix} s_{rice_{qual}} & : \begin{bmatrix} s_{rice.col} : \mathrm{white}(x) \\ \dots \end{bmatrix} \\ f_{rice_{quant}}(s_{rice_{qual}}) & : \mathbb{N}_{10} \end{bmatrix} \right\|_{\mathfrak{J}}}{\left\| \begin{bmatrix} s_{rice_{qual}} & : \begin{bmatrix} s_{rice.col} : \mathrm{white}(x) \\ \dots \end{bmatrix} \\ f_{rice_{quant}}(s_{rice_{qual}}) & : \mathbb{N}_{10} \end{bmatrix} \right\|_{\mathfrak{J}}} \\
& = \frac{0.9}{0.9 + 0.9} \\
& = 0.5
\end{aligned}
$$

Given the judgements in Figure 3, the result is 0.5. A gloss on the importance of this value is that the learner has as much reason to believe that the predicate *rice* can be applied to around 10 grains of rice as she does for not thinking so, given her judgement set. By "around 10", we mean that the output of the quantitative function may only approximate actual numbers of grains for quantities above that which can be directly quantified via the PI (parallel individuation) cognitive system (Section 4.3).

[15] In a more sophisticated model, reflected in the probability value should be that the evidence for $s_{\text{making dinner}}$ is indirect (making $s_{\text{making dinner}} : \neg\text{rice}$ less certain). A promising route would be to adopt something akin to Lassiter's representation of indirect evidence in terms of Bayesian networks (Lassiter 2016).

[16] We assume, following Cooper *et al.* (2015), that negation is classical. Instances of $[x : \mathit{Stuff}]$ have been suppressed for brevity.

However, across contexts in which there are much larger quantities of grains present, the judgement set may be far more consistent. For example, a learner will rarely experience a whole packet of rice not being judged as rice, i.e., not falling under the predicate *rice*, and so is far more confident about classifying larger amounts of rice as *rice*. In short, in this way we capture the observation by Chierchia (2010) that when it comes to classifying with granular nouns like *rice*, quantity matters. Independently of context, one may not be safe to classify ten grains as *rice*, but one could, with high confidence, classify a packet of rice as *rice*.

This means that, although an individuation function for *rice* that identifies single grains does succeed in identifying disjoint (potentially countable) entities, it is not wholly reliable when applied across contexts to establish, with a high degree of certainty, when to apply the predicate *rice*. A more reliable schema could be found by opting for an individuation schema that picks out larger collections of rice grains. Such a move could end up failing to properly individuate disjoint entities suitable for counting, however (since larger collections overlap). In Section 5, we will show how this tension can be formally captured within probM-TTR.

5 THE LEARNING PRESSURES OF RELIABILITY AND INDIVIDUATION

Given the insights of formal (mereological) theories (Section 3), we propose that in addition to identifying a reliable criterion for applying a predicate, learners also seek to identify an individuation schema for a predicate. This means that pressures on nominal predicate learning will be at least twofold:

(i) RELIABILITY: to establish with a high degree of certainty when to apply a predicate.

(ii) INDIVIDUATION: to establish (if possible) an individuation schema for a predicate.

In some cases, these two pressures will operate in unison, for example, for *cat*, accurately judging a situation to contain one or more cat-individuals is a very good ground to judge those entities as falling under the number neutral predicate *cat*. However, as we shall argue,

this is not always the case for other predicates. For example, individual rice grains are the clearly individuable units for counting rice, but the presence of a single grain is not a reliable criterion for applying *rice* since there are many contexts in which a single grain is not a sufficient quantity to count as *rice*. Furthermore, we argue in Section 7 that tensions between these two pressures generate exactly the cases where we find cross- and intralinguistic mass/count variation.

5.1 *Formalising the requirement of reliability*

Reliability is a pressure on a learner to find a set of properties that reliably predict when to apply a predicate. We have proposed that these properties include both qualitative and quantitative criteria. Reliability itself is therefore a balance between using a P_{Ind} predicate, the upward closure of which ($^{*}P_{Ind}$) includes too much in P; and using a P_{Ind} predicate, the upward closure of which ($^{*}P_{Ind}$) includes too little in P. Of course, the ideal balance means using a P_{Ind} predicate that neither includes too much nor too little in P. In other words, the most reliable individuating predicate will be one that maximises the conditional probabilities in (34) and (35).[17]

(34) $\mathit{Max}_j(p(s : [\ s_P\ :\ P(x)\ ] \mid s : [\ s_{P\text{-}ind}\ :\ {}^{*}P_{Ind_j}(x)\ ]))$

(35) $\mathit{Max}_j(p(s : [\ s_{P\text{-}ind}\ :\ {}^{*}P_{Ind_j}(x)\ ] \mid s : [\ s_P\ :\ P(x)\ ]))$

Maximising the probability in (34) means that being a P-individual or a sum of P-individuals is a very strong indicator of being a P. This militates against the over-inclusivity of P_{Ind_j}. Maximising the probability in (35) means that being a P is a very strong indicator of being a P-individual or a sum of P-individuals. This militates against the under-inclusivity of P_{Ind_j}. Balancing these two (optimally maximising both probabilities) should result in as close an approximation of P and $^{*}P_{Ind}$ as possible. In the trivial case, this would just be to use P as P_{Ind}. However, in most cases, doing this would fail to individuate any entities at all.

To make this clearer, take the three simple cases which are graphically represented in Figure 4. (i) This represents the case where the application conditions for the predicate are perfectly matched to the application conditions for the upward closure of the P_{Ind} predicate,

[17] The specifications of $x : \mathit{Stuff}$ here and further below are omitted for brevity.

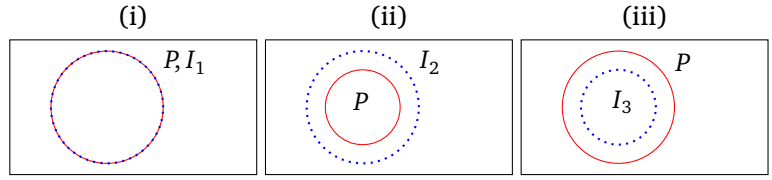


Figure 4: Maximising both conditional probabilities vs. maximising one.

$P = [\ s_P\ :\ P(x)\ ]$ (solid line) and
$I_j = [\ s_{P\text{-}ind}\ :\ {}^*P_{Ind_j}(x)\ ]$ (dotted line)

therefore both conditional probabilities (34) and (35) are maximised. (ii) This represents the case where there are some things which are P individuals or sums thereof that are not correctly judged as P. This means the the probability in (35) is maximised, but the probability in (34) is not. (iii) This represents the case where there are some things which are correctly judged as P which are not P individuals or sums thereof. This means that the probability in (34) is maximised, but the probability in (35) is not.

5.2 *Formalising the requirement of individuation*

The pressure that can push in the opposite direction to reliability is *individuation*. This pressure can be derived from a more general pressure towards informativeness (Piantadosi *et al.* 2011). The main idea in the context of countability is that disjoint individuation schemas/predicates P_{Ind} have minimum entropy with respect to determining counting results compared with predicates that are not disjoint. The reason for this, building on Landman's (2011) insights, is that when we have an overlapping set of entities, there are multiple answers to the question 'how many?'. Uncertainty over how many things (relative to a predicate) there are equates to a higher level of entropy compared with a single answer.

In order to formally capture the pressure towards individuation, we will define a probabilistic notion of disjointness of a type, and then relate this to minimising entropy (with respect to the disjoint variants of a type). The (probabilistic) notion of disjointness which will be used below is a condition for the maximal individuation of P_{Ind} predicates. This follows the standard mereological notion of disjointness, but adds the condition that the only relevant entities are those that are of the

relevant type with sufficient amounts of certainty. We formalise 'sufficient degree of certainty' using a probability threshold θ.

> A type T is mereologically pairwise disjoint relative to a probability threshold θ iff:
> $\forall x, y[(p(x:T) \geq \theta \wedge p(y:T) \geq \theta) \rightarrow \neg\exists z[z \leq x \wedge z \leq y]]$

In words, any two entities, taken pairwise, that are of a type with a probability above the threshold cannot share a part with one another.

Disjoint types have only one maximally disjoint subtype (akin to variants in Landman (2011)), namely the type itself. For types that are not disjoint, one can form, possibly multiple, maximally disjoint subtypes. For example, if $a, b, a \sqcup b : T$, then, relative to $a, b, a \sqcup b$, there are two maximally disjoint subtypes v_1 and v_2 such that $a, b : v_1$ and $a \sqcup b : v_2$, but where $a \sqcup b \not: v_1$ and $a, b \not: v_2$.

The pressure of individuation can be modelled as pushing towards the use of a disjoint P_{Ind} type. At a first pass, we could, therefore, suggest that the pressure of individuation is a requirement merely to minimise entropy as in (36).

(36) $$\mathit{Min}_j\Big(-\sum_{v_i \in V} p(v_i|P_{Ind_j}) \times \log p(v_i|P_{Ind_j})\Big)$$

Here, entropy values give the average amount of information needed to determine a specific counting result. For example, assuming an equal distribution over variants, and a base-2 logarithm, numbers of variants and entropy values would be as follows:[18]

Number of variants	1	2	4	8	16
Entropy	0	1	2	3	4

The effect is that minimising entropy pushes towards a disjoint P_{Ind} predicate because disjoint predicates have an entropy of 0.

However, the definition in (36) misses some details. As we have seen, there are nouns such as *fence*, *twig*, *line* which display context-sensitivity with respect to what counts as a single individual (focused on by Zucchi and White (1996, 2001) and Rothstein (2010)). If the

[18] For example, if there are four variants such that $p(v_1|P_{Ind_j}) = p(v_2|P_{Ind_j}) = p(v_3|P_{Ind_j}) = p(v_4|P_{Ind_j}) = 0.25$, then the surprisal for each variant equals $0.25 \times \log_2 0.25 = -0.5$ and entropy equals $-(-0.5 + -0.5 + -0.5 + -0.5) = 2$.

Rothstein-type analysis for such nouns is correct, then, at every (default) context, there will only be a disjoint set of fence entities, even if across contexts, these entities overlap. The denotations of prototypical count nouns, such as *cat* and *chair*, intuitively, have inherently individuated denotations, unlike the denotations of count nouns such as *fence* that require context to identify countable entities in their denotation. We should, therefore, include, in the calculation for minimising entropy, some cost C that increases entropy in relation to the number of admissible (disjoint) P_{Ind} predicates. This is given in (37).

(37) $$\mathit{Min}_j\Big(-\Big(\sum_{v_i \in V} p(v_i|P_{Ind_j}) \times \log p(v_i|P_{Ind_j})\Big) + C\Big)$$

To give an example, let us compare three cases. Context general P_{Ind} predicates are predicates that can be applied to correctly individuate Ps across all contexts.

(i) A context general, disjoint P_{Ind} predicate. This will apply to nouns with individuation schemas that pick out naturally atomic, clearly disjoint objects. In this case, there is only one variant, P_{Ind} itself, so $\log p(v_i|P_{Ind_j})) = 0$, and there is only one P_{Ind} predicate, so $C = 0$. Applying (37) gives an entropy value of 0.

(ii) A context general, not-disjoint P_{Ind} predicate with, for instance, four variants. This will apply to nouns such as *furniture*, which, following the analysis in Landman (2011), have denotations in which what counts as one overlaps in the same context. If the four variants are equally probable, conditional on P_{Ind}, this would make $\sum_{v_i \in V} p(v_i|P_{Ind_j}) \times \log p(v_i|P_{Ind_j})) = 2$. There is only one P_{Ind} predicate, so $C = 0$. The total entropy value will thus be 2 (see footnote 18).

(iii) A collection of two disjoint P_{Ind} predicates that each only apply in specific contexts. This will apply to nouns, such as *fence* and *huonekalu* ('furniture', Finnish, count), where what counts as one varies from context to context. Each context-specific P_{Ind} predicate is disjoint and has only one variant (itself), so $\sum_{v_i \in V} p(v_i|P_{Ind_j}) \times \log p(v_i|P_{Ind_j}) = 0$ for each j. However, given that there are, in this simple example, two P_{Ind} predicates (one for some contexts, the other for the other contexts), one of which must be chosen in each context, there is some cost added. The final result will be the value of C.[19]

[19] Arguably, C should itself be sensitive to the probability that a context selects

On its own, individuation pushes towards finding a single disjoint P_{Ind} predicate and sticking with it across contexts. However, if individuation were the only factor, then any arbitrary disjoint P_{Ind} predicate would suffice as an individuation schema, since there would be no requirement for this schema to reliably identify *P*s. Clearly, the upward closure of this predicate type should predict, with reasonable certainty, when to apply P. This is what the competing pressure of *reliability* ensures.

5.3 *Summary of reliability and individuation*

Considered independently, the pressures of *individuation* and *reliability* are each insufficient to capture a good criterion for applying a predicate P. Reliability alone would not ensure adequate informativeness for individuation (entropy would be high, on the assumption made here that disjoint individuation schemas/predicates P_{Ind} have minimum entropy compared to predicates that are not disjoint). Individuation on its own would not ensure adequate reliability (what counts as one P in some contexts is not a reliable indicator for what counts as one P in another, hence a single individuation schema will not reliably indicate what counts as P across contexts).

In Sections 6–7, we will see that some nouns allow a ready balance between these pressures. These nouns will turn out to be those that are fairly stably lexicalized as count cross-linguistically e.g. *cat* and *chair*. Other nouns will exemplify how these two pressures can be in direct conflict. Resolution of this conflict can only come by prioritising one pressure or the other. In these cases, the count/mass encoding of the noun will reflect which pressure is prioritised, and most importantly, as we argue, predicts the variation in mass/count encoding across different languages (e.g., the mass noun *furniture* and the Finnish count noun *huonekalu* ('furniture')) as well as within a particular language. In yet a third case, we will argue that individuation cannot really be satisfied. This case motivates the 'stubborn' encoding of nouns as mass, as we find with prototypical mass nouns, such as *mud* and *blood*.

---

a particular P_{Ind} predicate. For example, there should, reasonably, be a lower cost for a situation where the same individuation schema is selected 99 percent of the time, versus the situation in which two schemas are equally probable. We leave inclusion of such factors to further work.

6 THE SEMANTICS OF CONCRETE NOUNS IN PROBM-TTR

Some recent theories of the mass/count distinction propose to represent lexical entries of nouns as pairs where one projection of the pair determines the standard denotation of a noun and the other determines the counting base and/or individuation schema to apply to that noun (Rothstein 2010; Landman 2011, 2016; Sutton and Filip 2016a). In keeping with this general idea, we represent lexical entries of common nouns as a complex frame in which there are two record types.

One specifies the situation type for the predicate being learned, for instance, cat(x), which is a number neutral predicate a learner associates with situations in which competent speakers make judgements that something is a cat or that some things are cats. We label this s_{pred}. The learner simultaneously learns how such number neutral predicates correspond to the perceptual and functional properties of the objects they witness competent speakers referring to.

Representations of such perceptual and functional properties are encoded in the other part of the lexical entry which we label s_{c_base}. This specifies the counting base for this predicate and includes both the quantitative and qualitative criteria of application for the number neutral predicate in the sense of Krifka (1989). The disjointness of this type is what enters into calculations regarding individuation (Section 5.2). The upward closure of this type is what enters into the reliability calculations (Section 5.1). The motivation for this bipartite lexical structure is that a learner requires both kinds of information in order to learn how to use natural language predicates such as *cat*.

A schema for a noun entry is given in (38). In the frame-based representation offered by TTR, these two record types feature as parts of the same complex type and so can be abstracted over so as to receive the same values when applied to a record containing some physical entity (e.g. an entity such as felix labelled x in the record r in (38)).

(38) $\lambda r : [x : \mathit{Stuff}].\ p\left(\begin{bmatrix} s_{pred} : \begin{bmatrix} \text{Rec. type for predicate} \\ \text{(contains label } r.x\text{)} \end{bmatrix} \\ s_{c\_base} : \begin{bmatrix} \text{Rec. type for counting base} \\ \text{(contains label } r.x\text{)} \end{bmatrix} \end{bmatrix}\right)$

We also adopt the basic structure for a common noun lexical entry in prob-TTR given by Cooper *et al.* (2015)[20] for which the result of applying a record of the requisite type is the probability of the relevant record type.

The simplest case is that of prototypical count nouns such as *woman*, *cat*, *chair*, *car*. They are associated with a record type that includes the relevant number neutral predicate, and a record type for the counting base which contains the relevant P_{Ind} predicate. For example, the entry for *cat* is given in (39), and the entry for *cats* is given in (40).

(39) $[\![\text{cat}]\!] = \lambda r : [x : Stuff].p(\begin{bmatrix} s_{pred} & : & [\ s_{cat} : \text{cat}(r.x)\ ] \\ s_{c\_base} & : & [\ s_{cat_{Ind}} : \text{cat}_{Ind}(r.x)\ ] \end{bmatrix})$

(40) $[\![\text{cats}]\!] = \lambda r : [x : Stuff].p(\begin{bmatrix} s_{pred} & : & [\ s_{cat} : \text{cat}(r.x)\ ] \\ s_{c\_base} & : & [\ s_{cat_{Ind*}} : {}^{*}\text{cat}_{Ind}(r.x)\ ] \end{bmatrix})$

This structure for the semantics of concrete nouns also encodes our conception of the semantic learning of these nouns as being guided by the establishment, if possible, of a counting base, namely the type labeled s_{c_base} which may then serve as reliable criteria for applying the type labelled s_{pred}. For cases such as *cat*, this is relatively straightforward, because being of the type of a cat individual is a reliable criterion for applying the number neutral predicate *cat* (39).

Reliability: On the assumption that a suitably accurate individuation schema for cats can be found, the upward closure of this predicate will be a highly reliable indicator of when to use the predicate *cat*. That is to say, there are very few instances of things having the requisite properties of being a cat individual or of a cat sum for which the judgement $cat(x)$ would be inappropriate (recall that we assume that the predicate type $cat(x)$ is number neutral). Likewise, there are relatively few mispredications (relatively few *cat* judgements made by competent speakers to refer to entities without the properties of being cat individuals (or sums thereof)). Hence, such a cat_{Ind} predicate would yield high conditional probabilities of the sort in (41) and (42).

(41) $p(r : [s_{cat} : \text{cat}(x)] \mid r : [s_{cat\text{-}ind} : {}^{*}\text{cat}_{Ind}(x)]) = \text{high}$

(42) $p(r : [s_{cat\text{-}ind} : {}^{*}\text{cat}_{Ind}(x)]) \mid r : [s_{cat} : \text{cat}(x)] = \text{high}$

[20] Cooper *et al.* use pTTR.

Individuation: Furthermore, whatever schema yields the highest balance of these probabilities will be a predicate type that is disjoint (it will be the type for individual cats). This means that it will satisfy the entropy minimisation requirement as in (43).[21]

(43) $$-\sum_{v_i \in V} p(v_i|\text{cat}_{Ind}) \times \log p(v_i|\text{cat}_{Ind}) = 0$$

With reliability and individuation acting in unison, the lexical entry in (39) is predicted. It has a disjoint counting base, which constitutes a part of the lexical entry for a count noun. This account further predicts that lexicalisations for the *cat* predicate will be stably count.

However, there are nouns for which it is less straightforward to establish a counting base, because the learning pressures of reliability and individuation conflict. In these cases, we argue, there are two ways of conceptualising a referent, one in which individuation is paramount, in which case the result is a count noun, and another in which reliability is paramount, in which case the result is a mass noun. We connect the source of this conflict between reliability and individuation to the types of context sensitivity that play a key role in the theories of the mass/count distinction in Chierchia (2010), Rothstein (2010) and Landman (2011), which we discussed in Section 3. In the next section, we offer a probM-TTR proposal of how context sensitivity can impact the weighting of individuation and reliability in generating predictions about variation in mass/count lexicalization patterns.

7 MASS/COUNT VARIATION: THE EFFECTS OF CONTEXT SENSITIVITY ON INDIVIDUATION AND RELIABILITY

Contemporary mereological theories of the mass/count distinction converge on the idea that the concepts on which the distinction is based are context-sensitive in one way or another. However, the proposals differ with respect to the degree and nature of the relevant context-sensitivity. In particular, Rothstein (2010) and Landman (2011) emphasise disjointness and overlap, which we argue can be interpreted as generating a conflict between the learning pressures of

[21] We omit the cost C when $C = 0$.

reliability and individuation (or, alternately, as generating competing requirements on the nature of lexical predicate's meaning). Another source of conflict between the learning pressures of reliability and individuation stems from what Chierchia (2010) calls 'vagueness'. However, we suggest this would be more aptly considered a form of context-sensitivity (in a sense we explain below).

Crucially, we argue that the way these conflicts are resolved tracks differences in count/mass lexicalisation of nouns, and hence the specific resolution strategies could serve as a motivation for the lexicalisation of nouns as mass or count, both within a particular language and crosslinguistically.

Let us first consider the notion of disjointness. As in Landman (2011), we assume that there is a grammatical counting function which is sensitive to disjointness. In our account, the counting function is of the type in (44) and applies to the record type in a lexical entry labeled s_{c_base}, which captures the idea that what is counted are the entities of the type in the counting base. Hence, for a counting function f_{count} and probability threshold θ, we propose a type restriction:

(44) $$f_{count,\theta} : (\mathit{RecType} \wedge \mathit{Disj}_{\theta} \rightarrow \mathbb{R})$$

This type restriction means that the counting function is only defined for types that are disjoint relative to some probability threshold and outputs a real number.

The notion of context used in Rothstein (2010) is that of a 'counting context'. Counting contexts are subsets of the domain, i.e., a set of entities that count as atoms, as one, in a particular context, which are then intersected with the root noun denotation of a noun to form a disjoint set, in 'default' cases. As a formal device, this representation has the right motivation and effect for nouns such as *fence*. However, there are two weak points of Rothstein's account that we improve upon. First, both counting contexts and individuation remain at a pretheoretical level in Rothstein's account. Second, Rothstein (2010) effectively subsumes prototypical count nouns like *cat* as a special case of context-sensitive count nouns like *fence*, for which there just happens to be no variation in what counts as one across contexts. Intuitively, what is one cat is stable across all contexts, but what is one fence varies with context. Our formalism improves on Rothstein's idea of counting contexts by modelling them as contexts in which some in-

dividuation schema is used, where the different individuation schemas that are licensed in a given context are grounded in our account of semantic learning. As a result, we can explain and better motivate why there is only one licensed individuation schema for nouns such as *cat*, but multiple such schemas for nouns such as *fence*.

7.1 *Mass/count variation in collective artifacts*

The term 'collective artifacts' here refers to nouns which denote collections of entities like, for example, furniture and kitchenware. The main point of this section is to show that the way in which nouns for collective artifacts denote in context allows for two distinct ways of forming concepts. One is compatible with counting, and the other is not. Therefore, we should expect to find variation in mass/count encoding in this class. A good example is the mass noun *furniture* in English and the count noun *huonekalu-*$t_{+C,PL}$ ('furniture') in Finnish. What ultimately motivates whether a noun, say, *furniture* is mass in English, and not count, may well be wholly conventional (guided by, for example, etymological factors). Here, we focus on the issue of variation in mass/count encoding, which is separate from the question why, say, *furniture* is mass in English, which we leave aside here.

When it comes to collective artifacts, we propose that it is reasonable to assume that what counts as one item of furniture will largely be derived from the function of the relevant item and to a lesser degree from its perceptual qualities. A vanity, formed of a mirror and a table, has a joint function *qua* item of furniture, so plausibly counts as a single item. However, the mirror and the dressing table each have its own function and each can stand and be used as an individual item of furniture in its own right. Likewise, we see similar patterns crosslinguistically, even when collective artifacts are lexicalized as count nouns. For example, what counts as one for the count noun *huonekalu* ('furniture', Finnish) varies with context in the same way as for the English *furniture*.

The kind of contextual variation that we observe with respect to what counts as one for *furniture* or for *huonekalu* presents a learning challenge to our basic picture of learning an individuation schema for a given noun predicate. Recall that individuation schemas are modelled as the type for which a quantitative function outputs 1. They apply to functionally – and also perceptually – characterised situation

types (of the qualitative criterion type) and are used to individuate the entities within a situation (record) relative to a predicate. A learning challenge arises because the same entity, the vanity, plausibly counts as one item of furniture in one schema and as two items of furniture in another. But that means that no single function should be able to output both of these results for a situation (record) containing a vanity. Put simply, witnessing the same kind of item being treated as one thing in one situation, and two (or more) things in other situations, is evidence that there is more than one felicitous individuation schema for *furniture*.

In other words, we have a case such as the one outlined in Section 5, in which a learner has evidence for multiple context-specific individuation schemas. This generates a conflict between the two learning pressures of individuation and reliability. Take *furniture*, for instance. Opting for a single individuation schema would keep stable what is individuated as one item of furniture in every situation, but it is not a reliable way of individuating, since one single schema will wrongly individuate in some situations. For example, a schema that individuates a table and a mirror as two entities will be incorrect in circumstances where they should count as one item of furniture, a vanity.

Faced with this challenge, a learner has two strategies available: namely, either to learn to apply a different schema depending on the situation, or to form a single complex join type based on all licensed schemas. For example, if P_{Ind_i} is a licensed individuation schema, a more generally applicable type would be the join type formed of all such schemas. This is shown in (45).

(45) $$P_{Ind_{join}} = P_{Ind_1} \vee P_{Ind_2} \vee \ldots \vee P_{Ind_n}$$

The kind of context sensitivity that affects what counts as one for collective artifacts gives rise to two alternative ways of encoding the semantics of a *furniture*-like noun. We now detail these with respect to the formal characterisations of reliability and individuation.

First, one can opt for a complex join schema, and so have a reliable indicator of when to apply the noun predicate in most (possibly all) contexts. This is indicated by the high conditional probabilities in (46) and (47). Given an individuating predicate that is the join of all contextually specific ones, the upward closure of this predicate will closely track how one should apply the predicate *furn*.

(46) $p(r : [s_{furn} : \text{furn}(x)] | r : [s_{furn\text{-}ind} : {}^{*}\text{furn}_{Ind_{join}}(x)]) = \text{high}$

(47) $p(r : [s_{furn\text{-}ind} : {}^{*}\text{furn}_{Ind_{join}}(x)]) | r : [s_{furn} : \text{furn}(x)] = \text{high}$

Obviously, an individuating predicate that is the join of all contextually specific ones cannot individuate or yield a disjoint type which can support determinate counting results in a specific situation, due to overlaps of the individuals picked by different individuation schemas of that join individuating predicate. This captures Landman's (2011) intuition that overlapping entities can "simultaneously in the same context" count as one. As shown in (48), the fact that there are a number of different ways to resolve overlap in respect to what counts as one item of furniture leads to a comparatively high level of entropy.

(48) $-\sum_{v_i \in V} p(v_i|\text{furn}_{Ind_{join}}) \times \log p(v_i|\text{furn}_{Ind_{join}}) = \text{high}$

Second, one can make the selection of one's individuation schema context sensitive, i.e. apply individuation schemas that may vary from situation to situation. However, as shown in (49) and (50), this has a negative effect on reliability. Although the probability of *furn* is high given the upward closure of any specific individuating predicate $\text{furn}_{Ind_{c_i}}$, the inverse conditional probability is lower than in the complex join individuation schema, because most particular schemas will exclude those bits of furniture that are parts of that which count as one under a different individuation schema. For example, if an agent has an individuation schema that classifies a vanity (table and mirror) as one item of furniture, then this will exclude its parts from counting as one. However, this means that the probability of applying this particular schema given a *furn* judgement will be lower than for the context general join-type case, since the context specific schema (for vanity) will not be reliable for situations in which the table and mirror should count as two items of furniture. This lowers the conditional probability in (50).

(49) $p(r : [s_{furn} : \text{furn}(x)] \mid r : [s_{furn\text{-}ind} : {}^{*}\text{furn}_{Ind_{c_i}}(x)]) = \text{high}$

(50) $p(r : [s_{furn\text{-}ind} : {}^{*}\text{furn}_{Ind_{c_i}}(x)]) \mid r : [s_{furn} : \text{furn}(x)] = \text{lowish}$

However, the entropy value is arguably lower than in the single join schema case. That is, in the case of a single join schema, every specific schema will yield an entropy value of 0, since each one is a disjoint predicate, but since there are many such schemas, the task of determining the right one in context incurs a cost C (see case (iii) in Section 5.2). This is shown in (51). Provided that the cost value is lower than the entropy value for the join type in (48), the specific case will fare better at minimising entropy (maximising individuation).

(51) $$-\Big(\sum_{v_j \in V}\sum_{c_i \in C} p(v_j|\text{furn}_{Ind_{c_i}}) \times \log p(v_j|\text{furn}_{Ind_{c_i}})\Big) + C = C$$

In summary, if one tries to maximise reliability, the context general join-type individuation schema wins out over adopting a number of context specific ones. However, if one minimises entropy, selecting a context specific schema wins out over the context general join-type individuation schema.

This creates a tension. One simple outcome is merely to prioritise either reliability (and thereby encode a context general join-type individuation schema), or prioritise individuation (and thereby encode a context specific schema). Choosing the former strategy results in a counting base that is not disjoint. This means, as per our Landman-inspired account of the mass-count distinction, that the resulting lexical entry is one for a mass noun. Choosing the latter strategy results in a counting base that is disjoint. This means that the resulting lexical entry is one for a count noun, as in the case of the Finnish singular count noun *huonekalu*. The results of these two strategies are given in (52) and (53).

(52) $[\![\text{furniture}]\!]^{c_i} = \lambda r : [x : Stuff].p(\begin{bmatrix} s_{pred} & :[s_{furn} : \text{furn}(r.x)] \\ s_{c\_base} & :[s_{furn_{Ind}} : \text{furn}_{Ind_{join}}(r.x)] \end{bmatrix})$

(53) $[\![\text{huonekalu}]\!]^{c_i} = \lambda r : [x : Stuff].p(\begin{bmatrix} s_{pred} & :[s_{furn} : \text{furn}(r.x)] \\ s_{c\_base} & :[s_{furn_{Ind}} : \text{furn}_{Ind_{c_i}}(r.x)] \end{bmatrix})$

The lexical entry for *furniture* in (52) has a generalized schema as the base, but this will not yield a noun suitable for counting, since $furn_{Ind_{join}}$ will not be a disjoint type (both the vanity and the mirror and table that comprise the vanity will be of this type). However, the lexical entry for *huonekalu* in (53) will yield a count noun, since every single individuation schema $furn_{Ind_{c_i}}$ will be disjoint.

7.2 *Mass/count variation in non-bounded objects*

The explanation we have just used to motivate the variation in the mass/count encoding of collective artifacts across different languages (e.g., *furniture* (mass) versus *huonekalu* (count) 'furniture', Finnish) can also be applied to motivate the intralinguistic variation exhibited by pairs such as *fence* (count) and *fencing* (mass), which constitute a well-defined semantic subclass we dub here 'non-bounded objects.' The entry for *fence* is given in (54) and the entry for *fencing* is given in (55).

(54) $[\![\text{fence}]\!]^{c_i} = \lambda r : [x : Stuff].p(\begin{bmatrix} s_{pred} & :[s_{fence} : \text{fence}(r.x)] \\ s_{c\_base} & :[s_{fence_{Ind}} : \text{fence}_{Ind_{c_i}}(r.x)] \end{bmatrix})$

(55) $[\![\text{fencing}]\!]^{c_i} = \lambda r : [x : Stuff].p(\begin{bmatrix} s_{pred} & :[s_{fence} : \text{fence}(r.x)] \\ s_{c\_base} & :[s_{fence_{Ind}} : \text{fence}_{Ind_{join}}(r.x)] \end{bmatrix})$

Across situations, *fence* is interpreted relative to a context specific individuation schema $fence_{Ind_{c_i}}$ where c_i is selected depending on the situation. From this it follows that individuation is maximised, but not reliability. In a given context, $fence_{Ind_{c_i}}$ is disjoint, and so defined for counting, which leads to the desirable prediction that the exact result of counting the same stretch of fencing may result in different answers across situations.

In contrast, *fencing* applies the same individuation schema across situations, namely, one that is defined in terms of a join individuation schema type $\text{fence}_{Ind_{join}}$, which consists of a number of individuation schemas. But this means that it is not disjoint. Take, for example, Rothstein's square field example, where the sum of four fence sides is of type $\text{fence}_{Ind_{join}}$, but so too are the four fence-sides taken individually, whereby the former overlaps with the latter. But this means that the question 'How many fences are there?' has two different possible answers: 'one' or 'four'. In this sense, non-disjoint types are not countable, and so *fencing* is mass (Landman 2011).

7.3 *Mass/count variation in granulars*

In Section 5, we outlined how, for small quantities of rice, an agent may be left with a high degree of uncertainty whether or not to judge it as satisfying the predicate *rice*. If, as observed by Chierchia (2010), quantity of grains is a major factor affecting the applicability of a predicate like *rice* to a collection of entities, then we should also expect

the amount of uncertainty along an axis of quantity to be relatively smooth (graded).

For nouns that are context sensitive in this way, a learning challenge arises. We argued, in Section 5, that semantic learning for concrete nouns is largely governed by two pressures. One is to ascertain a consistent and reliable criterion for the application of a noun; the other is to establish what, if anything, the individuable units in the noun's denotation are, which is a prerequisite for counting. In simple cases, identifying the individuable units in a noun's denotation is to identify what the minimal entities are to license applying a predicate. This is the case for prototypical count nouns such as *cat*. If one has either a single cat, or a sum of single cats, one can correctly use the noun *cat* of them.

It is precisely the context sensitivity of granular nouns, such as *rice* and *lentils*, which provides a compelling argument in support of reliability and individuation as two pressures on semantic learning implicated in the acquisition of the mass/count distinction, because learning of granulars pushes reliability and individuation in opposite directions. The denotations of granulars contain perceptually individuable units (e.g., single rice grains, or single lentils). However, having either a single grain of rice or a single lentil does not always license that the noun *rice* or *lentil(s)* can be felicitously applied to them. This is because there are many contexts in which single grains of rice or single lentils, or even small quantities of rice grains or lentils are insufficient in quantity to count as *rice* or *lentil(s)*. Hence, individuating in terms of grains loses reliability.

One way to increase reliability is to make the quantitative function one that identifies aggregates of entities with the requisite properties such as colour, shape, etc., especially if the sizes of these aggregates are those most predictive of the appropriate conditions for using the relevant predicate. The most diagnostic sizes of aggregates will be those that are frequently encountered and have a high correlation with correct application of the relevant predicate. For example, if, say, spoonfuls, bowlfuls and packets of lentils are the most frequently encountered aggregates of lentils and almost always get judged to be lentils by competent speakers, then if an aggregate of lentils is a spoonful, a bowlful or a packet of lentils in size, then one has very good reason to apply *lentils* to that aggregate. Furthermore, if someone has

used the term *lentils* one has good reason to expect it to refer to such a frequently encountered aggregate size. Doing this would satisfy the pressure to establish a reliable criterion; however, it would do so at the expense of satisfying the individuation pressure.

We label the individuating predicate that is based on such single-grain properties $\textit{lentil}_{\textit{Ind\_gr}}$. This schema will not provide good results for one aspect of reliability such as the conditional probability in (56), because small quantities of lentils are not good predictors for when to make a *lentil* judgement. In many contexts, larger quantities are required to count as *lentils*. With respect to the inverse conditional probability (57), it fares better, since the upward closure of the predicate which picks out single lentils will match the conditions for applying *lentil* in all but the cases where sub-grain parts of lentils count as *lentils*.

(56) $p(r : [s_{\text{lentil}} : \text{lentil}(x)] \,|\, r : \left[s_{\textit{lentil-ind}} : {}^{*}\text{lentil}_{\textit{Ind\_gr}}(x)\right]) = \text{lowish}$

(57) $p(r : \left[s_{\textit{lentil-ind}} : {}^{*}\text{lentil}_{\textit{Ind\_gr}}(x)\right]) | r : [s_{\textit{lentil}} : \text{lentil}(x)] = \text{highish}$

The individuating predicate $\text{lentil}_{\textit{Ind\_gr}}$ maximises individuation, however. It applies to single lentils, which are disjoint. This means that there is only one variant, the predicate itself, hence entropy is 0.

(58) $$-\sum_{v_i \in V} p(v_i|\text{lentil}_{\textit{Ind\_gr}}) \times \log p(v_i|\text{lentil}_{\textit{Ind\_gr}}) = 0$$

So, adopting $\text{lentil}_{\textit{Ind\_gr}}$ maximises individuation, but does so at the expense of reliability.

The alternative strategy is to choose a schema that is more reliable, namely in terms of aggregates (which were formally characterised in Sections 4.2-4.2). Instead of individuating only in terms of single grains, one could instead use a schema that identifies the sizes of aggregates of grains that are most diagnostic of when to apply the predicate *lentil* (a join type of the most diagnostic lentil aggregate sizes). Call this $\text{lentil}_{\textit{join\_agg}}$. As formalised in (60), the $\text{lentil}_{\textit{join\_agg}}$ predicate may also miss out on some cases where very small collections of lentils count as lentils (which indicates that our representation is missing some element of further context-sensitivity for granulars). However, it will do better with respect to predicting when to apply

lentil, given the schema as shown in (59). This is because the cases where there are insufficient amounts of lentils to make a *lentil* judgement will also be cases where it is insufficient to make a $lentil_{join\_agg}$ judgement.

(59) $p(r : [s_{\text{lentil}} : \text{lentil}(x)] | r : \left[s_{lentil\text{-}ind} : {}^{*}\text{lentil}_{join\_agg}(x)\right]) = \text{high}$

(60) $p(r : \left[s_{lentil\text{-}ind} : {}^{*}\text{lentil}_{join\_agg}(x)\right]) | r : [s_{lentil} : \text{lentil}(x)] = \text{highish}$

However, the aggregating strategy fares badly with respect to individuation. A join individuating predicate that identifies aggregates of some minimum sizes is not disjoint because e.g. spoonful sized aggregates form proper parts of e.g. bowlful sized aggregates. Such join types have multiple maximally disjoint variants. Therefore, each context specific schema yields a higher entropy value than in the $lentil_{Ind\_gr}$ case:

(61) $-\sum_{v_i \in V} p(v_i | \text{lentil}_{join\_agg}) \times \log p(v_i | \text{lentil}_{join\_agg}) = \text{high}$

Neither of the two alternatives for individuation schemas can satisfy both pressures of individuation and reliability. $Lentil_{Ind\_gr}$ minimises entropy, thereby maximising individuation, but does not maximise reliability. $Lentil_{join\_agg}$ maximises reliability, but does not maximise individuation. As in the *furniture* and *fence* cases, this tension can result in two kinds of lexical entries involving the same number-neutral type *lentil*. Equation (62) uses $lentil_{Ind\_gr}$, has a disjoint counting base, and so is the entry for a count noun such as the English *lentil*. Equation (63) uses $lentil_{join\_agg}$, does not have a disjoint counting base, and so is the entry for a mass noun such as the Czech *čočka* ('lentil').

(62) $[\![\text{lentil}]\!] = \lambda r : [x : Stuff].p(\begin{bmatrix} s_{pred} & : [s_{lentil} : \text{lentil}(r.x)] \\ s_{c\_base} & : [s_{lentil\text{-}ind} : \text{lentil}_{Ind\_gr}(r.x)] \end{bmatrix})$

(63) $[\![\text{čočka}]\!] = \lambda r : [x : Stuff].p(\begin{bmatrix} s_{pred} & : [s_{lentil} : \text{lentil}(r.x)] \\ s_{c\_base} & : [s_{lentil\text{-}ind} : lentil_{join\_agg}(r.x)] \end{bmatrix})$

7.4 *Mass/count stability in substances, liquids and gasses*

When it comes to mass nouns like *mud*, *blood*, and *air*, similarly to granulars, the quantity of a substance has an impact on the applica-

bility of the noun in a way that varies with context. For example, a speck of mud on one's shoes could count as *mud* in a scientific clean room context, but not in a context where one is entering a garden shed. The principal difference between substances and granulars is in the perceptual properties of their references. Whereas in the granular case, there are clearly individuable entities which could be judged to be of some P_{Ind} type (e.g. $lentil_{Ind\_gr}$), substances lack any such thing.

This means that, of the strategies so far considered, there is only one type individuation schema one might try to use, namely an amassment, the substance noun counterpart to an aggregating schema, that individuates in terms of a join of amassments of stuff with mud properties which are, jointly, the best indicators of when to apply mud. In a similar vein to the granulars case we have considered, the amassment schemas would fare well with respect to reliability ((64) and (65)).

(64) $p(r : [s_{mud} : \text{mud}(x)] \,|\, r : \left[s_{mud\text{-}ind} : {}^{*}\text{mud}_{join\_amass}(x)\right]) = \text{high}$

(65) $p(r : \left[s_{mud\text{-}ind} : {}^{*}\text{mud}_{join\_amass}(x)\right]) | r : [s_{mud} : \text{mud}(x)] = \text{highish}$

Individuation is militated against with such a schema, however, since there is a high number of admissible (disjoint) variants and presumably none of them will be particularly weighted over the others:

(66) $-\sum_{v_i \in V} p(v_i|\text{mud}_{join\_amass}) \times \log p(v_i|\text{mud}_{join\_amass}) = \text{high}$

On the face of it, it may look as though this strategy is the only viable one. It maximises reliability, but does so at the expense of individuation. This leads us to expect most languages to develop a lexical entry for mud with an overlapping counting base, thus lexicalized with a mass noun. This is the case in English as in (67).

(67) $[\![\text{mud}]\!]^{c_i} = \lambda r{:}[x : Stuff].p(\begin{bmatrix} s_{pred} & :[s_{mud} : \text{mud}(r.x)] \\ s_{c\_base} & :[s_{mud\text{-}ind} : \text{mud}_{join\_amass}(r.x)] \end{bmatrix})$

Our account, therefore predicts relative stability in the mass lexicalization of substance, liquid and gas denoting nouns crosslinguistically.

However, we might ask if there is any way one could boost individuation, even for noun concepts which denote substances such as mud and blood. A clue for what kind of strategy might do this comes from languages like Yudja as reported in Lima (2014, 2016,

a.o.). In Yudja, different sizes/portions of substances such as blood can be directly counted provided that they are contextually disjoint. Lima's (2014) analysis of Yudja relies on mereotopological concepts from Grimm (2012), and specifically on the concept of Maximal Self Connectedness (MSC) which is the property of countable entities. Informally, "an entity is self-connected means that whenever we partition this entity into two parts, these two parts are connected to each other." (Lima 2014, p. 140)

We formalise this in terms of *bounded amassments* (Section 4.2), namely, identifying, at a perceptual level, distinct bounded regions formed from stuff with the requisite properties. For example, an individuation schema such as $blood_{bounded}$ applies to stuff with blood properties that also forms a bounded region; namely, a disjoint part of space containing blood. As such, the $blood_{bounded}$ predicate will individuate as there will not be multiple variants (e.g. a drop of blood will not be formed of disjoint bounded drops of blood). Namely, we have zero entropy as shown in (68).

(68) $$-(\sum_{v_j \in V} p(v_j|\text{blood}_{bounded}) \times \log p(v_j|\text{blood}_{bounded})) = 0$$

However, although being a bounded region of e.g. blood may be a reliable indicator for applying *blood* (69), being blood may not be a reliable indicator for being a bounded region of blood or a sum thereof, since blood (and other substances) do not always come in bounded portions. This translates into a lowering of the conditional probability in (70).

(69) $p(r : [s_{blood} : \text{blood}(x)] \,|\, r : [s_{blood\text{-}ind} : {}^{*}\text{blood}_{bounded}(x)]) = \text{high}$

(70) $p(r : [s_{blood\text{-}ind} : {}^{*}\text{blood}_{bounded}(x)]) | r : [s_{blood} : \text{blood}(x)] = \text{not high}$

Yudja does not have a rich lexicalized measurement system (aside from loan words (Lima p.c.)). The result is that the only way to quantify stuff (be it intuitively individuated or not) is by direct counting.[22] Languages with such relatively rare characteristics could therefore be ones which adopt a strategy of individuating any bounded, disjoint amounts of stuff with the relevant perceptual (or functional) properties (e.g. colour, consistency, etc.). This strategy, applied across the

[22] We are grateful to S. Rothstein for raising the possibility of this connection.

board to substance denoting nouns, could result in there being no genuine mass nouns in such languages, as is reported to be the case in Yudja (Lima 2014).

Substance denoting noun entries would, therefore, look like that for *apeta* ('blood', Yudja), as in (71), and would be count.

(71) $[\![\text{apeta}]\!]^{c_i} = \lambda r\!:\![x : \mathit{Stuff}].p(\begin{bmatrix} s_{pred} & :[s_{blood}:\text{blood}(r.x)] \\ s_{c\_base} & :[s_{blood\text{-}ind}:\text{blood}_{bounded}(r.x)] \end{bmatrix})$

8 CONCLUSION AND SUMMARY

The formalism we have developed as a mereological enrichment of prob-TTR can be justified independently of issues surrounding the mass/count distinction. With respect to probabilistic semantics, there is increasing recognition that semantic, pragmatic, and knowledge representations, in order to be cognitively plausible, should be able to reflect gradience in judgements, and be consistent with a tractable account of semantic learning. Mereology is widely used in semantics for modelling plurality, tense, and aspect as well as the mass/count distinction. Using these formal tools, we tried to flesh out the intuition of Krifka (1989) that applying nouns involves both qualitative and quantitative criteria. We sketched how some properties, such as the size and boundedness of an aggregate of rice grains, could be modelled in a manner inspired by work on linking TTR representations to perceptual inputs, and how spatial perception is one factor in guiding the quantitative process of individuating entities. We have also shown how probM-TTR naturally accommodates cutting edge ideas on the semantics of the mass/count distinction, and, significantly, we are able to offer a unified explanation of why some classes of nouns display a wide amount of cross and intralinguistic mass/count variation while others do not; namely, as the result of balancing the pressures of individuation and reliability in semantic learning. Sometimes these pressures align (*prototypical objects*), sometimes they do not (*collective artifacts*, *non-bounded objects*, and *granulars*) and sometimes individuation cannot easily be prioritised at all (*substances*).

This yields four semantic classes of nouns which pattern differently with respect to the distribution their nouns have over the two grammatical properties MASS and COUNT. These are summarised in Table 1 and elaborated on below.

Noun class	Properties of counting base	Mass/count variation
Prototypical objects	Disjoint, single individuation schema across contexts	Rare
Collective artifacts & non-bounded objects	Multiple, disjoint, context specific schemas OR a single multiplicity of overlapping schemas	Common
Granulars	Disjoint schema picking out single grains, OR overlapping schema amassing aggregates of grains (e.g., spoonfuls, bowlfuls)	Common
Substances	Usually, an overlapping schema that groups frequently encountered amassments of stuff. Sometimes a schema that identifies contextually provided bounded amassments	Rare

Table 1: Summary of noun classes and their properties

Prototypical objects: The types that pick out the individuable entities in the denotations of prototypical object nouns are also highly consistent indicators of when to apply the nouns. The pressures of individuation and reliability work in the same direction, i.e., they converge on the count encoding. We, therefore, have no reason to expect much variation from the count encoding, cross- and intralinguistically.

Collective artifacts and non-bounded objects: The context-sensitivity of nouns in these classes affects the reliability with which any single individual predicate type applies. For example, across contexts, a sum of fence pieces can count as one fence, or two fences; and a pestle and mortar can count as one item of kitchenware or two items of kitchenware. This means that any particular individuation schema will unreliably determine the extension. To prioritise individuation, multiple individuation schemas, each indexed to a context, can be used. This yields count nouns such as *fence*, and *Küchengeräte* ('kitchenware' German). Alternatively, to prioritise reliability, all individuation schemas can be merged together. This yields a non-disjoint schema, and so motivates the encoding of nouns, such as *fencing* and *kitchenware*, as mass nouns.

Granulars: Context-sensitivity with granular noun denotations has an effect on what quantities of the relevant stuff are needed to qualify for that stuff to fall under a given noun denotation. Granular nouns tend to be easily perceptually individuable (in terms of salient indi-

vidual grains), but given that single grains are not always enough to qualify as falling under a given noun denotation across all contexts, the type for single grains, that prioritises individuation, is inconsistent as a basis for applying a noun. Prioritising individuation yields a count noun encoding, which is commonly presupposed by pluralisation, e.g. *lentils*, *oats*, *kaurahiutale-et* ('oatmeal', Finnish). On the other hand, prioritising reliability yields a non-disjoint individuation schema, and so leads to a mass noun encoding, as in *oatmeal*, *kaura* ('oats', Finnish), *čočka* ('lentils', Czech).

Substances: Context-sensitivity also has an effect on amounts of quantities (e.g., of substances, liquids, and gases) reaching a certain threshold to qualify as falling under a given noun (e.g., *mud*, *blood*, and *air*). However, the perceptual qualities of the denotations of these nouns does not easily enable the prioritisation of individuation that could be achieved for count granular nouns. If individuation cannot easily be prioritised, then we should expect to find more cases where reliability will be. Therefore, we expect a heavy tendency towards mass encoding for these nouns.

REFERENCES

J. L. Austin (1950/1979), Truth, in J. O. Urmson and G. J. Warnock, editors, *Philosophical Papers, Third Edition*, pp. 117–133, Oxford University Press, Oxford, Originally in: *Symposium: Truth, Proceedings of the Aristotelian Society*, Vol. 24 (1950).

Jon Barwise and John Etchemendy (1987), *The Liar: An Essay on Truth and Circularity*, Oxford University Press USA.

Gennaro Chierchia (2010), Mass Nouns, Vagueness and Semantic Variation, *Synthese*, 174:99–149.

Robin Cooper (2012), Type Theory and Semantics in Flux, in R. Kempson, T. Fernando, and N. Asher, editors, *Philosophy of Linguistics, Handbook of the Philosophy of Science*, pp. 271–323, Elsevier.

Robin Cooper, Simon Dobnik, Shalom Lappin, and Staffan Larsson (2014), A Probabilistic Rich Type Theory for Semantic Interpretation, *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics*.

Robin Cooper, Simon Dobnik, Staffan Larsson, and Shalom Lappin (2015), Probabilistic Type Theory and Natural Language Semantics, *LILT*, 10(4).

Simon DOBNIK, Robin COOPER, and Staffan LARSSON (2012), Modelling language, action, and perception in type theory with records, in *Constraint Solving and Language Processing*, pp. 70–91, Springer Berlin Heidelberg.

Jan van EIJCK and Shalom LAPPIN (2012), Probabilistic Semantics for Natural Language, in P. CHRISTOFF, N. GIERASIMSZUK, A. MARCOCI, and S. SMETS, editors, *Logic and Interactive Rationality Volume 2*, pp. 17–35, University of Amsterdam: ILLC.

Hana FILIP and Peter SUTTON (2017), Singular Count NPs in Measure Constructions, manuscript, to be presented at SALT 2017.

Charles J. FILLMORE (1975), An Alternative to Checklist Theories of Meaning, *Proceedings of the First Annual Meeting of the Berkeley Linguistics Society*, 1:123–131.

Charles J. FILLMORE (1976), Frame semantics and the nature of language, *Annals of the New York Academy of Sciences*, 280(1):20–32, ISSN 1749-6632.

Scott GRIMM (2012), *Number and Individuation*, PhD Dissertation, Stanford University.

Daniel C. HYDE and Elizabeth S. SPELKE (2011), Neural signatures of number processing in human infants: evidence for two core systems underlying numerical cognition, *Developmental Science*, 14(2):360–371, ISSN 1467-7687, doi:10.1111/j.1467-7687.2010.00987.x, `http://dx.doi.org/10.1111/j.1467-7687.2010.00987.x`.

Ray JACKENDOFF (1991), Parts and Boundaries, *Cognition*, 41:9–45.

A. KOLMOGOROV (1950), *Foundations of probability*, Chelsea Publishing, New York.

Manfred KRIFKA (1989), Nominal Reference, Temporal Constitution and Quantification in Event Semantics, in Renate Bartsch and J. F. A. K. van Benthem and P. van Emde BOAS, editor, *Semantics and Contextual Expression*, pp. 75–115, Foris Publications.

Fred LANDMAN (2011), Count Nouns – Mass Nouns – Neat Nouns – Mess Nouns, *The Baltic International Yearbook of Cognition*, 6:1–67.

Fred LANDMAN (2016), Iceberg Semantics for Count Nouns and Mass Nouns: The evidence from portions, *The Baltic International Yearbook of Cognition Logic and Communication*, 11:1–48.

Daniel LASSITER (2016), Must, knowledge, and (in)directness, *Natural Language Semantics*, 24(2):117–163, ISSN 1572-865X, doi:10.1007/s11050-016-9121-8, `http://dx.doi.org/10.1007/s11050-016-9121-8`.

Suzi LIMA (2014), All notional mass nouns are count nouns in Yudja, *Proceedings of SALT*, 24:534–554.

Suzi LIMA (2016), Container constructions in Yudja: locatives, individuation and measure, *The Baltic International Yearbook of Cognition Logic and Communication*, 11:1–40.

Godehard LINK (1983), The Logical Analysis of Plurals and Mass Terms: A Lattice-Theoretic Approach, in P. PORTNER and B. H. PARTEE, editors, *Formal Semantics - the Essential Readings*, pp. 127–147, Blackwell.

S. PIANTADOSI, H. TILY, and E. GIBSON (2011), The communicative function of ambiguity in language, *PNAS*, 108(9):3526–3529.

Roberta PIRES DE OLIVEIRA and Susan ROTHSTEIN (2011), Bare singular noun phrases are mass in Brazilian Portugese, *Lingua*, 121:2153–2175.

James PUSTEJOVSKY (1995), *The Generative Lexicon*, MIT Press.

Susan ROTHSTEIN (2010), Counting and the Mass/Count Distinction, *Journal of Semantics*, 27(3):343–397, doi:10.1093/jos/ffq007.

Peter R. SUTTON and Hana FILIP (2016a), Mass/Count Variation, a Mereological, Two-Dimensional Semantics, *The Baltic International Yearbook of Cognition Logic and Communication*, 11:1–45.

Peter R. SUTTON and Hana FILIP (2016b), A probabilistic, mereological account of the mass/count distinction, *LNCS 10148, Proceedings of TbiLLC 2015*, p. To appear.

Leonard TALMY (2000), *Toward a Cognitive Semantics – Vol. 1*, The MIT Press.

Sandro ZUCCHI and Michael WHITE (1996), Twigs, Sequences and the Temporal Constitution of Predicates, in Teresa GALLOWAY and Justin SPENCE, editors, *Proceedings of SALT 6*, pp. 223–270, Linguistic Society of America.

Sandro ZUCCHI and Michael WHITE (2001), Twigs, Sequences and the Temporal Constitution of Predicates, *Linguistics and Philosophy*, 24(2):223–270.

Quantification in frame semantics with binders and nominals of hybrid logic*

Laura Kallmeyer[1], *Rainer Osswald*[1], *and Sylvain Pogodalla*[1,2]
[1] Heinrich Heine Universität, Düsseldorf, Germany
[2] INRIA, Villers-lès-Nancy, France
Université de Lorraine, LORIA, Vandœuvre-lès-Nancy, France
CNRS, LORIA, Vandœuvre-lès-Nancy, France

ABSTRACT

This paper aims to integrate logical operators into frame-based semantics. Frames are semantic graphs that allow lexical meaning to be captured in a fine-grained way but that do not come with a natural way to integrate logical operators such as quantifiers. The approach we propose stems from the observation that modal logic is a powerful tool for describing relational structures, including frames. We use its hybrid logic extension in order to incorporate quantification and thereby allow for inference and reasoning. We integrate our approach into a type theoretic compositional semantics, formulated within Abstract Categorial Grammars. We also show how the key ingredients of hybrid logic, nominals and binders, can be used to model semantic coercion, such as the one induced by the *begin* predicate. In order to illustrate the effectiveness of the proposed syntax-semantics interface, all the examples can be run and tested with the Abstract Categorial Grammar development toolkit.

Keywords: Frame semantics, quantification, hybrid logic, Abstract Categorial Grammar

1 FRAMES AND LEXICAL SEMANTICS

Frames emerged as a representation format of conceptual and lexical knowledge (Fillmore 1977; Barsalou 1992; Löbner 2014a). They

*This work was supported by the INRIA sabbatical program and by the CRC 991 "The Structure of Representations in Language, Cognition, and Science" funded by the German Research Foundation (DFG).

are commonly presented as semantic graphs with labelled nodes and edges, such as the one in Figure 1, where nodes correspond to entities (individuals, events, …) and edges correspond to (functional or non-functional) relations between these entities. In Figure 1 all relations except *part-of* are meant to be functional.

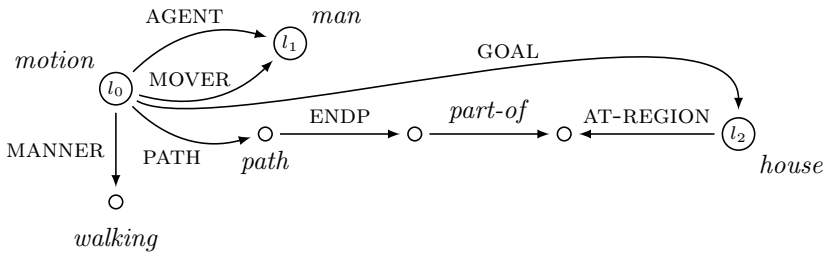


Figure 1: Frame for the meaning of *the man walked to the house* (adapted from Kallmeyer and Osswald 2013)

Structuring the knowledge as frames offers a fine-grained and systematic decomposition of meaning. This conception of frames is however not to be confused with the somewhat simpler FrameNet frames, although the former can help to capture the structural relations of the latter (see Osswald and Van Valin 2014).

Frames can be formalized as extended typed feature structures (Petersen 2007; Kallmeyer and Osswald 2013) and specified as models of a suitable logical language, the *labelled attribute-value description (LAVD) language*. Such a language allows for the composition of lexical frames on the sentential level by means of an explicit syntax-semantics interface (Kallmeyer and Osswald 2013).

1.1 *Logical representation of feature structures*

The syntax-semantics interface of (Kallmeyer and Osswald 2013) relies on a formal representation of semantic frames as *base-labelled feature structure with types and relations*. This definition extends the standard definition of feature structures in two respects. First, in addition to features, proper relations between nodes can be expressed. Moreover, it is not required that every node be accessible from a single root node via a feature path; instead, it is required that every node be accessible from one of the base-labelled nodes. Semantic frames defined in this way can be seen as finite first-order structures which conform to a signature consisting of a set $\mathsf{Label} \cup \mathsf{Type}$ of unary relation symbols and

a set Feat∪Rel of binary relation symbols subject to the constraints that the members of Label denote singletons, the members of Feat denote *functional* relations, and that the above accessibility condition holds. In the example frame of Figure 1, symbols inside nodes (l_0, l_1, …) indicate base labels, symbols attached to nodes (*man*, *motion*, …) belong to Type, members of Feat are marked by small caps (AGENT, ENDP, …), and *part-of* is the only member of Rel occurring in this frame.

But the logical framework of (Kallmeyer and Osswald 2013) does not provide means for explicit quantification. As a consequence, the referential entities of the domain of discourse are implicitly treated as definite, which is reflected by the *naming* of nodes l_0, l_1, etc.

Such relational structures can also easily be turned into Kripke structures. Thus, semantic frames, or feature structures, provide a natural application domain for modal languages and, in particular, for hybrid extensions because of the need to cope with node labels and feature path re-entrancies (Blackburn 1993).

1.2 *Semantic frames and hybrid logic*

As Blackburn (1993) points out, attribute-value structures can be described using the logical language of *Hybrid Logic* (HL, cf. Areces and ten Cate 2007), an extension of the language of modal logic, well-suited to the description of graph structures like the one of Figure 1. HL introduces *nominals*, i.e., node names, that allow the logical formulas to refer to specific nodes of the graph. The nominal l_0 for instance refers to the *motion* node in Figure 1. It is then possible, for example, to specify that the AGENT and the MOVER edges from the node l_0 should meet on the same node in Figure 1. This additional expressiveness of HL over modal logic allows one to express node sharing in attribute value structures (Blackburn 1993). HL is an established logical formalism which has been extensively studied, in particular with respect to the addition of *variables* for nodes, and the associated *binders*, that can appear in the logical formulas. Its relation to attribute-value structures and its expressiveness make it a natural candidate to relate quantified expressions and frame semantics.

With respect to Kallmeyer and Osswald (2013), the approach we propose here does not consider frames as “genuine semantic representations”. The one-to-one equivalence between the logical formulas of the LAVD language of Kallmeyer and Osswald (2013) and the frames

as graph (or relational) structures relies on the existence of minimal models for such formulas. While HL with nominals but without variables nor binders is very close to the LAVD language, it is not obvious what the notion of minimal model of the latter becomes when using quantification. Thus, we have a more traditional view where the sense of an expression is a hybrid logical formula and its reference is computed against models. The latter are the frames we wish to consider. But, contrary to what happens with minimal models, they are then not fully specified by the logical formulas which serve as frame descriptions.

This move from encodings to models is closely related to the one from feature value matrices, as *directly encoding a graph*, to *descriptions of admissible structures* when negation or disjunction were introduced (Blackburn 1993, see the end of Section 1).

1.3 *Related work*

Hybrid logic with nominals but without quantification over states was already used to describe semantic dependency graphs by Baldridge and Kruijff (2002). Natural language quantification is there encoded using RESTR and BODY relations. However, it remains unclear how to compute relations between such representations (e.g., how to check that *John kisses Mary* holds in case *every man kisses Mary* holds). An additional step of interpretation of the graphs seems to be required.

A similar approach is proposed by Kallmeyer and Richter (2014) for quantification in frame semantics. In this approach, "quantifier frames" also introduce RESTR and BODY attributes that point to nodes (typically representing an entity and an event, respectively). But they do not directly encode the truth conditions that would be associated with a model-theoretic interpretation. Bridging the gap between the quantifier frame and the model-theoretic interpretation requires the additional extraction of a predicate-logical formula; this, in turn, can be model-theoretically interpreted in order to compute the truth value of the expression.

Baldridge and Kruijff (2002) and Kallmeyer and Richter (2014) consider frames to be expressions that need to be further interpreted, possibly as formulas of predicate logic that, in turn, can be given a model-theoretic interpretation in a usual way.

In the approach we presently propose, and contrary to these previous approaches, there is no quantifier frame as such. The frames themselves are the models. The quantifiers are part of the formulas *describing* or *constraining* the frames (as models of the logical formulas) that can make an expression true. The HL formulas are the expressions to be interpreted as frames. But for the latter, no additional interpretation is required. The logical operators and the frames (as models) are kept separate, following the approach suggested by Muskens (2013).

2 HYBRID LOGIC AND SEMANTIC FRAMES

2.1 *Hybrid logic*

We use the notations of Areces and ten Cate (2007).

Definition 1 (Formulas)**.** Let $\mathsf{Rel} = \mathsf{Func} \cup \mathsf{PropRel}$ be a set of functional and non-functional relational symbols, Prop a set of propositional variables, Nom a set of nominals (node names), and Svar a set of state variables. Let $\mathsf{Stat} = \mathsf{Nom} \cup \mathsf{Svar}$.

The language of formulas Forms is defined as:

$$\mathsf{Forms} ::= \top \mid p \mid s \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle R\rangle\phi \mid \mathbf{\exists}\phi \mid @_s\phi \mid \downarrow x.\phi \mid \exists x.\phi$$

where $p \in \mathsf{Prop}$, $s \in \mathsf{Stat}$, $R \in \mathsf{Rel}$, $x \in \mathsf{Svar}$, and $\phi, \phi_1, \phi_2 \in \mathsf{Forms}$.

Moreover, we define:

- $\mathbf{\forall}\phi \equiv \neg\mathbf{\exists}\neg\phi$
- $[R]\phi \equiv \neg\langle R\rangle\neg\phi$
- $\phi \Rightarrow \psi \equiv \neg\phi \vee \psi$

We call $\mathbf{\forall}$ and $[R]$ universal operators, and $\mathbf{\exists}$ and $\langle R\rangle$ existential operators. The elements of Func will be written in small caps.

The $\langle R\rangle$ and $[R]$ operators are the usual modal operators corresponding to some accessibility relation R. The semantics of the new operators are given in the definitions to come, but the intuition behind them is as follows. The $\mathbf{\exists}\phi$ formula states that somewhere in the relational structure there is a node where ϕ holds. $\mathbf{\forall}\phi$ holds only if ϕ holds at each node of the structure. The binder $\downarrow$ in $\downarrow x.\phi$ gives the name x to the current node, so that x can be referred to arbitrarily deep in ϕ. The quantifier $\exists x.\phi$ does not change the evaluation node,

but states that some other node in the structure exists, is given the name x, and is such that ϕ (that possibly refers to x) is true at the current evaluation node. Finally, $@_s\phi$ states that ϕ holds at the node named s. With this operator, it can be checked from any place in the relational structure that the property ϕ holds at this specific node.

Definition 2 (Model)**.** A *model* $\mathcal{M}$ is a triple $\langle M, (R^{\mathcal{M}})_{R\in\mathsf{Rel}}, V\rangle$ such that M is a non-empty set, each $R^{\mathcal{M}}$ is a binary relation on M, and the valuation $V : \mathsf{Prop}\cup\mathsf{Nom} \longrightarrow \wp(M)$ is such that if $i \in \mathsf{Nom}$ then $V(i)$ is a singleton. An assignment g is a mapping $g : \mathsf{Svar} \longrightarrow M$. For an assignment g, g^x_m is an assignment that differs from g at most on x and $g^x_m(x) = m$. For $s \in \mathsf{Stat}$, we also define $[s]^{\mathcal{M},g}$ to be the only m such that $V(s) = \{m\}$ if $s \in \mathsf{Nom}$ and $[s]^{\mathcal{M},g} = g(s)$ if $s \in \mathsf{Svar}$.

Definition 3 (Satisfaction relation)**.** Let $\mathcal{M}$ be a model, $w \in M$, and g an assignment for $\mathcal{M}$. The *satisfaction relation* is defined as follows:

$$
\begin{array}{ll}
\mathcal{M},g,w \vDash \top & \\
\mathcal{M},g,w \vDash s & \text{iff } w = [s]^{\mathcal{M},g} \text{ for } s \in \mathsf{Stat} \\
\mathcal{M},g,w \vDash \neg\phi & \text{iff } \mathcal{M},g,w \nvDash \phi \\
\mathcal{M},g,w \vDash \phi_1 \wedge \phi_2 & \text{iff } \mathcal{M},g,w \vDash \phi_1 \text{ and } \mathcal{M},g,w \vDash \phi_2 \\
\mathcal{M},g,w \vDash \langle R\rangle\phi & \text{iff there is a } w' \in M \text{ such that} \\
 & \quad R^{\mathcal{M}}(w,w') \text{ and } \mathcal{M},g,w' \vDash \phi \\
\mathcal{M},g,w \vDash p & \text{iff } w \in V(p) \text{ for } p \in \mathsf{Prop} \\
\mathcal{M},g,w \vDash @_s\phi & \text{iff } \mathcal{M},g,[s]^{\mathcal{M},g} \vDash \phi \text{ for } s \in \mathsf{Stat} \\
\mathcal{M},g,w \vDash \downarrow x.\phi & \text{iff } \mathcal{M},g^x_w,w \vDash \phi \\
\mathcal{M},g,w \vDash \exists x.\phi & \text{iff there is a } w' \in M \text{ such that } \mathcal{M},g^x_{w'},w \vDash \phi \\
\mathcal{M},g,w \vDash \mathbf{\exists}\phi & \text{iff there is a } w' \in M \text{ such that } \mathcal{M},g,w' \vDash \phi
\end{array}
$$

We can then check that $\mathcal{M},g,w \vDash \mathbf{\forall}\phi$ iff $\forall w'\ \mathcal{M},g,w' \vDash \phi$. $\mathbf{\forall}$ is the universal modality. $\mathbf{\forall}\phi$ states that the property ϕ should hold at each node of the model.

Definition 4 (Satisfaction and validity)**.** A formula ϕ is:

- *satisfiable* if there is a model $\mathcal{M}$, and an assignment g on $\mathcal{M}$, and a state $w \in M$ such that $\mathcal{M},g,w \vDash \phi$
- *globally true* in a model $\mathcal{M}$ under an assignment g if it is satisfiable at all states of the model, i.e., $\mathcal{M},g,w \vDash \phi$ for all $w \in M$. We write $\mathcal{M},g \vDash \phi$
- *valid* if for all models $\mathcal{M}$ and assignments g, $\mathcal{M},g \vDash \phi$.

We can reformulate the frame of Figure 1 (Section 1) within this framework. The Prop vocabulary we use in the HL formulas corresponds to the unary relation symbols of Type used by Kallmeyer and Osswald (2013) to represent frames (see Section 1.1). The Nom vocabulary corresponds to the unary relation symbols of Label, and the Rel vocabulary subsumes the Feat binary relations of Kallmeyer and Osswald (2013). Note that the functionality of the members of Feat must be enforced separately by axioms. The semantic frame of Figure 1 is then a model that satisfies the formula (1) at the element named by l_0. This formula also highlights the crucial role of nominals in this setting. Several other formulas would be possible, but using an @ operator, here $@_v$, is needed to specify that the AT-REGION and the *part-of* edges meet on the same node (the $@_{l_2}$ is not completely necessary to describe the structure of Figure 1 but naturally arises in a compositional computing of this representation).

$$\begin{aligned}(1)\quad & l_0 \wedge \mathit{motion} \wedge \langle \textsc{agent} \rangle (l_1 \wedge \mathit{man}) \wedge \langle \textsc{mover} \rangle l_1 \wedge \\ & \langle \textsc{goal} \rangle (l_2 \wedge \mathit{house}) \wedge \langle \textsc{manner} \rangle \mathit{walking} \wedge \\ & (\exists v\ w. \langle \textsc{path} \rangle (\mathit{path} \wedge \langle \textsc{endp} \rangle v) \wedge \\ & \qquad @_{l_2}(\langle \textsc{at-region} \rangle w) \wedge @_v(\langle \mathit{part\text{-}of} \rangle w))\end{aligned}$$

2.2 *Expressive power*

According to the satisfaction relation definition, $\downarrow$ and $\exists$ bind node variables without changing the current evaluation node. In addition to $\mathbf{\exists}$, Blackburn and Seligman (1995) introduce another quantifier Σ for which the satisfaction relation also changes the evaluation node:[1]

$$\mathcal{M}, g, w \vDash \Sigma x.\phi \text{ iff } \exists w'\ \mathcal{M}, g^x_{w'}, w' \vDash \phi$$

This defines two independant families of operators: $\downarrow$ and $\exists$, and $\mathbf{\exists}$ and Σ.[2] However, using any two operators of both families (for instance $\downarrow$ and $\mathbf{\exists}$, the "weakest" ones) is expressively equivalent to using the most expressive fragment of the hybrid languages (the full hybrid language).

[1] Blackburn and Seligman (1995) call $\mathbf{\exists}$ the *somewhere* operator, and write it $\Diamond$, and $\mathbf{\forall}$ is the *universal* modality, written $\Box$.

[2] Note that $\downarrow$ can be defined in terms of $\exists$ by $\downarrow x.\phi \equiv \exists x.x \wedge \phi$ and that $\mathbf{\exists}$ can be defined in terms of Σ by $\mathbf{\exists}\phi \equiv \Sigma z.\phi$ with z not occurring in ϕ.

It is usual to refer to the hybrid languages $\mathscr{H}(\theta_1, \ldots, \theta_n)$ as the extension of the modal language with nominals and the operators $\theta_1, \ldots, \theta_n \in \{\downarrow, @, \mathbf{\exists}, \exists\}$. It is worth noting that even using the simplest binder $\downarrow$ already causes the satisfiability problem for $\mathscr{H}(\downarrow)$ to be undecidable (Areces *et al.* 1999) where the satisfiability problem corresponds to answering the question whether given a formula ϕ, there is a model $\mathscr{M}$, an assignment g and a node w such that $\mathscr{M}, g, w \vDash \phi$.

Nevertheless, there are syntactic restrictions on formulas that make the satisfiability problem decidable. In particular, formulas of the full hybrid language that do not contain the pattern "universal operator scoping over a $\downarrow$ operator scoping over a universal operator" have a decidable satisfiability problem (ten Cate and Franceschet 2005). Such formulas are used by Kallmeyer *et al.* (2015).

But the formulas we use in the present paper do show this pattern. On the other hand, they do not use the pattern "existential operator scoping over a $\downarrow$ operator scoping over an existential operator". For such formulas, the *validity problem* is shown to be decidable (ten Cate and Franceschet 2005). Although the validity problem for first-order logic is undecidable, this result by itself does not really improve on first-order logic representations. A more promising approach would be to consider semantic restrictions of the underlying class of models. For instance, (Schneider 2007) describes some classes where decidability results hold. As we do not take advantage so far of the Frame Semantics hypothesis that considers attributes to be functional, the class of models with such a semantic restriction is a natural candidate for studying the satisfiability problem. In any case, for every hybrid language, testing a given formula against a given finite model is decidable (Franceschet and de Rijke 2006).

2.3 *Frame semantics with quantification*

Since the models we are considering are *semantic frames* instead of arbitrary first-order models, we first present some models in which we consider the sentences (2a), (3a), and (4a). When the model is the frame of Figure 2, we expect (2a) to be true. There indeed is a *kissing* event with AGENT and THEME attributes linking to persons named (represented by the NAME attribute) *John* and *Mary* respectively. Accordingly, we wish to represent the semantics of (2a) by the hybrid logic formula (2b).

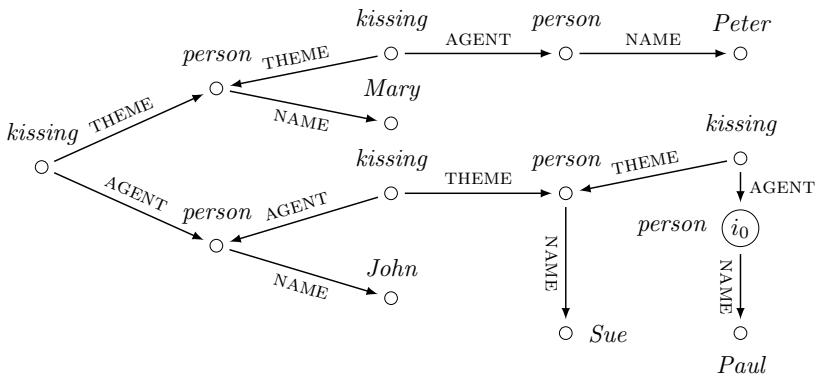


Figure 2: Quantification

On the other hand, (3a) is expected to be false as there is a person named *Paul* who is AGENT of a single *kissing* event whose THEME is a person named *Sue*. The frame of Figure 2 indeed falsifies the formula (3b) because we can find a node (namely, i_0) at which *man* holds,[3] but there is no *kissing* node from which we can both reach i_0 through an AGENT relation and, through a THEME relation, a node at which *person* $\wedge \langle$NAME$\rangle$*Mary* also holds.

With the object wide scope reading, we also expect (4a) to be false in the frame of Figure 2 because while the person named *Paul* and *Peter* both are AGENT of *kissing* events, these events do not have the same THEME. However, with the subject wide scope reading, (4a) is expected to be true in this frame.

(2) *a.* *John kisses Mary*

b. $\exists(\textit{kissing} \wedge \langle\text{AGENT}\rangle(\textit{person} \wedge \langle\text{NAME}\rangle\textit{John}) \wedge$
$\langle\text{THEME}\rangle(\textit{person} \wedge \langle\text{NAME}\rangle\textit{Mary}))$

(3) *a.* *Every man kisses Mary*

b. $\forall(\downarrow i.\textit{man} \Rightarrow \exists(\textit{kissing} \wedge \langle\text{AGENT}\rangle i \wedge$
$\langle\text{THEME}\rangle(\textit{person} \wedge \langle\text{NAME}\rangle\textit{Mary})))$

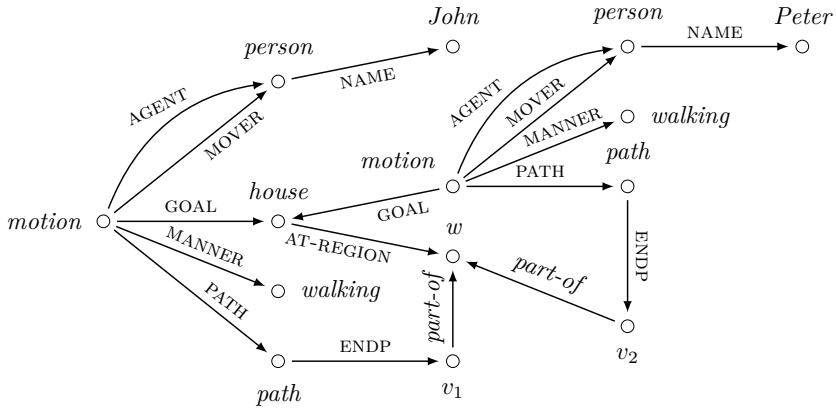
(4) *a.* *Every man kisses some woman*

b. $\forall(\downarrow i.\textit{man} \Rightarrow \exists(\downarrow i'.\textit{woman} \wedge$
$\exists(\textit{kissing} \wedge \langle\text{AGENT}\rangle i \wedge \langle\text{THEME}\rangle i')))$

c. $\exists(\downarrow i.\textit{woman} \wedge \forall(\downarrow i'.\textit{man} \Rightarrow$
$\exists(\textit{kissing} \wedge \langle\text{AGENT}\rangle i' \wedge \langle\text{THEME}\rangle i)))$

[3] Actually, in Figure 2, only *person* holds at i_0. We can have *man* hold as well with the additional postulate that (*person* $\wedge \langle$NAME$\rangle$*Paul*) $\Rightarrow$ *man*, and similarly of each node with a NAME attribute.

Figure 3: Quantification and node sharing



(5a) shows how state storing with the ↓ operator correctly interacts with the @ operator in order to describe node sharing. This sentence is expected to be true (both readings) in the model given by the frame of Figure 3.The frame semantics analysis of bounded motions verbs in (Kallmeyer and Osswald 2013) requires the *motion* to have a GOAL attribute. It is moreover required that the node reached is the same as the one of the entity provided by the *PP*. We express this requirement in the HL formulas (5b) and (5c):

1. by binding to the variable i' a ***house*** node,
2. by binding to the variable g a node that is accessible from the ***motion*** node via the $\langle$GOAL$\rangle$ relation,
3. and by stating that i' and g should be the same node, i.e., $g \wedge i'$ should hold.

(5) *a.* *Every man walked to some house*

b. $\mathbf{\forall}(\downarrow i.\mathit{man} \Rightarrow (\mathbf{\exists}(\downarrow i'.\mathit{house} \wedge$
$(\exists a\ g.\mathbf{\exists}(\mathit{motion} \wedge \langle\text{AGENT}\rangle a \wedge \langle\text{MOVER}\rangle a \wedge \langle\text{GOAL}\rangle g \wedge$
$\langle\text{PATH}\rangle \mathit{path} \wedge \langle\text{MANNER}\rangle \mathit{walking} \wedge @_a i \wedge$
$(\exists r\ v\ w.\mathit{event} \wedge \langle\text{PATH}\rangle(\mathit{path} \wedge \langle\text{ENDP}\rangle v) \wedge$
$@_r(\langle\text{AT-REGION}\rangle w) \wedge @_v(\langle \mathit{part\text{-}of} \rangle w) \wedge @_r(g \wedge i'))))))))$

c. $\mathbf{\exists}(\downarrow i'.\mathit{house} \wedge (\mathbf{\forall}(\downarrow i.\mathit{man} \Rightarrow$
$(\exists a\ g.\mathbf{\exists}(\mathit{motion} \wedge \langle\text{AGENT}\rangle a \wedge \langle\text{MOVER}\rangle a \wedge \langle\text{GOAL}\rangle g \wedge$
$\langle\text{PATH}\rangle \mathit{path} \wedge \langle\text{MANNER}\rangle \mathit{walking} \wedge @_a i \wedge$
$(\exists r\ v\ w.\mathit{event} \wedge \langle\text{PATH}\rangle(\mathit{path} \wedge \langle\text{ENDP}\rangle v) \wedge$
$@_r(\langle\text{AT-REGION}\rangle w) \wedge @_v(\langle \mathit{part\text{-}of} \rangle w) \wedge @_r(g \wedge i'))))))))$

Our goal is to compositonally associate each expression in natural language to an HL formula. This logical formula is to be checked against the possible models, and the sentence is true w.r.t. a model $\mathcal{M}$ in case this model satisfies the logical formula. More precisely, given a sentence s and its semantic representation $[\![s]\!]$, we say that s *is true* iff for all assignments g, $\mathcal{M}, g \vDash [\![s]\!]$ (i.e., $[\![s]\!]$ is globally true in $\mathcal{M}$ under any assignment).

Note that we use several modal operators. Each of them describes the accessibility relations corresponding to one of the attributes we find in frames (AGENT, GOAL, etc.). They should not be confused with other possible modal operators that are used for natural language semantics (e.g., knowledge and belief, intensionality, etc.). Clarifying the interaction between these different kinds of modal operators, for instance following Blackburn and Rijke (1997), is an important issue. But this goes beyond the scope of this paper and is left for future work.

3 SYNTAX–SEMANTICS INTERFACE WITH ABSTRACT CATEGORIAL GRAMMARS

In order to exemplify our approach to quantification in frame semantics, we rely on the framework of Abstract Categorial Grammars (ACG) (de Groote 2001). ACGs derive from type-theoretic grammars in the tradition of Lambek (1958), Curry (1961), and Montague (1974). Rather than being a grammatical formalism on their own, they provide a framework in which several grammatical formalisms may be encoded (de Groote and Pogodalla 2004). Since our focus is on the semantic modelling of quantification in frame semantics and its compositional account, we provide a Montague grammar based syntactic modelling that is sufficient for our purpose. Integration of the modelling of scope ambiguity in a TAG encoding (de Groote 2002) for instance would require an embedding into an underspecified representation language (Bos 1995; Pogodalla 2004; Kallmeyer and Romero 2008) that plays no role in the final interpretation of the logical formula to be interpreted.

3.1 *Abstract Categorial Grammars*

The definition of an ACG is based on a small set of mathematical primitives from type theory, λ-calculus, and linear logic. These primitives

combine via simple composition rules, offering ACGs good flexibility. In particular, ACGs generate languages of linear λ-terms, which generalize both string and tree languages. Crucially, ACG provides the user with direct control over the parse structures of the grammar, the *abstract language*. Such structures are later interpreted by a morphism, the *lexicon*, to get the concrete *object language*. A *vocabulary* is the *higher-order signature* that defines the atomic elements (atomic types and typed constants).

For sake of self-containedness, we review here the basic definitions of ACGs.

Definition 5 (Types)**.** Let A be a set of atomic types. The set $\mathscr{T}(A)$ of *implicative types* built upon A is defined with the following grammar:

$$\mathscr{T}(A) ::= A | \mathscr{T}(A) \multimap \mathscr{T}(A) | \mathscr{T}(A) \to \mathscr{T}(A)$$

The set of *linear implicative types* built upon A is defined with the following grammar:

$$\mathscr{T}^0(A) ::= A | \mathscr{T}^0(A) \multimap \mathscr{T}^0(A)$$

Definition 6 (Higher-order signatures)**.** A *higher-order signature* Σ is a triple $\Sigma = \langle A, C, \tau \rangle$ where:

- A is a finite set of atomic types;
- C is a finite set of constants;
- $\tau : C \to \mathscr{T}(A)$ is a function assigning types to constants.

A higher-order signature $\Sigma = \langle A, C, \tau \rangle$ is *linear* if the codomain of τ is $\mathscr{T}^0(A)$.

Definition 7 (λ-Terms)**.** Let X be an infinite countable set of λ-variables. The set $\Lambda(\Sigma)$ of λ-terms built upon a higher-order signature $\Sigma = \langle A, C, \tau \rangle$ is inductively defined as follows:

- if $c \in C$ then $c \in \Lambda(\Sigma)$;
- if $x \in X$ then $x \in \Lambda(\Sigma)$;
- if $x \in X$ and $t \in \Lambda(\Sigma)$ and x occurs free in t exactly once, then $\lambda^o x.t \in \Lambda(\Sigma)$;
- if $x \in X$ and $t \in \Lambda(\Sigma)$, then $\lambda x.t \in \Lambda(\Sigma)$;
- if $t, u \in \Lambda(\Sigma)$ then $(t\, u) \in \Lambda(\Sigma)$.

Note there is a linear λ-abstraction (denoted by λ^{o}) and a (usual) intuitionistic λ-abstraction (denoted by λ). There also are the usual notions of α, β, and η conversions (Barendregt 1984).

Definition 8 (Typing judgment)**.** Given a higher-order signature Σ, the typing rules are given with an inference system whose judgments are of the form: $\Gamma; \Delta \vdash_{\Sigma} t : \alpha$ where:

- Γ is a finite set of non-linear variable typing declarations;
- Δ is a finite set of linear variable typing declarations.

Both Γ and Δ may be empty. If both of them are empty, we usually write $t : \alpha$ (t is of type α) instead of $\vdash_{\Sigma} t : \alpha$. Moreover, we drop the Σ subscript when the context permits. Table 1 gives the typing rules.

Table 1: Typing rules for deriving typing judgments

$$\frac{}{\Gamma; \vdash_{\Sigma} c : \tau(c)}\ \text{(const.)}$$

$$\frac{}{\Gamma; x : \alpha \vdash_{\Sigma} x : \alpha}\ \text{(lin. var.)} \qquad \frac{}{\Gamma, x : \alpha; \vdash_{\Sigma} x : \alpha}\ \text{(var.)}$$

$$\frac{\Gamma; \Delta, x : \alpha \vdash_{\Sigma} t : \beta}{\Gamma; \Delta \vdash_{\Sigma} \lambda^{o} x.t : \alpha \multimap \beta}\ \text{(l. abs.)} \qquad \frac{\Gamma; \Delta_1 \vdash_{\Sigma} t : \alpha \multimap \beta \quad \Gamma; \Delta_2 \vdash_{\Sigma} u : \alpha}{\Gamma; \Delta_1, \Delta_2 \vdash_{\Sigma} (t\,u) : \beta}\ \text{(l. app.)}$$

$$\frac{\Gamma, x : \alpha; \Delta \vdash_{\Sigma} t : \beta}{\Gamma; \Delta \vdash_{\Sigma} \lambda x.t : \alpha \to \beta}\ \text{(abs.)} \qquad \frac{\Gamma; \Delta \vdash_{\Sigma} t : \alpha \to \beta \quad \Gamma; \vdash_{\Sigma} u : \alpha}{\Gamma; \Delta \vdash_{\Sigma} (t\,u) : \beta}\ \text{(app.)}$$

Remark. In the rule (app.), the linear context needs to be empty. Otherwise, a linear variable occurring in u could be duplicated or removed if the non-linear abstracted variable in t for which it substitutes in a β-reduction is duplicated or removed.

Definition 9 (Lexicon)**.** Let $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ be two higher-order signatures, Σ_1 being linear. A lexicon $\mathscr{L} = \langle F, G \rangle$ from Σ_1 to Σ_2 is such that:

- $F : A_1 \to \mathscr{T}(A_2)$. We also note $F : \mathscr{T}^0(A_1) \to \mathscr{T}(A_2)$ its homomorphic extension;[4]
- $G : C_1 \to \Lambda(\Sigma_2)$. We also note $G : \Lambda(\Sigma_1) \to \Lambda(\Sigma_2)$ its homomorphic extension;

[4] Such that $F(\alpha \multimap \beta) = F(\alpha) \multimap F(\beta)$ and $F(\alpha \to \beta) = F(\alpha) \to F(\beta)$

- F and G are such that for all $c \in C_1$, $\vdash_{\Sigma_2} G(c) : F(\tau_1(c))$ is provable.

We also use $\mathscr{L}$ instead of F or G.

Definition 10 (Abstract Categorial Grammar and vocabulary)**.** An *abstract categorial grammar* is a quadruple $\mathscr{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}, S \rangle$ where:

- $\Sigma_1 = \langle A_1, C_1, \tau_1 \rangle$ and $\Sigma_2 = \langle A_2, C_2, \tau_2 \rangle$ are two higher-order signatures and Σ_1 is linear. Σ_1 is called the *abstract vocabulary* and $\Lambda(\Sigma_1)$ is the set of *abstract terms*; similarly, Σ_2 is called the *object vocabulary* and $\Lambda(\Sigma_2)$ is the set of *object terms*.
- $\mathscr{L} : \Sigma_1 \to \Sigma_2$ is a lexicon.
- $S \in \mathscr{T}(A_1)$ is the *distinguished type* of the grammar.

Given an ACG $\mathscr{G}_{name} = \langle \Sigma_1, \Sigma_2, \mathscr{L}_{name}, S \rangle$, we use the following notational variants for the interpretation of the type α (resp. the term t): $\mathscr{L}_{name}(\alpha) = \beta$, $\mathscr{G}_{name}(\alpha) = \beta$, $\alpha :=_{name} \beta$, and $[\![\alpha]\!]_{name} = \beta$ (resp. $\mathscr{L}_{name}(t) = u$, $\mathscr{G}_{name}(t) = u$, $t :=_{name} u$, and $[\![t]\!]_{name} = u$). The subscript may be omitted if clear from the context.

Definition 11 (Abstract and object languages)**.** Given an ACG $\mathscr{G}$, the *abstract language* is defined by

$$\mathscr{A}(\mathscr{G}) = \{t \in \Lambda(\Sigma_1) \mid \vdash_{\Sigma_1} t : S \text{ is derivable}\}$$

The *object language* is defined by

$$\mathscr{O}(\mathscr{G}) = \{u \in \Lambda(\Sigma_2) \mid \exists t \in \mathscr{A}(\mathscr{G}) \text{ s.t. } u = \mathscr{L}(t)\}$$

3.2 *The syntax-semantics interface as ACG composition*

The lexicon defines the way structures are interpreted. It plays a crucial role in the way ACG models the syntax-semantics interface. The basic idea is to have a given (abstract) structure interpreted either as a surface form (e.g., a string) or as a meaning form (e.g., a logical formula). This boils down to having two interpretations that share the same abstract vocabulary, hence mapping a single structure into two different ones. This composition is illustrated by $\mathscr{G}_{form}$ and $\mathscr{G}_{meaning}$ sharing the $\Sigma_{abstract}$ vocabulary in Figure 4.

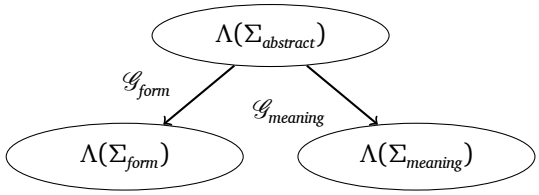


Figure 4:
ACG composition for the syntax-semantics interface

4 TYPE-THEORETIC SEMANTICS WITH FRAMES

We now provide the type-theoretic syntax-semantics interface allowing for a compositional building of the meanings. We use the architecture described in Figure 4. As we are concerned in this article with semantic modelling and quantification rather than with parsing, we use higher-order types for quantified noun-phrases.

All the following examples can be run and tested with the ACG toolkit[5] and the companion example files.[6]

4.1 *The ACG of surface forms*

At the abstract level, we use the signature defined with the type assignment of Table 2. It makes use of the usual syntactic types: *NP*, *S*, *N*, and *PP*. Note that following the usual type-logical approach, determiners have a higher-order type.

John, Mary	$: NP$	kisses	$: NP \multimap NP \multimap S$
man, woman, house	$: N$	every, some	$: N \multimap (NP \multimap S) \multimap S$
to, into	$: NP \multimap PP$	walked	$: PP \multimap NP \multimap S$

Table 2:
$\Sigma_{abstract}$ type assignement

The object vocabulary of surface forms uses the standard modelling of strings as λ-terms. It is built on Σ_{form} that contains a single atomic type o, and the type σ (for strings) is defined by $\sigma \triangleq o \multimap o$. The concatenation is then defined as functional composition by $\cdot + \cdot = \lambda^o f\ g.\lambda^o z.f\ (g\ z) : \sigma \multimap \sigma \multimap \sigma$. It is associative, and it admits the identity function $\epsilon \triangleq \lambda^o x.x : \sigma$ as a neutral element. Σ_{form} also contains the constants *John*, *Mary*, *kisses*, *every*, *man* ... of type σ.

The ACG $\mathscr{G}_{form}$ is then defined using the interpretations given in Table 3. The terms defined in Equations (6) correspond to the syntactic

[5] ACGtk can be downloaded and installed from `http://calligramme.loria.fr/acg/#Software`.

[6] These files are available at `https://hal.inria.fr/hal-01417853/file/quantification-and-frames.zip`.

derivations of the sentence for which we want to provide a semantic representation. Their surface forms are given by Equations (7)–(12).

(6)
$$
\begin{aligned}
u_{2b} &= \text{kisses Mary John}\\
u_{3b} &= \text{(every man) } (\lambda^{o}x.\text{kisses Mary } x)\\
u_{4b} &= \text{(every man) } (\lambda^{o}x.\text{(some woman) } (\lambda^{o}y.\text{kisses } y\ x))\\
u_{4c} &= \text{(some woman) } (\lambda^{o}y.\text{(every man) } (\lambda^{o}x.\text{kisses } y\ x))\\
u_{5b} &= \text{(every man) } (\lambda^{o}x.\text{(some house)}(\lambda^{o}y.\text{walked (to } y)\ x))\\
u_{5c} &= \text{(some house)}(\lambda^{o}y.\text{(every man) } (\lambda^{o}x.\text{walked (to } y)\ x))
\end{aligned}
$$

(7) $u_{2b} :=_{form} \mathit{John} + \mathit{kisses} + \mathit{Mary}$

(8) $u_{3b} :=_{form} \mathit{every} + \mathit{man} + \mathit{kisses} + \mathit{Mary}$

(9) $u_{4b} :=_{form} \mathit{every} + \mathit{man} + \mathit{kisses} + \mathit{some} + \mathit{woman}$

(10) $u_{4c} :=_{form} \mathit{every} + \mathit{man} + \mathit{kisses} + \mathit{some} + \mathit{woman}$

(11) $u_{5b} :=_{form} \mathit{every} + \mathit{man} + \mathit{walked} + \mathit{to} + \mathit{some} + \mathit{house}$

(12) $u_{5c} :=_{form} \mathit{every} + \mathit{man} + \mathit{walked} + \mathit{to} + \mathit{some} + \mathit{house}$

Table 3: $\mathscr{G}_{form}$ interpretation of the abstract atomic types and constants

John	$:=_{form}$ *John*	Mary	$:=_{form}$ *Mary*
man	$:=_{form}$ *man*	woman	$:=_{form}$ *woman*
house	$:=_{form}$ *house*		
to	$:=_{form} \lambda^{o}n.\mathit{to} + n$	into	$:=_{form} \lambda^{o}n.\mathit{into} + n$
every	$:=_{form} \lambda^{o}n\ P.P\ (\mathit{every} + n)$	some	$:=_{form} \lambda^{o}n\ P.P\ (\mathit{some} + n)$
kisses	$:=_{form} \lambda^{o}o\ s.s + \mathit{kissed} + o$	walked	$:=_{form} \lambda^{o}p\ s.s + \mathit{walked} + p$

4.2 *The ACG of meaning representations*

In accordance with the ACG architecture of Figure 4, the syntax-semantics interface relies on sharing the abstract language of the two ACGs responsible for the surface interpretation on the one hand and for the semantic interpretation on the other hand. The abstract vocabulary we use is $\Sigma_{abstract}$, defined in the previous section.

Our goal is to associate every sentence with a hybrid-logical formula. It's important to note that we are not concerned with higher-order hybrid logic in this work; not even first-order hybrid logic. The binders and quantifiers we use only bind node variables, and not entities nor higher-order predicates. This contrasts with quantified hybrid

logic (QHL) (Blackburn and Marx 2002). We do not directly adopt the Hybrid Type Theory (HTT) proposed by Areces *et al.* (2011, 2014). Contrary to what could be expected from (Gallin 1975) type theory of higher-order modal logic, Areces *et al.* (2014) do not use a specific type s to denote nodes (or worlds) and nominals are typed t as propositions.

We do introduce a specific type s for nominals, so that the set of atomic types of $\Sigma_{meaning}$ is $\{s, t\}$. We also introduce a coercion operator $\# : s \to t$ in order to use nominals as propositions in formulas. This ensures we only build formulas of Forms. Table 4 shows the semantic constants we use, including logical operators and quantifiers.

$event, kissing, motion, person, John, Mary, \ldots$	$: t$
$\langle \text{AGENT} \rangle, \langle \text{THEME} \rangle, \langle \text{MOVER} \rangle, \langle part\text{-}of \rangle, \ldots$	$: t \multimap t$
$\#$	$: s \to t$
$\wedge, \Rightarrow$	$: t \multimap t \multimap t$
$@$	$: s \to t \multimap t$
$\mathbf{\exists}, \mathbf{\forall}$	$: t \multimap t$
$\downarrow, \exists$	$: (s \to t) \multimap t$

Table 4: Constant terms of the semantic language

We can now define $\mathscr{G}_{meaning}$ using the interpretations of the atomic types of the constants of Table 5. We follow Kallmeyer and Osswald (2013) in the semantics and meaning decomposition of motion verbs.

$S, NP, N :=_{meaning} t$	$PP :=_{meaning} t \multimap t$
John	$:=_{meaning} \mathit{John}$
Mary	$:=_{meaning} \mathit{Mary}$
man	$:=_{meaning} \mathit{man}$
woman	$:=_{meaning} \mathit{woman}$
house	$:=_{meaning} \mathit{house}$
some	$:=_{meaning} \lambda^{\mathbf{o}} P\ Q.\mathbf{\exists}(\downarrow i.P \wedge (Q\ (\# i)))$
every	$:=_{meaning} \lambda^{\mathbf{o}} P\ Q.\mathbf{\forall}(\downarrow i.P \Rightarrow (Q\ (\# i)))$
kisses	$:=_{meaning} \lambda^{\mathbf{o}} o\ s.\mathbf{\exists}(\mathit{kissing} \wedge \langle \text{AGENT} \rangle s \wedge \langle \text{THEME} \rangle o)$
walked	$:=_{meaning} \lambda^{\mathbf{o}} pp\ s.\exists a\ g.\mathbf{\exists}(\mathit{motion} \wedge \langle \text{AGENT} \rangle (\# a) \wedge \langle \text{MOVER} \rangle (\# a) \wedge \langle \text{GOAL} \rangle (\# g) \wedge \langle \text{PATH} \rangle \mathit{path} \wedge \langle \text{MANNER} \rangle \mathit{walking} \wedge @_a s \wedge (pp\ (\# g)))$
to	$:=_{meaning} \lambda^{\mathbf{o}} n\ g.\exists r\ v\ w.\mathit{event} \wedge \langle \text{PATH} \rangle (\mathit{path} \wedge \langle \text{ENDP} \rangle v) \wedge @_r \langle \text{AT-REGION} \rangle (\# w) \wedge @_v \langle part\text{-}of \rangle (\# w) \wedge @_r (g \wedge n)$
into	$:=_{meaning} \lambda^{\mathbf{o}} n\ g.\exists r\ v\ w.\mathit{event} \wedge \langle \text{PATH} \rangle (\mathit{path} \wedge \langle \text{ENDP} \rangle v) \wedge @_r \langle \text{IN-REGION} \rangle (\# w) \wedge @_v \langle part\text{-}of \rangle (\# w) \wedge @_r (g \wedge n)$

Table 5: Semantic interpretation of the constants of $\Sigma_{abstract}$

Remark. Nominal variables are allowed to occur non-linearly in semantic terms. This is required, for instance, in order to specify that a same nominal is reached from two different paths (see for instance the variable a in $[\![\text{walked}]\!]_{meaning}$ in Table 5).

What the ACG framework does not express, though, are the lexical or meaning postulates that can be added to the logical theory. Such postulates are additional constraints that any model should also satisfy and that do not depend on the actual semantic representation that is being built. They include for instance the representation of the ontology of propositions (types, in the frame semantics terminology) such as: *man* $\Rightarrow$ *person*, or any standard modal-logical axiom such as $\Box(p \Rightarrow q) \Rightarrow (\Box p \Rightarrow \Box q)$.

It follows that the following equalities hold, where t_{2b} is the term in (2b), t_{3b} is the term in (3b), etc., such that every nominal variable is preceded by the # coercion operator:

(13) $[\![\text{kisses Mary John}]\!] = t_{2b}$

(14) $[\![(\text{every man})\ (\lambda^o x.\text{kisses Mary } x)]\!] = t_{3b}$

(15) $[\![(\text{every man})\ (\lambda^o x.(\text{some woman})\ (\lambda^o y.\text{kisses } y\ x))]\!] = t_{4b}$

(16) $[\![(\text{some woman})\ (\lambda^o y.(\text{every man})\ (\lambda^o x.\text{kisses } y\ x))]\!] = t_{4c}$

Table 5 shows the interaction of the storing operator with path equalities. It compositionally derives from the verb and the preposition semantic interpretations. In the verb semantics, the path equalities specify that the MOVER and the AGENT attributes of the event are the same, and that the information provided by the *pp* argument should hold for the GOAL g. In its semantics, the preposition contributes on the one hand to the main event (as the *event* proposition is evaluated at the current state) and on the other hand by specifying that the g state (meant to be the target node of the verb that the proposition modifies, here the target of the GOAL attribute) should be identified to the n argument (the noun phrase which is argument of the preposition). This leads to the interpretations (5b) and (5c) of (5a) given in (17) and (18).

(17) $[\![(\text{every man})\ (\lambda^o x.(\text{some house})(\lambda^o y.\text{walked (to } y)\ x))]\!] = t_{5b}$

(18) $[\![(\text{some house})(\lambda^o y.(\text{every man})\ (\lambda^o x.\text{walked (to } y)\ x))]\!] = t_{5c}$

5 TYPE COERCION AS EXISTENTIAL QUANTIFICATION

We now have two ingredients at our disposal: the decomposition of the lexical semantics offered by frame semantics, and the power of binding states. We illustrate how to combine them in order to model semantic coercion. Sentence (19) shows how a predicate can take another event predicate as argument. On the other hand, sentence (20) shows that the same predicate can take a noun phrase as argument. In the latter case, it instead conveys the meaning that the entity referred to by the noun phrase should be part of some event. It is even the case that if this event is not salient in the context, it can be inferred from the lexicon, for instance using the *qualia* structure and the telic quale as defined by the Generative Lexicon (Pustejovsky 1998), or a subclass of the $\mathbf{S}_2$ lexical function in the framework of the Explanatory and Combinatorial Lexicology (Mel'čuk *et al.* 1995; Polguère 2003).

(19) *John began to read a book*

(20) *John began a book*

We first model (19). The assumed syntactic constructions are given by the extension of the $\Sigma_{abstract}$ signature of Table 6 and by its interpretation by $\mathscr{G}_{form}$ of Table 7.

Semantically, the idea is that events are structured (Moens and Steedman 1988). We in particular consider the structures required by aspectual predicates such as *begin* as in (Pustejovsky and Bouillon 1995). We structure the events with the notion of *transition* that has an ANTE attribute and a POST attribute (see Figure 5). When an event has begun, it is set as the value of the POST attribute. This is what the interpretation of begin_1 in Table 8 states. This interpretation also requires the event argument to be a process (*proc*) or an accomplishment (*acc*) (Im and Lee 2015).

begin_1	$: S_{inf} \multimap NP \multimap S$
begin_2	$: NP \multimap NP \multimap S$
to read	$: NP \multimap S_{inf}$

Table 6:
Extension of $\Sigma_{abstract}$

S_{inf}	$:=_{form} \sigma$
begin_1	$:=_{form} \lambda^{o} c\ s.s + began + c$
$\mathsf{begin}_2', \mathsf{begin}_2$	$:=_{form} \lambda^{o} o\ s.s + began + o$
to read	$:=_{form} \lambda^{o} o.to + read + o$

Table 7:
Interpretation of types and constants of $\Sigma_{abstract}$ by $\mathscr{G}_{form}$

Figure 5: Event structure

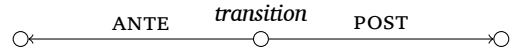


Table 8: Interpretation of types and constants of $\Sigma_{abstract}$ by $\mathscr{G}_{meaning}$

S_{inf}	$:=_{meaning}\ t \multimap t$
begin_1	$:=_{meaning}\ \lambda^{o} c\ s.\exists(\mathit{transition} \wedge \langle \textsc{post} \rangle((\mathit{proc} \vee \mathit{acc}) \wedge (c\ s)))$
begin'_2	$:=_{meaning}\ \lambda^{o} o\ s.\exists(\mathit{transition} \wedge \langle \textsc{post} \rangle ((\mathit{proc} \vee \mathit{acc}) \wedge \langle \textsc{agent} \rangle s \wedge \langle \textsc{ug} \rangle o))$
begin_2	$:=_{meaning}\ \lambda^{o} o\ s.\exists(\mathit{transition} \wedge \langle \textsc{post} \rangle (\downarrow s.(\mathit{proc} \vee \mathit{acc}) \wedge \langle \textsc{agent} \rangle s \wedge \langle \textsc{ug} \rangle (o \wedge \langle \mathit{proto} \rangle \langle \mathit{e\text{-}q} \rangle (\downarrow s'.@_s \langle \mathit{proto} \rangle (\# s')))))$
to read	$:=_{meaning}\ \lambda^{o} o\ s.\mathit{reading} \wedge \langle \textsc{agent} \rangle s \wedge \langle \textsc{theme} \rangle o$

(21) (a book) ($\lambda^{o} y.\mathsf{begin}_1$ (to read y) John) $:=_{forms}$
$\mathit{John} + \mathit{began} + \mathit{to} + \mathit{read} + \mathit{a} + \mathit{book}$

(22) (a book) ($\lambda^{o} y.\mathsf{begin}_1$ (to read y) John) $:=_{meaning}$
$\exists(\downarrow i.\mathit{book} \wedge (\exists(\mathit{transition}$
$\wedge \langle \textsc{post} \rangle((\mathit{proc} \vee \mathit{acc}) \wedge \mathit{reading}$
$\wedge \langle \textsc{agent} \rangle(\mathit{person} \wedge \langle \textsc{name} \rangle \mathit{John})$
$\wedge \langle \textsc{theme} \rangle(\# i)))))$

With the provided ACGs, we can then compute the semantic interpretation of the syntactic derivation associated with (19). Equation (21) shows that the syntactic derivation indeed corresponds to the sentence, and Equation (22) shows its semantic interpretation. In order to be true, the model should have a node i where *book* holds, and a node where *transition* holds and from which there is a $\langle \textsc{post} \rangle \langle \textsc{theme} \rangle$ path to i.

It is actually this path that we require to exist in the semantic recipe for *begin* when used with a direct object. This requirement appears in the interpretation of begin'_2 as given by Table 8 by specifying that the event given as the value of the POST attribute itself has an UNDERGOER (UG) that should target the direct object. This interpretation also accounts for the following constraints (Pustejovsky and Bouillon 1995): the subject of *begin* is also the agent of the argument event, and the latter is either a process or an accomplishment. Equations (23) and (24) show the achieved effects from the derivation of (20).

(23) (a book) ($\lambda^{o} y.\mathsf{begin}'_2\ y$ John) $:=_{forms} \mathit{John} + \mathit{began} + \mathit{a} + \mathit{book}$

(24)
$$\begin{aligned}(\text{a book})\ (\lambda^{o} y.\text{begin}'_2\ y\ \text{John}) :=_{\textit{meaning}}\\ \exists(\downarrow i.\textit{book} \wedge \exists(\textit{transition}\\ \wedge\langle\text{POST}\rangle((\textit{proc} \vee \textit{acc}) \wedge \langle\text{AGENT}\rangle(\textit{person}\\ \wedge\langle\text{NAME}\rangle\textit{John}) \wedge \langle\text{UG}\rangle(\#\, i))))\end{aligned}$$

It is not specified, though, what kind of event it is: *reading*, *writing*, etc. The latter lexically depends on the object. We want to model this dependency by adding lexically determined conditions on the possible models that make the formula true. We already met conditions in the form of meaning postulates, such as $(\textit{person} \wedge \langle\text{NAME}\rangle\textit{Paul}) \Rightarrow \textit{man}$. The conditions we introduce now are different and also make use of another feature of hybrid logic that we have not used so far: actual nominals, and not only state variables. These nominals encode lexical properties of the entities to be used in the meaning representation of the lexical items.

So we introduce the nominals $i_{\text{book}}, i_{\text{reading}}, i_{\text{writing}}, i_{\text{translating}} \ldots$ corresponding to the propositions (or types, in the frame semantics terminology) *book*, *reading*, *writing*, *translating*… For each of these pairs, the following schema holds:

(25)
$$(\langle \textit{proto} \rangle i_{\text{p}}) \Rightarrow p$$

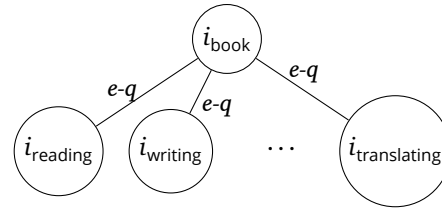
If we additionally require that each node has a *proto* attribute, each node in a frame should be associated with a prototypical node named by a nominal, and the proposition that holds at the former can be inferred from the latter.

We also encode that the i_{book} node is related through the *e-q* (event quale) relation to some event nominals, requiring the postulates of (26) to hold as well (see Figure 6).

(26)
$$\begin{aligned}&@_{i_{\text{book}}}\langle \textit{e-q} \rangle i_{\text{reading}}\\ &@_{i_{\text{book}}}\langle \textit{e-q} \rangle i_{\text{writing}}\\ &\ldots\end{aligned}$$

Remark (Nominals as prototypical entities). It is very important that the postulates of (26) use nominals rather than properties. Stating these postulates directly with propositions, such as $\textit{book} \wedge \langle \textit{e-q} \rangle \textit{reading}$, $\textit{book} \wedge \langle \textit{e-q} \rangle \textit{wrting}$, etc., would amount to require any node where *book* holds to relate to every (quale) events with an $\langle \textit{e-q} \rangle$ relation. These

Figure 6:
Qualia values associated to i_{book}



events would then be part of the model even if no linguistic element, such as *begin*, triggers them.

We can now state the following condition on a node s that has o as undergoer: when looking at the event quale of the prototype of o, if we call this event quale s', then s' should be a prototype of s. Formula (27) states this condition in hybrid logic terms. It is the formula used for the interpretations of begin_2 in Table 8. It can be paraphrased as follows: if s is a state that has o as undergoer, we set s' to be a quale event associated to o, via a prototype of o. For instance, if *book* holds at o, $o \wedge \langle \textit{proto} \rangle$ is i_{book}. Then s' is one of the i_{reading}, i_{writing}, etc. Say it is i_{reading}. $@_s \langle \textit{proto} \rangle s'$ finally ensures that the prototype of s is i_{reading}. Together with (25), we thus have that *reading* holds at s.

(27) $$\downarrow s.\langle \text{UG} \rangle (o \wedge \langle \textit{proto} \rangle \langle \textit{e-q} \rangle (\downarrow s'.@_s \langle \textit{proto} \rangle s'))$$

Thus, as Equations (28) and (29) show, together with the postulates (25) and (26), we have the semantic coercion of the object (here a book) to its associated possible telic quales through the *prototype* relation.

(28) $$\text{(a book)}\ (\lambda^{\text{o}} y.\text{begin}_2\ y\ \text{John}) :=_{\textit{forms}} \textit{John} + \textit{began} + a + \textit{book}$$

(29)
$$\begin{aligned} &\text{(a book)}\ (\lambda^{\text{o}} y.\text{begin}_2\ y\ \text{John}) :=_{\textit{meaning}} \\ &\exists (\downarrow i.\textit{book} \wedge \exists (\textit{transition} \\ &\wedge \langle \text{POST} \rangle (\downarrow s.(\textit{proc} \vee \textit{acc}) \wedge \langle \text{AGENT} \rangle (\textit{person} \\ &\wedge \langle \text{NAME} \rangle \textit{John}) \\ &\wedge \langle \text{UG} \rangle (\# i \wedge \langle \textit{proto} \rangle \langle \textit{e-q} \rangle (\downarrow s'.@_s \langle \textit{proto} \rangle (\# s'))))))) \end{aligned}$$

While accounting for the lexical knowledge, this approach makes no use of a possible specific context where it is *not* required to use the lexical information. For instance, (30) does not make sense without any context, as *ball* does not come with a telic quale.

(30) *John began his ball*

However, in a context that John was asked to paint a ball, for instance, this information could be used to correctly interpret (30). Such an account could possibly be provided by making use of a selection operator in some context, akin to the one proposed by de Groote (2006) and Lebedeva (2012). The introduction of node binders should indeed allow us to propose such an approach to a continuation-based approach to event context. In particular we may *presuppose* in the semantics of begin_2 the path conditions that apply to the object. If this property is already satisfied (for instance by a painting event), nothing else happens beyond the retrieval of this event. Otherwise, the telic quale of the object might be projected, possibly resulting in a failure if no prototypical telic quale is available.

6 CONCLUSION AND PERSPECTIVES

We used hybrid logic as a means to integrate logical operators with frame semantics. We illustrated the approach with the modelling of quantifier scopes. We embedded the proposed semantic representation within the Abstract Categorial Grammar framework in order to show how to compositionally derive different quantifier scope readings. We also showed how the key ingredients of hybrid logic, nominals, and binders can be used to model semantic coercion, such as the one induced by the *begin* predicate.

Binding nodes also offers the possibility of using continuation semantics in order to model a dynamic reference to events. In the particular case of semantic coercion, we plan to study how to integrate the model we proposed with a representation of the context. The projection of the telic quale of some (object) entity would then depend on the availability of some previously introduced events.

We also plan to take advantage of the semantic structuring induced by frame semantics to account for representation and copredication of dot type objects. More generally, frame semantics offers several ways to account for subtyping and meaning shifts. A first possibility is to use an ontology by means of axioms (e.g., $man \Rightarrow person$). A second possibility is to use structural properties of frames, as proposed for metonymy by Löbner (2014b). A third possibility is to combine the two previous techniques as we propose in this article, encoding

the qualia structures in prototype frames and linking nodes to their prototypes using axioms.

Finally, we plan to investigate the computational properties of the framework we propose with respect to the hybrid inferential systems (Blackburn and Marx 2002) and the specific properties induced by the frame models we consider, typically the functionality of the attribute relations (Schneider 2007). Modal and hybrid logics indeed generally provide better computability properties than first-order logic in terms of decidability and complexity. Some of these properties are lost when using quantification over nominals (see Section 2.2) and recovering them using restrictions induced by frames on models would be interesting in order to provide semantic representations with tractable automated reasoning capabilities.

REFERENCES

Carlos Areces, Patrick Blackburn, Antonia Huertas, and María Manzano (2011), Hybrid Type Theory: A Quartet in Four Movements, *Principia*, 15(2):225–247, DOI: 10.5007/1808-1711.2011v15n2p225.

Carlos Areces, Patrick Blackburn, Antonia Huertas, and María Manzano (2014), Completeness in Hybrid Type Theory, *Journal of Philosophical Logic*, 43(2-3):209–238, ISSN 0022-3611, DOI: 10.1007/s10992-012-9260-4.

Carlos Areces, Patrick Blackburn, and Maarten Marx (1999), A Road-Map on Complexity for Hybrid Logics, in Jörg Flum and Mario Rodriguez-Artalejo, editors, *Computer Science Logic: 13th International Workshop, CSL'99 8th Annual Conference of the EACSL Madrid, Spain, September 20–25, 1999 Proceedings*, pp. 307–321, Springer Berlin Heidelberg, DOI: 10.1007/3-540-48168-0_22.

Carlos Areces and Balder ten Cate (2007), Hybrid logics, in Patrick Blackburn, Johan Van Benthem, and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, chapter 14, pp. 821–868, Elsevier, DOI: 10.1016/S1570-2464(07)80017-6.

Jason Baldridge and Geert-Jan Kruijff (2002), Coupling CCG and Hybrid Logic Dependency Semantics, in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pp. 319–326, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, DOI: 10.3115/1073083.1073137, ACL anthology: P02-1041.

Hendrik Pieter Barendregt (1984), *The lambda calculus*, volume 103 of *Studies in Logic and the Foundations of Mathematics*, North-Holland.

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.

Patrick BLACKBURN (1993), Modal Logic and Attribute Value Structures, in Maarten DE RIJKE, editor, *Diamonds and Defaults*, volume 229 of *Synthese Library*, pp. 19–65, Springer Netherlands, ISBN 978-90-481-4286-6, DOI: 10.1007/978-94-015-8242-1_2.

Patrick BLACKBURN and Maarten MARX (2002), Tableaux for Quantified Hybrid Logic, in Uwe EGLY and Chritian G. FERMÜLLER, editors, *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference, TABLEAUX 2002 Copenhagen, Denmark, July 30 – August 1, 2002 Proceedings*, pp. 38–52, Springer, Berlin, Heidelberg, DOI: 10.1007/3-540-45616-3_4.

Patrick BLACKBURN and Maarten De RIJKE (1997), Zooming In, Zooming Out, *Journal of Logic, Language and Information*, 6(1):5–31, ISSN 0925-8531, DOI: 10.1023/A%3A1008204403391.

Patrick BLACKBURN and Jerry SELIGMAN (1995), Hybrid languages, *Journal of Logic, Language and Information*, 4(3):251–272, DOI: 10.1007/BF01049415.

Johan BOS (1995), Predicate Logic Unplugged, in *Proceedings of the Tenth Amsterdam Colloquium*, `http://www.let.rug.nl/bos/pubs/Bos1996AmCo.pdf`.

Haskell Brooks CURRY (1961), Some Logical Aspects of Grammatical Structure, in Roman JAKOBSON, editor, *Structure of Language and its Mathematical Aspects: Proceedings of the Twelfth Symposium in Applied Mathematics*, pp. 56–68, American Mathematical Society.

Philippe DE GROOTE (2001), Towards Abstract Categorial Grammars, in *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference*, pp. 148–155, ACL anthology: P01-1033.

Philippe DE GROOTE (2002), Tree-Adjoining Grammars as Abstract Categorial Grammars, in *Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+6)*, pp. 145–150, Università di Venezia, `http://www.loria.fr/equipes/calligramme/acg/publications/2002-tag+6.pdf`.

Philippe DE GROOTE (2006), Towards a Montagovian account of dynamics, in Masayuki GIBSON and Jonathan HOWELL, editors, *Proceedings of Semantics and Linguistic Theory (SALT) 16*, DOI: 10.3765/salt.v16i0.2952.

Philippe DE GROOTE and Sylvain POGODALLA (2004), On the expressive power of Abstract Categorial Grammars: Representing context-free formalisms, *Journal of Logic, Language and Information*, 13(4):421–438, DOI: 10.1007/s10849-004-2114-x, HAL open archive: inria-00112956.

Charles J. FILLMORE (1977), The case for case reopened, in Peter COLE and Jerrold M. SADOCK, editors, *Grammatical Relations*, volume 8 of *Syntax and Semantics*, pp. 59–81, Academic Press, New York.

Massimo FRANCESCHET and Maarten DE RIJKE (2006), Model checking hybrid logics (with an application to semistructured data), *Journal of Applied Logic*, 4(3):279–304, DOI: 10.1016/j.jal.2005.06.010.

Daniel GALLIN (1975), *Intensional and Higher-Order Modal Logic*, North-Holland.

Thomas GAMERSCHLAG, Doris GERLAND, Rainer OSSWALD, and Wiebke PETERSEN, editors (2014), *Frames and Concept Types*, volume 94 of *Studies in Linguistics and Philosophy*, Springer International Publishing, DOI: 10.1007/978-3-319-01541-5.

Seohyun IM and Chunngmin LEE (2015), A Developed Analysis of Type Coercion Based on Type Theory and Conventionality, in Robin COOPER and Christian RETORÉ, editors, *Type Theory and Lexical Semantics*, ESSLLI 2015, Barcelona, Spain, `http://www.lirmm.fr/tytles/Articles/Im.pdf`.

Laura KALLMEYER and Rainer OSSWALD (2013), Syntax-Driven Semantic Frame Composition in Lexicalized Tree Adjoining Grammars, *Journal of Language Modelling*, 1(2):267–330, DOI: 10.15398/jlm.v1i2.61.

Laura KALLMEYER, Rainer OSSWALD, and Sylvain POGODALLA (2015), Progression and Iteration in Event Semantics - An LTAG Analysis Using Hybrid Logic and Frame Semantics, in *Colloque de Syntaxe et Sémantique à Paris (CSSP 2015)*, HAL open archive: hal-01184872.

Laura KALLMEYER and Frank RICHTER (2014), Quantifiers in Frame Semantics, in Glyn MORRILL, Reinhard MUSKENS, Rainer OSSWALD, and Frank RICHTER, editors, *Formal Grammar*, volume 8612 of *Lecture Notes in Computer Science*, pp. 69–85, Springer, DOI: 10.1007/978-3-662-44121-3_5.

Laura KALLMEYER and Maribel ROMERO (2008), Scope and Situation Binding for LTAG, *Research on Language and Computation*, 6(1):3–52, DOI: 10.1007/s11168-008-9046-6.

Joachim LAMBEK (1958), The Mathematics of Sentence Structure, *American Mathematical Monthly*, 65(3):154–170.

Ekaterina LEBEDEVA (2012), *Expression de la dynamique du discours à l'aide de continuations*, Ph.D. thesis, Université de Lorraine, in English.

Sebastian LÖBNER (2014a), Evidence for frames from human language, in Gamerschlag *et al.* (2014), chapter 2, pp. 23–67, DOI: 10.1007/978-3-319-01541-5_2.

Sebastian LÖBNER (2014b), Frames and metonymy – Shifting the center and refocusing the frame, Concept Types and Frames in Language, Cognition, and Science (CTF14), invited talk, `http://www.sfb991.uni-duesseldorf.de/fileadmin/Vhosts/SFB991/CTF14Abstr/Loebner_-_AK.pdf`.

Igor A. MEL'ČUK, André CLAS, and Alain POLGUÈRE (1995), *Introduction à la lexicologie explicative et combinatoire*, Éditions Duculot, Louvain-la-Neuve.

Marc MOENS and Mark STEEDMAN (1988), Temporal Ontology and Temporal Reference, *Computational Linguistics*, 14(2):15–28, ACL anthology: J88-2003.

Richard MONTAGUE (1974), The Proper Treatment of Quantification in Ordinary English, in *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, re-edited in "Formal Semantics: The Essential Readings", Paul Portner and Barbara H. Partee, editors. Blackwell Publishers, 2002.

Reinhard MUSKENS (2013), Data Semantics and Linguistic Semantics, in Maria ALONI, Michael FRANKE, and Floris ROELOFSEN, editors, *The dynamic, inquisitive, and visionary life of ϕ, ?ϕ, and $\Diamond\phi$*, chapter 24, pp. 175–183, Pumbo.nl, `http://www.illc.uva.nl/Festschrift-JMF/papers/23_Muskens.pdf`.

Rainer OSSWALD and Robert D. VAN VALIN, Jr. (2014), FrameNet, Frame Structure, and the Syntax-Semantics Interface, in Gamerschlag *et al.* (2014), chapter 6, pp. 125–156, DOI: 10.1007/978-3-319-01541-5_6.

Wiebke PETERSEN (2007), Representation of Concepts as Frames, *The Baltic International Yearbook of Cognition, Logic and Communication*, 2:151–170, `http://user.phil-fak.uni-duesseldorf.de/~petersen/paper/Petersen2007_proof.pdf`.

Sylvain POGODALLA (2004), Computing Semantic Representation: Towards ACG Abstract Terms as Derivation Trees, in *Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms - TAG+7*, pp. 64–71, Vancouver, BC, Canada, HAL open archive: inria-00107768.

Alain POLGUÈRE (2003), *Lexicologie et sémantique lexicale*, Les Presses de l'Université de Montréal.

James PUSTEJOVSKY (1998), *The Generative Lexicon*, MIT Press.

James PUSTEJOVSKY and Pierrette BOUILLON (1995), Aspectual Coercion and Logical Polysemy, *Journal of Semantics*, 12(2):133–162, DOI: 10.1093/jos/12.2.133.

Thomas SCHNEIDER (2007), *The Complexity of Hybrid Logics over Restricted Classes of Frames*, Ph.D. thesis, University of Jena, Germany, `http://www.cs.man.ac.uk/~schneidt/publ/sch07_phd.pdf`.

Balder TEN CATE and Massimo FRANCESCHET (2005), On the Complexity of Hybrid Logics with Binders, in Luke ONG, editor, *Computer Science Logic: 19th International Workshop, CSL 2005, 14th Annual Conference of the EACSL, Oxford, UK, August 22-25, 2005. Proceedings*, pp. 339–354, Springer Berlin Heidelberg, DOI: 10.1007/11538363_24.

Factivity and presupposition in Dependent Type Semantics

Ribeka Tanaka[1], *Koji Mineshima*[1,2], *and Daisuke Bekki*[1,2]
[1] Ochanomizu University
[2] CREST, Japan Science and Technology Agency

ABSTRACT

Dependent type theory has been applied to natural language semantics to provide a formally precise and computationally adequate account of dynamic aspects of meaning. One of the frameworks of natural language semantics based on dependent type theory is Dependent Type Semantics (DTS), which focuses on the compositional interpretations of anaphoric expressions. In this paper, we extend the framework of DTS with a mechanism to handle logical entailment and presupposition associated with factive verbs such as *know*. Using the notion of proof objects as first-class objects, we provide a compositional account of presuppositional inferences triggered by factive verbs. The proposal also gives a formal reconstruction of the type-distinction between propositions and facts, and thereby accounts for the lexical semantic differences between factive and non-factive verbs in a type-theoretical setting.

Keywords: *dependent type, anaphora, presupposition, proof object, factive verb*

1 INTRODUCTION

Dependent Type Semantics (DTS, Bekki 2014) is a framework of natural language semantics based on dependent type theory (Martin-Löf 1984; Nordström *et al.* 1990). In contrast to traditional model-theoretic semantics, DTS is a proof-theoretic semantics, where inference relations between sentences are characterized as provability relations between semantic representations. One of the distinctive features of DTS, as compared to other type-theoretical frameworks, is that it is augmented with underspecified terms, so as to provide

a unified analysis of inference, anaphora and presupposition from a logical/computational perspective. In contrast to previous work on anaphora in dependent type theory (cf. Ranta 1994), DTS gives a fully compositional account of inferences involving anaphora. It is also extended to the analysis of modal subordination (Tanaka *et al.* 2015).

In this paper, we provide the framework of DTS with a mechanism to handle logical entailment and presupposition associated with factive verbs. We will mostly focus on the epistemic verb *know*. Although there are numerous studies on factive verbs in natural language semantics, they are usually based on model-theoretic approaches; it seems fair to say that there has been little attempt to formalize inferences with factivity from a proof-theoretical perspective. On the other hand, various proof systems for knowledge and belief have been studied in the context of epistemic logic (cf. Meyer and van der Hoek 2004). However, such systems are mainly concerned with knowledge and belief themselves, not with how they are expressed in natural languages, nor with linguistic phenomena such as factivity presuppositions. Our study aims to fill this gap by providing a framework that explains logical entailment and presuppositions with factive verbs in dependent type theory.

2 DEPENDENT TYPE SEMANTICS

This section introduces the framework of DTS and explains how presuppositions are handled in this framework. In Section 2.1, we provide some necessary background on DTS, including the basics of dependent type theory and the analysis of anaphora within this approach. One of the important problems in the application of dependent type theory to natural language semantics is how to represent common nouns using the machinery of dependent types. Section 2.2 is devoted to discussing this problem. We give several reasons for preferring the view that common nouns are represented as *predicates* rather than as *types*. Given this background, Section 2.3 provides a compositional analysis of presupposition in DTS.

2.1 *Dependent type theory*

In dependent type theory, there are two type constructors, Σ and Π, which play a crucial role in forming the semantic representations for

natural language sentences. The type constructor Σ is a generalized form of the product type and behaves as an existential quantifier. An object of type $(\Sigma x : A)B(x)$ is a pair (m, n) such that m is of type A and n is of type $B(m)$. Conjunction $A \wedge B$ is a degenerate form of $(\Sigma x : A)B$ if x does not occur free in B. The Σ-types are associated with projection functions π_1 and π_2 that are computed with the rules $\pi_1(m, n) = m$ and $\pi_2(m, n) = n$, respectively. The type constructor Π is a generalized form of the functional type and behaves as a universal quantifier. An object of type $(\Pi x : A)B(x)$ is a function f such that for any object a of type A, $f\,a$ is an object of type $B(a)$. Implication $A \to B$ is a degenerate form of $(\Pi x : A)B$ if x does not occur free in B. The inference rules for Π-types and Σ-types are shown in the Appendix.[1] Throughout the paper, we will make use of the DTS-notation for Π-types and Σ-types as shown in Figure 1.

	Π-types	Σ-types
Standard notation	$(\Pi x : A)B(x)$	$(\Sigma x : A)B(x)$
Notation in DTS	$(x{:}A) \to B(x)$	$\left[\begin{array}{c} x : A \\ B(x) \end{array}\right]$
When $x \notin fv(B)$	$A \to B$	$\left[\begin{array}{c} A \\ B \end{array}\right]$

Figure 1:
Notation for Π-types and Σ-types in DTS ($fv(B)$ means the set of free variables in B)

Based on the Curry-Howard correspondence (Howard 1980), a type can be regarded as a proposition and a term can be regarded as a proof. Thus the judgement $a : A$ can be read as "a is a proof of proposition A", as well as "a is a term of type A". In this setting, the truth of a proposition A is defined as the existence of a term of type A. The term that serves as a proof of a proposition is called a *proof term* and plays an important role in representing natural language sentences in dependent type theory.

Since the work of Sundholm (1986) and Ranta (1994), dependent type theory has been applied to the analysis of various dynamic discourse phenomena, providing a type-theoretic alternative to model-theoretic frameworks such as Discourse Representation Theory (van der Sandt 1992; Kamp *et al.* 2011), Dynamic Predicate Logic (Groenendijk and Stokhof 1991), and Dynamic Semantics (Heim

[1] For more details, readers can refer to Martin-Löf (1984) and Ranta (1994).

1983). For instance, according to the analysis presented in Sundholm (1986) and Ranta (1994), a semantic representation for the donkey sentence in (1) can be given as (2) in terms of dependent types.

(1) Every farmer who owns a donkey beats it.

(2) $$\left(u{:} \left[\begin{array}{l} x : \mathbf{farmer} \\ \left[\begin{array}{l} y : \mathbf{donkey} \\ \mathbf{own}(x, y) \end{array} \right] \end{array} \right] \right) \rightarrow \mathbf{beat}(\pi_1 u, \pi_1 \pi_2 u)$$

The sentence (1) as a whole is a universal sentence, which is represented as a Π-type. The restrictor *farmer who owns a donkey* is analyzed as a Σ-type. A term u having this Σ-type would be a tuple $(f, (d, o))$, where f is a term of type **farmer**, d is a term of type **donkey**, and o is a proof-term of the proposition $\mathbf{own}(f, d)$. Recall that π_1 and π_2 are projection functions that take a pair and return the first and the second element, respectively. Thus the terms $\pi_1 u$ and $\pi_1 \pi_2 u$ appearing in the consequent of (2) pick up from u the term f of type **farmer** and the term d of type **donkey**, respectively. In this way, via the proof term u associated with the Σ-type, the discourse referents introduced in the antecedent in (2) can be successfully passed to the subsequent discourse. An advantage of dependent type theory over previous dynamic theories is that such an externally dynamic character of quantification can be captured without any further stipulation; Σ-types and Π-types, which are natural generalizations of existential and universal quantifiers in predicate logic, are equipped with the mechanism to handle dynamic aspects of discourse interpretations.

The work by Sundholm and Ranta[2] provides a foundation for applying the expressiveness of dependent types to problems in natural language discourse interpretation such as donkey anaphora. However, a problem remains: how can the semantic representation in (2) be systematically obtained from the sentence in (1)? From the viewpoint of standard compositional semantics, the problem can be divided into two tasks. The first is to deterministically map the sentence (1) into an

[2] Sundholm (1986) and Ranta (1994) only consider the so-called strong reading of donkey sentences. There are some later works using dependent types that treat other phenomena discussed in the dynamic semantics literature, in particular, Sundholm (1989) for the proportion problem. See also Tanaka *et al.* (2014) and Tanaka (2014) for discussion of Sundholm's analysis of donkey anaphora and treatment of weak and strong readings within the framework of DTS.

underspecified representation, a semantic representation that contains an underspecified element corresponding to the pronoun in question. The second task is to resolve anaphora. In our example, the underspecified element needs to be resolved to $\pi_1\pi_2 u$. The semantics satisfying the requirement of compositionality must provide an explicit procedure for these two tasks.

Dependent Type Semantics (Bekki 2014) provides such a procedure. To give an explicit compositional mapping from sentences to semantic representations, we adopt Combinatory Categorial Grammar (CCG, Steedman 2000) as a syntactic framework. Note that, as emphasized in Bekki (2014), DTS can be combined with other categorial grammars; see Kubota and Levine (2017) for a concrete proposal that combines DTS with a type-logical grammar. In compositional mapping, an anaphoric expression is mapped on to an underspecified element. The process of resolving underspecification is formulated as the process of *type checking*. Using the machinery of underspecified semantics in DTS, we will give an analysis of presuppositions in Section 2.3.

2.2 *Common nouns: types or predicates?*

There are two possible approaches to representing basic sentences like *A man entered* in dependent type theory. One is the approach proposed in Ranta (1994) and Luo (2012a,b), according to which common nouns like *man* are interpreted as *types* so that the sentence is represented as (3) in our notation.

(3) $\left[\begin{array}{l} x : \mathbf{man} \\ \mathbf{enter}(x) \end{array}\right]$

One problem with this approach is that it is not straightforward to analyze *predicational* sentences, i.e., sentences containing *predicate nominals*, such as (4a, b).[3]

(4) a. John is *a man*.
b. Bob considers Mary *a genius*.

One might analyze (4a) as a judgement **john** : **man**. However, a judgement itself can neither be negated nor embedded under a log-

[3] See Mikkelsen (2011) for a useful overview of the syntax and semantics of predicational sentences.

ical operator. Accordingly, it is not clear how to account for the fact that a predicational sentence can be negated, as in (5a), or appear in the antecedent of a conditional, as in (5b).

(5) a. John is not a man.

b. If John is a man, ….

Nor is it clear how to analyze a construction embedding a predicational sentence as in (4b).

One might try to analyze *be*-verbs as the so-called "*is*-of identity" along Russell-Montague lines (Russell 1919; Montague 1973). This enables us to represent (4a) as a *proposition*, as in (6), rather than as a judgement.

(6) $\left[\begin{array}{l} x : \mathbf{man} \\ \mathbf{john} =_{\mathbf{man}} x \end{array}\right]$

Then, (5a) and (5b) can be represented as follows:

(7) a. $\neg\left[\begin{array}{l} x : \mathbf{man} \\ \mathbf{john} =_{\mathbf{man}} x \end{array}\right]$

b. $\left[\begin{array}{l} x : \mathbf{man} \\ \mathbf{john} =_{\mathbf{man}} x \end{array}\right] \rightarrow \cdots$

There are two problems with this approach, however. First, this analysis predicts that the predicate nominal *a man* introduces a discourse referent in terms of Σ-types. Contrary to this prediction, a predicate nominal cannot serve as an antecedent of an anaphoric pronoun such as *he* or *she* (Kuno 1970; Mikkelsen 2005); hence it does not introduce an individual discourse referent.[4]

The second problem is the interpretation of equality. In dependent type theory, equality is relativised to some type A and the formation rule requires the arguments of equality symbols to have type A:

[4] The form of the pronoun anaphoric on a predicate nominal in (i) must be *it*, rather than *him*; the relative pronoun in (ii) must be *which*, not *who* (Kuno 1970; Mikkelsen 2005).

(i) He is a fool, although he doesn't look { it /*him }.

(ii) He is a gentleman, {which / *who} his brother is not.

See Fara (2001) for more discussion of the problems of the Russell-Montague analysis of predicate nominals.

(8) $$\frac{A : \mathbf{type} \quad t : A \quad u : A}{t =_A u : \mathbf{type}} \; =F$$

Accordingly, the proposition $\mathbf{john} =_{\mathbf{man}} x$ is well-formed only if $\mathbf{john} : \mathbf{man}$ is provable. This is also the case if the proposition is embedded under a logical operator. It thus follows that under the Russell-Montague analysis combined with the equality rule (8), not only the positive sentence (6), but also the negation (7a) and the conditional (7b) presuppose that John is a man. To rescue the common-nouns-as-types view from this problem, one has to provide a more complex analysis of logical operators such as negation and implication.[5] However, the resulting theory would then become more complicated.

As an alternative approach, we interpret a common noun as a predicate. Common nouns in argument position and in predicate position are both analyzed as predicates of type $\mathbf{entity} \to \mathbf{type}$.

(9) A man walks. $$\left[\begin{array}{l} u : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{man}(x) \end{array} \right] \\ \mathbf{walk}(\pi_1 u) \end{array} \right]$$

(10) John is a man. $\mathbf{man}(\mathbf{john})$

This approach is in line with the traditional analysis of common nouns, so we can integrate standard assumptions in formal semantics into our framework. Moreover, since predicates do not introduce discourse referents, we can explain the impossibility of referential anaphora to predicate nominals.

Retoré (2014) suggests that common nouns can be interpreted both as types and as predicates; for instance, using type **entity**, the common noun *animal* interpreted as a type **animal** could be related to a predicate $\mathbf{animal}^*$ of type $\mathbf{entity} \to \mathbf{type}$, via some suitably defined mapping $(\cdot)^*$ from one to another. The question of whether a type system for natural language semantics needs to be enriched with the structures of common nouns would ultimately depend on the treatment of the lexical semantic phenomena it attempts to capture, such as coercion and selectional restriction – phenomena that have been widely discussed in the recent literature on type-theoretical semantics (Asher 2011; Asher and Luo 2012; Bekki and Asher 2013; Retoré

[5] Some discussion of the treatment of negation in the context of dependent type theory can be found in Chatzikyriakidis and Luo (2014).

2014; Kinoshita *et al.* 2016). But investigating this matter further is beyond the scope of the present paper and we leave it to a future study.

2.3 *Analysis of presupposition in DTS*

To handle anaphora and presupposition in a compositional setting, DTS extends dependent type theory with a mechanism of context passing and underspecified terms.

Dependent Type Semantics distinguishes two kinds of propositions: *static* and *dynamic* propositions. Following the Curry-Howard correspondence, we call an object of type **type** (i.e., the type of types) a *static* proposition. A *dynamic* proposition is a function which maps a proof term of the static proposition representing the preceding discourse to a static proposition. The basic idea is that for each (static) proposition P, the information obtained up to that point is passed to P as a proof term. Such a proof term is called a *local context*.

Dependent Type Semantics extends the syntax of dependent type theory with an underspecified term $@_i$, which is used to represent anaphora and presupposition triggers.[6] We show how it can provide a compositional account of anaphora and presupposition. We take the existence presupposition triggered by a definite description as a representative example. Consider the following example.

(11) The book arrived.

The definite description *the book* here triggers the presupposition that there is a book.[7] We analyze the determiner *the* appearing in the subject position as having the CCG category $(S/(S\backslash NP))/N$, and give a semantic representation by using an underspecified term. The lexical

[6] Bekki (2014) provides an overview and comparison of previous approaches to representing underspecification in the context of dependent type theory (Dávila-Pérez 1995; Krahmer and Piwek 1999; Piwek and Krahmer 2000).

[7] Here we take it that the uniqueness presupposition is not part of the conventional meaning of a definite description but can be derived on pragmatic considerations along the lines of Heim (1982). Although it is technically possible to take the uniqueness implication as part of presupposition, the proof-search procedure to find the antecedent of an underspecified term would then become much more complicated.

$N^\bullet = \mathbf{entity} \to \delta \to \mathbf{type}$	$N^\bullet_{+c} = \mathbf{type} \to \delta \to \mathbf{type}$
$NP^\bullet = \mathbf{entity}$	$NP^\bullet_{+c} = \delta \to \mathbf{type}$
$S^\bullet = \delta \to \mathbf{type}$	$\overline{S}^\bullet = \delta \to \mathbf{type}$
$(C_1/C_2)^\bullet = (C_1\backslash C_2)^\bullet = C_2^\bullet \to C_1^\bullet$	

Figure 2: Mapping syntactic categories to semantic types[9]

entry for *the* can be specified as follows (a mapping $(\cdot)^\bullet$ from syntactic categories to semantic types can be defined as in Figure 2).[8]

(12) the; $(S/(S\backslash NP))/N$; $\lambda n.\lambda v.\lambda c.\, v\left(\pi_1\left(@_i c :: \left[\begin{array}{l} x : \mathbf{entity} \\ nxc \end{array}\right]\right)\right)c$

Determiner *the* denotes a function that takes a predicate n denoted by a restrictor and a predicate v denoted by a verb and returns a dynamic proposition, which is in turn a function from a local context c to a (static) proposition. The local context c is passed to the underspecified term $@_i$ as an argument. It is also sent to the predicates n and v as an extra argument, because n and v may contain underspecified terms.

The form $M :: A$ is called *type annotation* and specifies that the term M has type A. When an underspecified term $@_i$ is annotated with a type A, that is, when we have $@_i :: A$, the annotated type A represents the presupposition triggered by this underspecified term. In (12), the underspecified term with a local context, $@_i c$, is annotated with a Σ-type. This means that the underspecified term $@_i$ is a function that takes a local context c as an argument and returns a term having the annotated Σ-type. Given this type annotation, we see that $@_i c$ is a pair of an entity x and a proof term for the proposition that x satisfies

[8] For the purpose of concreteness, we use a type-raised form of semantic representations for determiners. The entry for the determiner *the* in the object position can be given as follows:

the; $((S\backslash NP)\backslash((S\backslash NP)/NP))/N$; $\lambda n.\lambda v.\lambda x.\lambda c.\, v\left(\pi_1\left(@_i c :: \left[\begin{array}{l} y : \mathbf{entity} \\ nyc \end{array}\right]\right)\right)xc$

Although there are other possible syntactic analyses of determiners in object position (cf. Bekki 2014), this entry would ensure a concise derivation tree for semantic composition.

[9] Subscripted $+c$ is a syntactic feature for content noun. We represent N_{-c} and NP_{-c} simply as N and NP, respectively. We also abbreviate other syntactic features, which are not relevant to the discussion in this paper.

the predicate n given a local context c. In other words, the annotated type represents the existence presupposition triggered by the definite article. What is applied to the main predicate appearing in v is the first projection of the obtained pair, i.e., a term of type **entity**.

The semantic representation for (11) is derived as follows.[10]

(13)

$$\dfrac{\dfrac{\dfrac{\text{The}}{\begin{array}{c}(S/(S\backslash NP))/N\\ \lambda n.\lambda v.\lambda c.\, v\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \textbf{entity} \\ nxc \end{array}\right]\right)\right)c\end{array}} \quad \dfrac{\text{book}}{\begin{array}{c}N\\ \lambda x.\lambda c.\, \textbf{book}(x)\end{array}}}{\begin{array}{c}S/(S\backslash NP)\\ \lambda v.\lambda c.\, v\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right]\right)\right)c\end{array}}\,> \quad \dfrac{\text{arrived}}{\begin{array}{c}S\backslash NP\\ \lambda x.\lambda c.\, \textbf{arrive}(x)\end{array}}}{\begin{array}{c}S\\ \lambda c.\, \textbf{arrive}\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right]\right)\right)\end{array}}\,>$$

The underspecified term is indexed by a natural number i and each number assigned to an underspecified term is mutually distinct. In the above derivation, an underspecified term introduced by *the* has index 1. Here we assume that the predicate **book** is not context-sensitive so that vacuous abstraction is involved as in $\lambda x \lambda c.\textbf{book}(x)$; in such a case, the input local context c is simply discarded in the body of the semantic representation. As shown here, the term $@_1 c$ in the final representation is annotated with a Σ-type corresponding to the proposition that there is a book. In this way, the annotated type represents the existence presupposition triggered by the definite description *the book*.

[10] In CCG derivation trees, we use two standard combinatory rules: forward (>) and backward (<) function application rules.

$$\frac{X/Y : m \quad Y : n}{X : mn}\,> \qquad \frac{Y : n \quad X\backslash Y : m}{X : mn}\,<$$

For instance, the combinatory rule (>) means that an expression having a syntactic category X/Y and a meaning m, combined with an expression having a syntactic category Y and a meaning n, yields an expression having a category X and a meaning mn. Each meaning is represented as a lambda term. See Steedman (2000) for more details.

The resolution of an underspecified term in a semantic representation A amounts to checking that A is well-typed in a given context. More specifically, it is triggered by the following:

(14) $\Gamma, \delta : \mathbf{type} \vdash A : \delta \to \mathbf{type}$

Here, Γ is a set of assumptions, called a *global* context, which represents the background knowledge; δ is the type of a local context (representing the previous discourse); $A : \delta \to \mathbf{type}$ in the consequence shows that A is a dynamic proposition in DTS, that is, a function mapping a given local context of type δ to a static proposition; (14) reflects the requirement that the semantic representation of a (declarative) sentence, i.e., a static proposition, must be of type **type**. This requirement is called the *felicity condition* of a sentence in DTS.

In the case of (11), the resolution process is launched by the following judgement:

(15) $\Gamma, \delta : \mathbf{type} \vdash \lambda c.\, \mathbf{arrive}\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{book}(x) \end{array}\right]\right)\right) : \delta \to \mathbf{type}.$

Assuming that $\mathbf{arrive} : \mathbf{entity} \to \mathbf{type}$ is in the global context Γ, the type of $@_1$ is determined by the following derivation.[11]

(16)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{@_1 : \delta \to \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{book}(x) \end{array}\right] \quad \dfrac{}{c : \delta}\,1}{@_1 c : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{book}(x) \end{array}\right]}\,(\Pi E)}{\left(@_1 c :: \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{book}(x) \end{array}\right]\right) : \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{book}(x) \end{array}\right]}\,(ann)}{\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{book}(x) \end{array}\right]\right) : \mathbf{entity}}\,(\Sigma E) \qquad \dfrac{}{\mathbf{arrive} : \mathbf{entity} \to \mathbf{type}}\,(CON)}{\mathbf{arrive}\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{book}(x) \end{array}\right]\right)\right) : \mathbf{type}}\,(\Pi E)}{\lambda c.\, \mathbf{arrive}\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{book}(x) \end{array}\right]\right)\right) : \delta \to \mathbf{type}}\,(\Pi I), 1$$

[11] Here we make use of the following rule, which ensures that one can obtain the annotated term $t :: A$ of type A from any term $t : A$.

$$\frac{t : A}{(t :: A) : A}\,(ann)$$

See also the Appendix for other derivation rules.

The open branch of the derivation, repeated in (17), requires that in order for the semantic representation in question to be well-typed, one has to construct a proof term for the proposition that there is a book. (This is due to the @-formation rule. See Definition 12 in the Appendix.)

(17) $@_1 : \delta \rightarrow \left[\begin{array}{l} x : \mathbf{entity} \\ \mathbf{book}(x) \end{array} \right]$

In other words, if one assumes that (11) is a felicitous utterance, the proposition that there is a book must be true. This requirement corresponds to the existence presupposition of (11).

At the final stage of presupposition resolution, a proof search is carried out to prove (17) and the underspecified term $@_1$ is replaced by the constructed term. More specifically, the process of anaphora/presupposition resolution is defined as follows (Bekki 2014).

(18) Suppose that $\Gamma \vdash @_i : A$ and $\Gamma \vdash M : A$, where Γ is a global context and A is a type. Then a resolution of $@_i$ by M under the context Γ is an equation $@_i =_A M$.

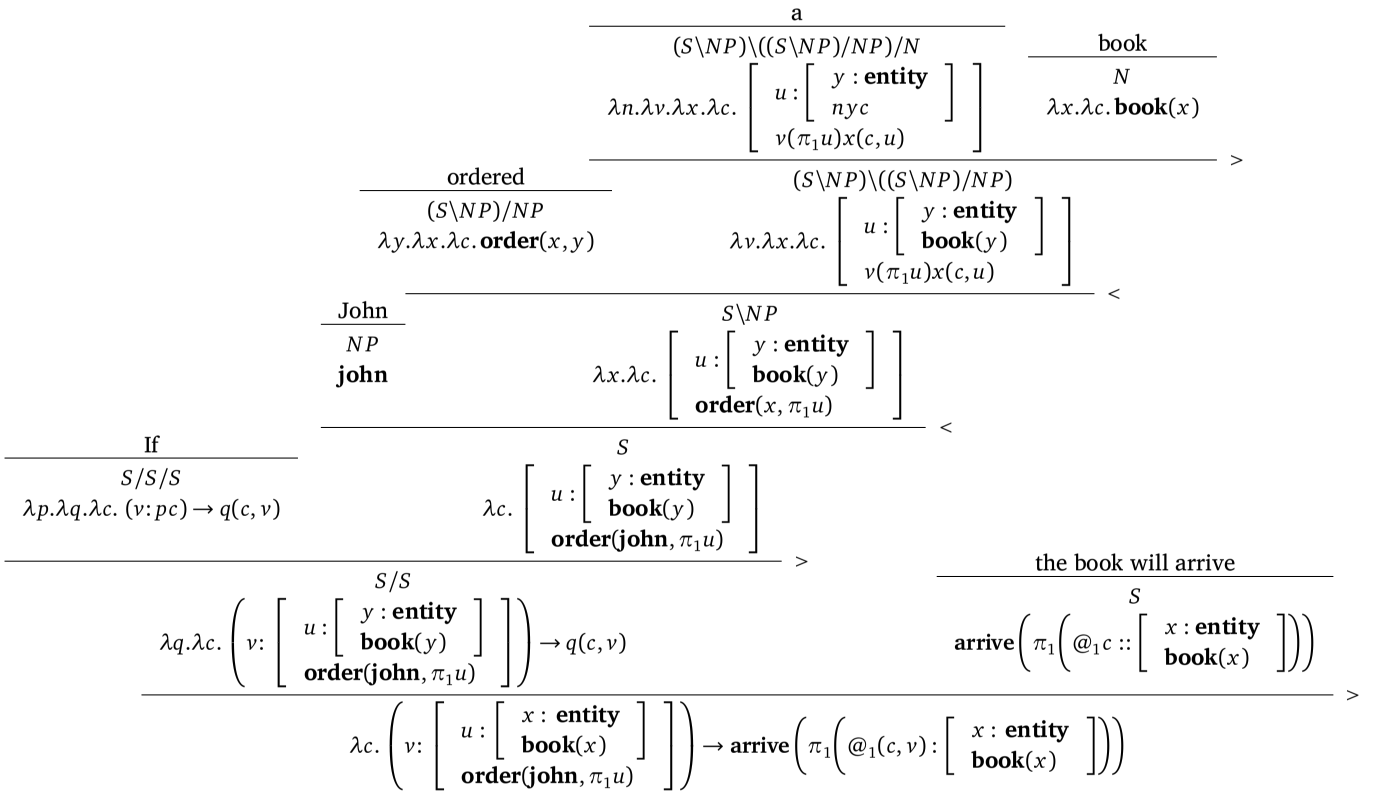
In the example considered here, if a proof term for the presupposition that there is a book is constructed, it can replace the underspecified term $@_1$. Such a proof construction is possible when, for instance, *the book* appears in contexts as shown in (19a, b).

(19) a. If John ordered a book last week, the book will arrive today.
b. John ordered a book last week and the book arrived today.

In general, if S' entails the presuppositions of S, constructions such as S' and S and If S' then S do not inherit the presuppositions of S. In such a case, it is said that the presupposition is *filtered*.

In DTS, examples such as (19a, b) can be handled in the following way. First, the (somewhat simplified) semantic representation for (19a) is derived as shown in (20). The type checking derivation for the final representation of (20) is shown in (21). This derivation specifies the type of $@_1$. We can see that what is required for the representation to be well-typed is to find a term substituted for $@_1$ in (22).

(20)


a
$(S\backslash NP)\backslash((S\backslash NP)/NP)/N$
$\lambda n.\lambda v.\lambda x.\lambda c.\left[\begin{array}{l} u:\left[\begin{array}{l} y:\textbf{entity} \\ nyc \end{array}\right] \\ v(\pi_1 u)x(c,u) \end{array}\right]$
book
N
$\lambda x.\lambda c.\textbf{book}(x)$
>
ordered
$(S\backslash NP)/NP$
$\lambda y.\lambda x.\lambda c.\textbf{order}(x,y)$
$(S\backslash NP)\backslash((S\backslash NP)/NP)$
$\lambda v.\lambda x.\lambda c.\left[\begin{array}{l} u:\left[\begin{array}{l} y:\textbf{entity} \\ \textbf{book}(y) \end{array}\right] \\ v(\pi_1 u)x(c,u) \end{array}\right]$
<
John
NP
$\textbf{john}$
$S\backslash NP$
$\lambda x.\lambda c.\left[\begin{array}{l} u:\left[\begin{array}{l} y:\textbf{entity} \\ \textbf{book}(y) \end{array}\right] \\ \textbf{order}(x,\pi_1 u) \end{array}\right]$
<
If
$S/S/S$
$\lambda p.\lambda q.\lambda c.\ (v{:}\,pc) \to q(c,v)$
S
$\lambda c.\left[\begin{array}{l} u:\left[\begin{array}{l} y:\textbf{entity} \\ \textbf{book}(y) \end{array}\right] \\ \textbf{order}(\textbf{john},\pi_1 u) \end{array}\right]$
>
the book will arrive
S
$\textbf{arrive}\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x:\textbf{entity} \\ \textbf{book}(x) \end{array}\right]\right)\right)$
S/S
$\lambda q.\lambda c.\left(v{:}\left[\begin{array}{l} u:\left[\begin{array}{l} y:\textbf{entity} \\ \textbf{book}(y) \end{array}\right] \\ \textbf{order}(\textbf{john},\pi_1 u) \end{array}\right]\right) \to q(c,v)$
>
$\lambda c.\left(v{:}\left[\begin{array}{l} u:\left[\begin{array}{l} x:\textbf{entity} \\ \textbf{book}(x) \end{array}\right] \\ \textbf{order}(\textbf{john},\pi_1 u) \end{array}\right]\right) \to \textbf{arrive}\left(\pi_1\left(@_1(c,v):\left[\begin{array}{l} x:\textbf{entity} \\ \textbf{book}(x) \end{array}\right]\right)\right)$

(21)

$$
\dfrac{
\begin{matrix} \vdots \\ \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix} : \mathbf{type} \end{matrix}
\qquad
\dfrac{
\dfrac{}{\mathbf{arrive} : \mathbf{entity} \to \mathbf{type}}
\qquad
\dfrac{
\dfrac{
\dfrac{
@_1 : \begin{bmatrix} \delta \\ \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix} \end{bmatrix} \to \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}
\qquad
\dfrac{
\dfrac{}{c : \delta}\,2
\qquad
\dfrac{}{t : \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix}}\,1
}{
(c, t) : \begin{bmatrix} \delta \\ \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix} \end{bmatrix}
}\,(\Sigma I)
}{
@_1(c, t) : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}
}\,(\Pi E)
}{
\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right) : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}
}\,(ann)
}{
\pi_1\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right) : \mathbf{entity}
}\,(\Sigma E)
}{
\mathbf{arrive}\left(\pi_1\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right)\right) : \mathbf{type}
}\,(\Pi E)
}{
\dfrac{
\left(t : \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix}\right) \to \mathbf{arrive}\left(\pi_1\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right)\right) : \mathbf{type}
}{
\lambda c. \left(t : \begin{bmatrix} u : \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix} \\ \mathbf{order}(\mathbf{john}, \pi_1 u) \end{bmatrix}\right) \to \mathbf{arrive}\left(\pi_1\left(@_1(c, t) :: \begin{bmatrix} x : \mathbf{entity} \\ \mathbf{book}(x) \end{bmatrix}\right)\right) : \delta \to \mathbf{type}
}\,(\Pi I), 2
}\,(\Pi F), 1
$$

$$(22)\quad @_1 : \left[\begin{array}{l} \delta \\ \left[\begin{array}{l} u : \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right] \\ \textbf{order}(\textbf{john}, \pi_1 u) \end{array}\right] \end{array}\right] \rightarrow \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right]$$

In this case, without using the information provided in the previous discourse in δ, a proof of the proposition that there is a book can be obtained from the antecedent of the conditional; one can find a term that can replace $@_1$, namely, $\lambda c.\pi_1\pi_2 c$.[12] By replacing $@_1$ with the constructed term $\lambda c.\pi_1\pi_2 c$ in the representation given in (20), we can eventually obtain the following semantic representation for (19a), which captures the intended reading.

$$(23)\quad \left(t : \left[\begin{array}{l} u : \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right] \\ \textbf{order}(\textbf{john}, \pi_1 u) \end{array}\right]\right) \rightarrow \textbf{arrive}(\pi_1\pi_1 t)$$

Another well-known characteristic property of a presupposition is that it *projects* out of embedded contexts such as negation and the antecedent of a conditional. Thus, not only the positive sentence (11) but also the negated sentence (24a) and the antecedent of a conditional (24b) imply that there is a book.

(24) a. The book didn't arrive. NEGATION

b. If the book arrives, Susan will be happy. CONDITIONAL

In DTS, (24a, b) can be given the following semantic representations.

$$(25)\quad \text{a.}\quad \lambda c.\neg\textbf{arrive}\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right]\right)\right)$$

$$\text{b.}\quad \lambda c.\,\textbf{arrive}\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \textbf{entity} \\ \textbf{book}(x) \end{array}\right]\right)\right) \rightarrow \textbf{happy}(\textbf{susan})$$

It can be shown that for the semantic representations (25a) and (25b) to be well-typed, it is required to find a proof term for the proposition that there is a book. Thus, in order to prove that (25b) has type $\delta \rightarrow \textbf{type}$, one has to prove that the antecedent is of type **type** under the given local context c. Since the antecedent in (25b) corresponds to the proposition that the book arrived, this yields the derivation that

[12] When such a filtration does not occur, the entire sentence can have a presupposition that is resolved by the information in δ (i.e., the information in the previous discourse) or by the information in the global context (background knowledge).

contains the type checking process in (16) as a sub-derivation. Accordingly, it is correctly predicted that (24b) has the same existence presupposition as the simple sentence in (11). Note that, in dependent type theory, the negation $\neg A$ is defined using the implication $A \to \bot$, which in turn is a degenerate form of a Π-type. Thus, the same explanation applies to the case of negation in (24a) as well. In this way, we can explain basic projection patterns of presuppositions within the framework of DTS.[13]

Before moving on to the case of factive presuppositions, let us mention one feature of underspecified terms in DTS; that is, an underspecified term can occur inside a type annotation of another underspecified term. This feature enables us to handle *nested* presuppositions. As a typical example, consider a definite noun phrase such as *the book that he wrote*. Omitting the details of compositional derivation, we can assign the following semantic representation to this complex NP.

$$(26)\quad \lambda v.\lambda c.\, v\left(\pi_1\left(@_1 c :: \left[\begin{array}{l} x : \mathbf{entity} \\ \left[\begin{array}{l} \mathbf{book}(x) \\ \mathbf{write}(@_2 c :: \mathbf{entity}, x) \end{array}\right] \end{array}\right]\right)\right) c$$

Here, the underspecified term $@_2$, introduced by the pronoun *he*, occurs inside the type annotation of the underspecified term $@_1$ introduced by *the*.[14] In this case, one can resolve the most embedded underspecified term $@_2$ first and then resolve the outer underspecified term $@_2$ subsequently. A more detailed discussion of nested presupposition is given in Bekki and Mineshima (2017).

3 ANALYZING FACTIVITY IN DTS

In this section, we provide an analysis of factive predicates in DTS. We take the verb *know* as a representative of a factive predicate and provide its semantic representation. We start by summarizing some semantic properties of *know*, in comparison with the non-factive verb *believe*.

[13] Bekki and Satoh (2015) provide a definition for decidable fragment of dependent type theory with an underspecified term and formulate its type-checking algorithm. They also provide an implementation of the algorithm.

[14] It is also possible to add gender information associated with personal pronouns as a presupposition. See Bekki and Mineshima (2017).

3.1 *Inferences with factive and non-factive verbs*

The factive verb *know* and the non-factive verb *believe* show different inference patterns with respect to the form of complements they take. In what follows, we will focus on two types of complements, declarative complements and NP-complements.

Consider the examples in (27), where the verbs *know* and *believe* take a declarative complement.

(27) a. John knows that Mary is successful.
b. John believes that Mary is successful.

(28) Mary is successful.

(27a) implies (28), while (27b) does not. In the context of epistemic logic (Hintikka 1962; Meyer and van der Hoek 2004), the inference from (27a) to (28) has usually been treated as an instance of entailment (hereafter, we use the notion "entailment" in the sense of logical entailment). In the linguistics literature, by contrast, it has been widely agreed that the inference from (27a) to (28) is not an entailment but a presupposition (Kiparsky and Kiparsky 1970; Beaver 2001), as witness examples in (29) and (30).

(29) a. John does not know that Mary is successful. NEGATION
b. If John knows that Mary is successful, ... CONDITIONAL

(30) a. If Mary is successful, John knows that she is.
b. Mary is successful, and John knows that she is.

The examples in (29a, b) show that the proposition in (28) projects out of the embedded contexts; the examples in (30a, b) shows the filtering of presupposition. Because the antecedent of (30a) or the first conjunct of (30b) entails (28), the sentences do not inherit the presupposition of *know*, in a similar way to (19a, b).

Another interesting difference between *know* and *believe* is shown in (31), where they take an NP-complement of the form *the N that P*.

(31) a. John believes the rumor that Mary came.
$\Rightarrow$ John believes that Mary came.
b. John knows the rumor that Mary came.
$\not\Rightarrow$ John knows that Mary came.

K1	x knows that P	$\rhd$	P
K2	x knows the N that P	$\not\Rightarrow$	x knows that P
K3	x knows the N that P	$\rhd$	There is a N that P
B1	x believes that P	$\not\Rightarrow$	P
B2	x believes the N that P	$\Rightarrow$	x believes that P
B3	x believes the N that P	$\rhd$	There is a N that P

Figure 3: Entailments ($\Rightarrow$) and presuppositions ($\rhd$) associated with factive and non-factive verbs (N refers to a non-veridical content noun)

The non-factive verb *believe* licenses the inference from *x Vs the N that P* to *x Vs that P*, where N is a (non-veridical) content noun, such as *rumor*, *story*, and *hypothesis*, that takes a propositional complement; by contrast, the factive verb *know* does not license this pattern of inference (Vendler 1972; Ginzburg 1995a,b; Uegaki 2016).

Figure 3 shows a summary of the inference patterns for *know* and *believe* that we are concerned with in this paper. A remark is in order regarding the non-entailment in K2. There is a class of content nouns that does not follow the pattern in K2. A typical example is the content noun *fact*; "x knows the fact that P" entails "x knows that P", and vice versa. We call this class of nouns *veridical* content nouns and distinguish them from *non-veridical* content nouns such as *rumor* and *story*. The inference pattern in K2 only applies to non-veridical content nouns. We discuss the case of veridical content nouns at the end of Section 3.3.

To predict these inference patterns in a compositional setting, one needs to provide an adequate account of the lexical semantic difference between factive and non-factive verbs. One possible approach is to consider the two types of verbs select for different semantic objects. More specifically, it has been proposed by a number of authors that the non-factive verb *believe* selects for a *proposition*, whereas the factive verb *know* selects for a *fact* (Vendler 1972; Parsons 1993; Ginzburg 1995a,b; King 2002). In the next section, we will explore such a semantic analysis of factive and non-factive verbs within the framework of DTS.

3.2 *Declarative complements*

We treat factive and non-factive verbs as predicates having different semantic types. We analyze the non-factive verb *believe* as taking two

arguments, a term of type **entity** and a proposition. In our notation, the predicate **believe** has the following type:[15]

(32) **believe** : **entity** → **type** → **type**

By contrast, we analyze the factive verb *know* as taking three arguments: (i) an entity representing the agent, (ii) a proposition that serves as the content of knowledge, and (iii) a proof term of that proposition. The predicate **know** has the following semantic type:

(33) **know** : **entity** → (P: **type**) → P → **type**.

As mentioned in Section 2.1, the existence of a proof term a of type P corresponds to the truth of proposition P. One may read **know**$(x)(\mathsf{P})(a)$ as *the agent* x *obtains evidence* a *of the proposition* P.

The standard analysis of *know* in formal semantics follows Hintikka's (1969) possible world semantics, which fails to capture the notion of evidence or justification that has been traditionally associated with the concept of knowledge. An advantage of dependent type theory is that it is equipped with proofs as first-class objects and thus enables us to analyze the factive verb *know* as a predicate over a proof (evidence) of a proposition. Our analysis is also compatible with Vendler's view that *know* and *believe* select for different semantic objects. Note that, in our approach, the notion of facts is not taken as primitive but analyzed in terms of the notion of evidence of a proposition.

The idea that a proof term of a proposition serves as an antecedent of anaphor can be traced back to Ranta (1994), where under the assumption that proofs are identified with *events* it is claimed that aspectual verbs like *stop* presuppose the existence of a proof. Also, Krahmer and Piwek (1999) briefly mentioned that the presuppositions triggered by noun phrases like *the fact that P* can be treated in a similar way (see also Section 3.3 for some discussion). Our claim is that the idea that proof terms act as antecedents of anaphora can be applied to the presuppositions of factive verbs in general.

[15] We leave open the possibility of decomposing the semantic representation of belief sentences in terms of possible worlds. See Tanaka *et al.* (2015) for discussion in the context of DTS. The problem of opacity and hyperintensionality (Fox and Lappin 2005) is beyond the scope of the present paper.

To account for the presuppositional inferences summarized in Section 3.1, we use the following lexical entry for *know*.[16]

(34) know; $(S\backslash NP)/\overline{S}$; $\lambda p.\lambda x.\lambda c.\,\mathbf{know}(x)(pc)(@_i c)$

Here the argument p is a dynamic proposition expressed by the declarative complement of *know*. The underspecified term $@_i$ takes a local context c as an argument and requires one to construct a proof term of type pc, i.e., to find *evidence* of the (static) proposition pc being true. If such a proof term is constructed, it fills the third argument position of the predicate **know**. In sum, the sentence x *knows that* P presupposes that there is a proof (evidence) of P and asserts that the agent x obtains it, i.e., x has a proof (evidence) of the proposition P.

Let us illustrate with (27a) how to give a compositional analysis of a construction containing *know*. The semantic representation for (27a) is given by the following CCG derivation tree.

(35)

			that	Mary is successful
			$\overline{S}/S$	S
		knows	λP.P	$\lambda c.\mathbf{successful}(\mathbf{mary})$
		$(S\backslash NP)/\overline{S}$		$\overline{S}$ >
John		$\lambda p.\lambda x.\lambda c.\,\mathbf{know}(x)(pc)(@_1 c)$		$\lambda c.\mathbf{successful}(\mathbf{mary})$
NP		$S\backslash NP$ >		
john		$\lambda x.\lambda c.\,\mathbf{know}(x)(\mathbf{successful}(\mathbf{mary}))(@_1 c)$		
		S <		
		$\lambda c.\,\mathbf{know}(\mathbf{john})(\mathbf{successful}(\mathbf{mary}))(@_1 c)$		

Then, the derivation (36) checks whether the semantic representation is well-typed.The open branch ending up with $\delta \to \mathbf{successful}(\mathbf{mary})$ shows the presupposition of this representation, which is the factive presupposition of (27a). In this way, we can correctly predict that the presuppositional inference from (27a) to (28) holds. The inference mechanism we described in Section 2.3 for the existence presupposition of definite descriptions can be extended for the case of *know*. In particular, it is easy to see that the projection inference in (29) and the filtering inference in (30) can be accounted for in the same way as those in (24) and (19), respectively.

[16] In (34), the underspecified term $@_i c$ is not annotated with its type pc, since it is inferable from the type of the predicate **know**.

(36)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\begin{array}{c}\mathbf{know}\\ :\mathbf{entity}\to(\mathsf{P}:\mathbf{type})\\ \to\mathsf{P}\to\mathbf{type}\end{array}}(CON) \quad \dfrac{}{\mathbf{john}:\mathbf{entity}}(CON)}{\begin{array}{c}\mathbf{know}(\mathbf{john})\\ :(\mathsf{P}:\mathbf{type})\to\mathsf{P}\to\mathbf{type}\end{array}}(\Pi E) \quad \dfrac{\dfrac{}{\begin{array}{c}\mathbf{successful}\\ :\mathbf{entity}\to\mathbf{type}\end{array}}(CON) \quad \dfrac{}{\mathbf{mary}:\mathbf{entity}}(CON)}{\mathbf{successful}(\mathbf{mary}):\mathbf{type}}(\Pi E)}{\begin{array}{c}\mathbf{know}(\mathbf{john})(\mathbf{successful}(\mathbf{mary}))\\ :\mathbf{successful}(\mathbf{mary})\to\mathbf{type}\end{array}}(\Pi E) \quad \dfrac{\dfrac{@_1}{:\delta\to\mathbf{successful}(\mathbf{mary})} \quad \dfrac{}{c:\delta}\,1}{\begin{array}{c}@_1 c\\ :\mathbf{successful}(\mathbf{mary})\end{array}}(\Pi E)}{\mathbf{know}(\mathbf{john})(\mathbf{successful}(\mathbf{mary}))(@_1 c):\mathbf{type}}(\Pi E)}{\lambda c.\,\mathbf{know}(\mathbf{john})(\mathbf{successful}(\mathbf{mary}))(@_1 c):\delta\to\mathbf{type}}(\Pi I),1$$

It is known that a sentence such as (37) poses the so-called *binding problem* (Karttunen 1971; Karttunen and Peters 1979; Cooper 1983).

(37) A student regrets that she talked.

Here, the existential quantification introduced by *a student* binds the pronoun *she* that appears in the presupposed content. This is an instance of the nested presuppositions that we mentioned in Section 2.3. In DTS, (37) can be given the semantic representation as shown in (38) (where the semantic representation of *regrets* is analogous to that of *knows* above). In this resulting representation, the Σ-type corresponding to the subject *a student* binds the variable u in the second argument of **regret**, which correponds to the type of $@_2(c, u)$ introduced by the factive presupposition of *regret*. With the help of the type checking procedure, this enables us to capture the dependency between assertion and presupposition.[17]

3.3 *NP-complements*

The analysis presented so far can be extended to the analysis of NP-complements. Let us first take the case of *believe*. Consider the example in (31a). The semantic representation of the definite NP *the rumor that Mary came* appearing in the object position can be derived as in (39). Since *rumor* is a content noun, we treat it as having the syntactic category N_{+c} with the syntactic feature $+c$. Correspondingly, the predicate **rumor** is analyzed as a predicate over propositions; its type is **type** → **type**. The semantic representation of definite article *the* is given in the same way as the one in Section 2.3 except that it combines with a predicate over propositions (i.e., objects of type **type**), rather than with a predicate over entities.[18]

The semantic representation for the premise sentence in (31a) can be derived as shown in (40). The resulting representation presupposes that there is a rumor whose content is identified with **come**(**mary**). This is the existence presupposition triggered by the NP-complement

[17] See Bekki and Mineshima (2017) for more details.

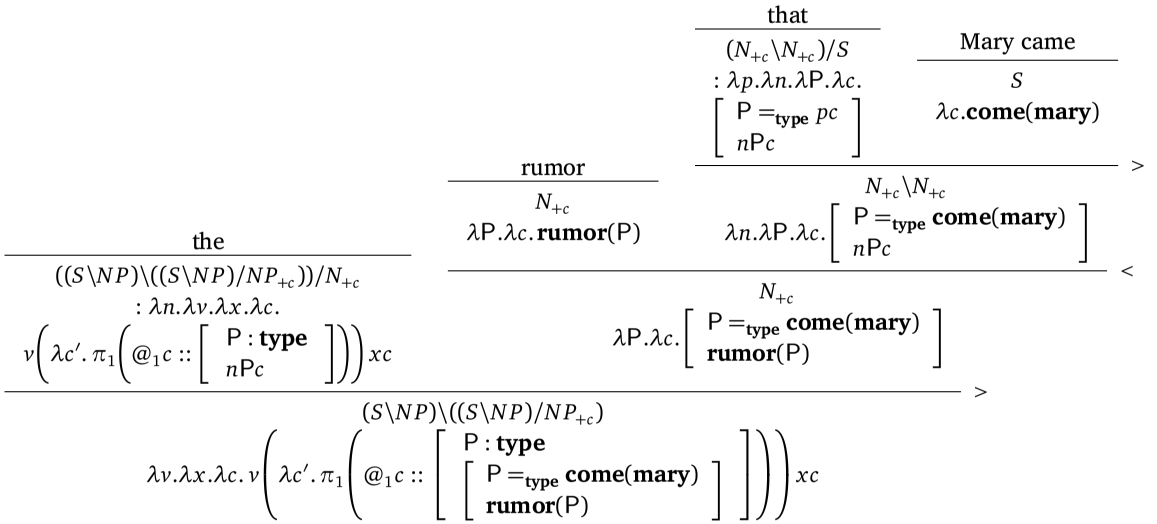
[18] Strictly speaking, to combine Σ-types with predicates over propositions requires the notion of type hierarchy (Martin-Löf 1984). For ease of exposition, we refer to the base type ($\mathbf{type}_0$) simply as **type**.

(38)

$$\dfrac{\dfrac{\text{A}}{S/(S\backslash NP)/N \quad \lambda n.\lambda v.\lambda c.\left[\begin{array}{l} u:\left[\begin{array}{l} x:\textbf{entity} \\ nxc \end{array}\right] \\ v(\pi_1 u)(c,u) \end{array}\right]} \quad \dfrac{\text{student}}{\begin{array}{c} N \\ \lambda x.\lambda c.\textbf{student}(x) \end{array}}}{S/(S\backslash NP) \quad \lambda v.\lambda c.\left[\begin{array}{l} u:\left[\begin{array}{l} x:\textbf{entity} \\ \textbf{student}(x) \end{array}\right] \\ v(\pi_1 u)(c,u) \end{array}\right]} > \quad \dfrac{\dfrac{\text{regrets}}{\begin{array}{c} (S\backslash NP)/\overline{S} \\ \lambda p.\lambda x.\lambda c. \\ \textbf{regret}(x)(pc)(@_2 c) \end{array}} \quad \dfrac{\dfrac{\text{that}}{\begin{array}{c} \overline{S}/S \\ \lambda p.p \end{array}} \quad \dfrac{\text{she talked}}{\begin{array}{c} S \\ \lambda c.\textbf{talk}(@_1 c :: \textbf{entity}) \end{array}}}{\begin{array}{c} \overline{S} \\ \lambda c.\textbf{talk}(@_1 c :: \textbf{entity}) \end{array}} >}{\begin{array}{c} S\backslash NP \\ \lambda x.\lambda c.\textbf{regret}(x)(\textbf{talk}(@_1 c :: \textbf{entity}))(@_2 c) \end{array}} >$$

$$\lambda c.\left[\begin{array}{l} u:\left[\begin{array}{l} x:\textbf{entity} \\ \textbf{student}(x) \end{array}\right] \\ \textbf{regret}(\pi_1 u)(\textbf{talk}(@_1(c,u) :: \textbf{entity}))(@_2(c,u)) \end{array}\right] \quad S \quad >$$

(39)


the
((S\NP)\((S\NP)/NP+c))/N+c
: λn.λv.λx.λc.
v(λc′. π1(@1c :: [P : type
nPc]))xc
rumor
N+c
λP.λc.rumor(P)
that
(N+c\N+c)/S
: λp.λn.λP.λc.
[P =type pc
nPc]
Mary came
S
λc.come(mary)
>
N+c\N+c
λn.λP.λc.[P =type come(mary)
nPc]
<
N+c
λP.λc.[P =type come(mary)
rumor(P)]
>
(S\NP)\((S\NP)/NP+c)
λv.λx.λc.v(λc′. π1(@1c :: [P : type
[P =type come(mary)
rumor(P)]]))xc

(40)

$$\dfrac{\dfrac{\text{John}}{\begin{array}{c}NP\\ \textbf{john}\end{array}} \qquad \dfrac{\dfrac{\text{believes}}{\begin{array}{c}(S\backslash NP)/NP_{+c}\\ \lambda p.\lambda x.\lambda c.\,\textbf{believe}(x)(pc)\end{array}} \qquad \dfrac{\text{the rumor that Mary came}}{\begin{array}{c}(S\backslash NP)\backslash((S\backslash NP)/NP_{+c})\\ \lambda v.\lambda x.\lambda c.\,v\left(\lambda c'.\pi_1\left(@_1 c :: \left[\begin{array}{l}\mathsf{P} : \textbf{type}\\ \mathsf{P} =_{\textbf{type}} \textbf{come}(\textbf{mary})\\ \textbf{rumor}(\mathsf{P})\end{array}\right]\right)\right)xc\end{array}}\,>}{\begin{array}{c}S\backslash NP\\ \lambda x.\lambda c.\,\textbf{believe}(x)\left(\pi_1\left(@_i c :: \left[\begin{array}{l}\mathsf{P} : \textbf{type}\\ \mathsf{P} =_{\textbf{type}} \textbf{come}(\textbf{mary})\\ \textbf{rumor}(\mathsf{P})\end{array}\right]\right)\right)\end{array}}\,>}{\begin{array}{c}S\\ \lambda c.\,\textbf{believe}(\textbf{john})\left(\pi_1\left(@_i c :: \left[\begin{array}{l}\mathsf{P} : \textbf{type}\\ \mathsf{P} =_{\textbf{type}} \textbf{come}(\textbf{mary})\\ \textbf{rumor}(\mathsf{P})\end{array}\right]\right)\right)\end{array}}$$

(cf. B3 in Figure 3). When this presupposition is satisfied, the resulting semantic representation can be reduced to **believe**(**john**, P), where we have P : **type**, P $=_{\textbf{type}}$ **come**(**mary**), and **rumor**(P). Thus, we can derive the representation **believe**(**john**, **come**(**mary**)), which is the representation for the conclusion in (31a). Hence, we can correctly derive the entailment pattern B2 in Figure 3.

It should be noted that, in the case of (31a), the predicate **rumor** does not contribute to the content of belief. By contrast, (31b) shows that, in the case of the factive verb *know* taking an NP-complement, the predicate **rumor** is part of the content of knowledge ascribed to the agent. The premise sentence in (31b) can be paraphrased as *John knows that there is a rumor that Mary came*. To handle (31b), then, we use the following lexical entries for the non-presuppositional use of *the*, which we refer to by the_{pred}.

(41) the_{pred} (subject position);

$$(S/(S\backslash NP_{+c}))/N_{+c}\,;\ \lambda n.\lambda v.\lambda c.\,v\left(\lambda c'.\left[\begin{array}{l}\mathsf{P}:\textbf{type}\\ n\mathsf{P}c\end{array}\right]\right)c$$

(42) the_{pred} (object position);

$$((S\backslash NP)\backslash((S\backslash NP)/NP_{+c}))/N_{+c}\,;\ \lambda n.\lambda v.\lambda x.\lambda c.\,v\left(\lambda c'.\left[\begin{array}{l}\mathsf{P}:\textbf{type}\\ n\mathsf{P}c\end{array}\right]\right)xc$$

In contrast to the entry given in (12), the_{pred} does not have existence presupposition and passes the whole existential proposition (Σ-type) to the main predicate.[19]

We take it that the existence presupposition associated with the premise sentence in (31b) comes from the factive verb *know*. Using the entry in (42), the semantic representation for the premise sentence in (31b) can be derived as in (43). The semantic representation derived in (43) presupposes that there is a rumor that Mary came and asserts that John has evidence for it. This is clearly distinguished from the reading

[19] Such a non-presuppositional use of definite description is also needed to handle examples such as *The king of France does not exist*, where the use of *the* does not presuppose the existence of the king of France.

(43)

$$\dfrac{\dfrac{\text{John}}{\begin{array}{c}NP\\ \textbf{john}\end{array}} \qquad \dfrac{\dfrac{\text{knows}}{\begin{array}{c}(S\backslash NP)/NP\\ \lambda p.\lambda x.\lambda c.\,\textbf{know}(x)(pc)(@_1 c)\end{array}} \qquad \dfrac{\text{the rumor that Mary came}}{\begin{array}{c}(S\backslash NP)\backslash((S\backslash NP)/NP)\\ \lambda v.\lambda x.\lambda c.\,v\left(\lambda c'.\left[\begin{array}{l}\mathsf{P} : \textbf{type}\\ \left[\begin{array}{l}\mathsf{P} =_{\textbf{type}} \textbf{come}(\textbf{mary})\\ \textbf{rumor}(\mathsf{P})\end{array}\right]\end{array}\right]\right)xc\end{array}}\,<}{S\backslash NP}\\ \lambda x.\lambda c.\textbf{know}(x)\left(\left[\begin{array}{l}\mathsf{P} : \textbf{type}\\ \left[\begin{array}{l}\mathsf{P} =_{\textbf{type}} \textbf{come}(\textbf{mary})\\ \textbf{rumor}(\mathsf{P})\end{array}\right]\end{array}\right]\right)(@_1 c)}\,<}{S}\\ \lambda c.\textbf{know}(\textbf{john})\left(\left[\begin{array}{l}\mathsf{P} : \textbf{type}\\ \left[\begin{array}{l}\mathsf{P} =_{\textbf{type}} \textbf{come}(\textbf{mary})\\ \textbf{rumor}(\mathsf{P})\end{array}\right]\end{array}\right]\right)(@_1 c)$$

Figure 4: Inferences associated with veridical content nouns

K4	x knows the fact that P	$\Leftrightarrow$	x knows that P
K5	x knows the fact that P	$\rhd$	There is a fact that P
K6	x knows the fact that P	$\rhd$	P

that John has evidence that Mary came, hence, we can account for the non-entailment in (31b), schematically given as K2 in Figure 3.[20]

As noted in Section 3.1, veridical content nouns such as *fact* and *truth* show a different entailment pattern from non-veridical content nouns such as *rumor* and *story*. The relevant inference patterns are summarized in Figure 4. The present analysis can naturally handle these inference patterns as well. Consider (44):

(44) John knows the fact that Mary came.

In the same way as the derivation in (43), we can obtain the semantic representation for (44):

(45) $\lambda c.\,\mathbf{know}(\mathbf{john})\left(\left[\begin{array}{l}\mathsf{P}:\mathbf{type}\\ \left[\begin{array}{l}\mathsf{P}=_{\mathbf{type}}\mathbf{come}(\mathbf{mary})\\ \mathbf{fact}(\mathsf{P})\end{array}\right]\end{array}\right]\right)(@_1 c)$

The underspecified term $@_1$ in (45) triggers the presupposition that there is the fact that Mary came, which accounts for K5 in Figure 4. To account for the other inference patterns, we may posit two axioms. The first axiom is the one concerning the lexical meaning of the veridical content noun *fact*:

(46) **Axiom 1**: $(\mathsf{P}:\mathbf{type}) \to (\mathbf{fact}(\mathsf{P}) \leftrightarrow \mathsf{P})$

Using this axiom, one can construct a proof term of **come**(**mary**) from the presupposition in (45), hence K6 in Figure 4 follows.

The second axiom we need is about the closure property of *know*:

(47) **Axiom 2**: $(x:\mathbf{entity}) \to (\mathsf{P}:\mathbf{type}) \to (\mathsf{Q}:\mathbf{type}) \to (a:\mathsf{P}) \to \mathbf{know}(x)(\mathsf{P})(a) \to (f:\mathsf{P}\to\mathsf{Q}) \to \mathbf{know}(x)(\mathsf{Q})(f\,a)$

The sentence *John knows that Mary came* can be compositionally assigned the semantic representation in (48), in the same way as the derivation shown in (35).

[20] As pointed out by an anonymous reviewer, the current analysis allows the combination of the verb *know* and the presuppositional *the*. This yields an unintended reading for (31b) where it is presupposed that Mary came. Though this undesirable reading could be blocked by involving a complicated syntactic analysis, we consider details of such an analysis as beyond the scope of this paper.

(48) $\lambda c.\mathbf{know}(\mathbf{john})(\mathbf{come}(\mathbf{mary}))(@_2 c)$

It is easy to derive (48) from (45) given an initial context c. First, assume that the presupposition in (45) is satisfied, that is, there is a proof term substituted for $@_1$ in (45). Using Axiom 1, we can construct a proof term substituted for $@_2$ in (48), that is, a proof term of **come**(**mary**), as well as a proof term for the proposition in (49).

(49) $\left[\begin{array}{l} \mathsf{P} : \mathbf{type} \\ \left[\begin{array}{l} \mathsf{P} =_{\mathbf{type}} \mathbf{come}(\mathbf{mary}) \\ \mathbf{fact}(\mathsf{P}) \end{array}\right] \end{array}\right] \rightarrow \mathbf{come}(\mathbf{mary})$

Hence, applying Axiom 2, we can obtain a proof term of (48). The other direction, i.e., the inference from (48) to (45), is derived in the same manner. Thus we can account for the pattern in K4.

Note that the present analysis of the inference pattern in K6 is different from what is suggested by Krahmer and Piwek (1999). These authors briefly discuss the presupposition triggered by the factive construction *(be) annoyed by the fact that P*. They treat this construction as one complex predicate, assuming that its presupposition is *P*. In our approach, *know the fact that P* is analyzed not as directly presupposing that *P*, but as presupposing the existence of the fact whose content is *P*. Under the present analysis, the inference in K6 is explained in terms of the lexical knowledge concerning the content noun *fact*.

4 CONCLUSION

This paper has attempted to provide an analysis of presuppositions and factivity within the framework of DTS. Under our analysis, factive and non-factive verbs are assigned different semantic types: while the non-factive predicate *believe* selects for a proposition as an object argument, the factive predicate *know* takes a proof-object as an extra argument. Using the machinery of underspecified semantics in DTS, we have illustrated how to account for a variety of inferences concerning factive and non-factive verbs.

Several open issues remain, most notably that of the interpretation of interrogative complements. It is acknowledged in the literature that the factive verb *know* takes interrogative complements, whereas the non-factive verb *believe* does not (Ginzburg 1995a,b; Egré 2008):

(50) a. John {knows, *believes} whether Ann or Bob came.
b. John {knows, *believes} who came.

Providing a detailed analysis of interrogative complements within our proof-theoretic framework is left for another occasion.

ACKNOWLEDGMENT

This paper is a revised and expanded version of a paper presented at the TYpe Theory and LExical Semantics (TYTLES) workshop during the 27th European Summer School in Logic, Language and Information (ESSLLI 2015). We are grateful to Robin Cooper, Christian Retoré, and the audience of TYTLES workshop for helpful discussions. We would also like to thank the three anonymous reviewers of this paper for their valuable comments and suggestions. This work was supported by JST CREST Grant Number JPMJCR1301, Japan. The first author acknowledges the financial support of the JSPS Grant-in-Aid for JSPS Fellows Grant Number 15J11772.

APPENDIX

Definition 1 (Alphabet for $\lambda_{\Pi\Sigma}$) *An alphabet for $\lambda_{\Pi\Sigma}$ is a $\langle Var, Con\rangle$, where Var is a set of variables, and Con is a set of constants. Dependent type semantics employs an alphabet as follows:*

$$Var \stackrel{def}{\equiv} \{x, y, z, u, v, ...\}$$
$$Con \stackrel{def}{\equiv} \{\textbf{entity}, \textbf{book}, \textbf{arrive}, ...\}$$

Definition 2 (Preterms) *The collection of preterms is recursively defined as follows (where $x \in Var$ and $c \in Con$, $j = 1, 2$, $i = 0, 1, 2, ...$).*

$$\begin{aligned}\Lambda := {} & x \mid c \mid @_i \mid \textbf{type}_i \mid \Lambda :: \Lambda \mid (x{:}\Lambda) \to \Lambda \mid \lambda x.\Lambda \mid \Lambda\Lambda \\ & \mid \begin{bmatrix} x : \Lambda \\ \Lambda \end{bmatrix} \mid (\Lambda, \Lambda) \mid \pi_j(\Lambda) \mid \Lambda =_\Lambda \Lambda \mid \mathsf{refl}_\Lambda(\Lambda) \mid \mathsf{idpeel}(\Lambda, \Lambda) \\ & \mid \bot \mid \top \mid \mathsf{case}_\Lambda(\Lambda_1, ..., \Lambda_n)\end{aligned}$$

Definition 3 (Signature) *A signature σ is defined recursively as follows.*

$$\sigma := () \mid \sigma, c : A$$

where $()$ is an empty signature, $c \in Con$, $A \in \Lambda$ s.t. $\vdash_\sigma A : \textbf{type}_i$ for some $i \in \mathbb{N}$.

Definition 4 (Context) *A context is defined recursively as follows.*

$$\Gamma := () \mid \Gamma, x : A$$

where $()$ *is an empty context,* $x \in Var$*,* $A \in \Lambda$ *s.t.* $\Gamma \vdash_\sigma A : \mathbf{type}_i$ *for some* $i \in \mathbb{N}$*.*

Definition 5 (Constant symbol rule) *For any* $(c : A) \in \sigma$*,*

$$\frac{}{c : A}\ (CON)$$

Definition 6 (Type rules) *For any* $i \in \mathbb{N}$*,*

$$\frac{A : \mathbf{type}_i}{A : \mathbf{type}_{i+1}}\ (\mathbf{type}I) \qquad \frac{}{\mathbf{type}_i : \mathbf{type}_{i+1}}\ (\mathbf{type}F)$$

Definition 7 (Π-type) *For any* $i, j \in \mathbb{N}$*,*

$$\frac{\begin{array}{c}\overline{x : A}^{\,j} \\ \vdots \\ A : \mathbf{type}_i \quad B : \mathbf{type}_i\end{array}}{(x{:}A) \to B : \mathbf{type}_i}\ (\Pi F), j \qquad \frac{\begin{array}{c}\overline{x : A}^{\,j} \\ \vdots \\ A : \mathbf{type}_i \quad M : B\end{array}}{\lambda x.M : (x{:}A) \to B}\ (\Pi I), j$$

$$\frac{M : (x{:}A) \to B \quad N : A}{MN : B[M/x]}\ (\Pi E)$$

Definition 8 (Σ-type) *For any* $i, j \in \mathbb{N}$*,*

$$\frac{\begin{array}{c}\overline{x : A}^{\,j} \\ \vdots \\ A : \mathbf{type}_i \quad B : \mathbf{type}_i\end{array}}{\begin{bmatrix} x : A \\ B \end{bmatrix} : \mathbf{type}_i}\ (\Sigma F), j \qquad \frac{M : A \quad N : B[M/x]}{(M, N) : \begin{bmatrix} x : A \\ B \end{bmatrix}}\ (\Sigma I)$$

$$\frac{M : \begin{bmatrix} x : A \\ B \end{bmatrix}}{\pi_1(M) : A}\ (\Sigma E) \qquad \frac{M : \begin{bmatrix} x : A \\ B \end{bmatrix}}{\pi_2(M) : B[\pi_1(M)/x]}\ (\Sigma E)$$

Definition 9 (Bottom type) *For any* $i \in \mathbb{N}$*,*

$$\frac{}{\bot : \mathbf{type}_0}\ (\bot F) \qquad \frac{M : \bot \quad C : \bot \to \mathbf{type}_i}{\mathsf{case}_M() : C(M)}\ (\bot E)$$

Definition 10 (Top type) *For any $i \in \mathbb{N}$,*

$$\frac{}{\top : \mathbf{type}_0}\ (\top F) \qquad \frac{}{() : \top}\ (\top I)$$

$$\frac{M : \top \quad C : \top \to \mathbf{type}_i \quad N : C()}{\mathsf{case}_M(N) : C(M)}\ (\top E)$$

Definition 11 (Id-type) *For any $i \in \mathbb{N}$,*

$$\frac{A : \mathbf{type}_i \quad M : A \quad N : A}{M =_A N : \mathbf{type}_i}\ (\mathsf{Id}F) \qquad \frac{A : \mathbf{type} \quad M : A}{\mathsf{refl}_A(M) : M =_A M}\ (\mathsf{Id}I)$$

$$\frac{E : M_1 =_A M_2 \quad C : (x{:}A) \to (y{:}A) \to (x =_A y \to \mathbf{type}_i) \quad N : (x{:}A) \to Cxx(\mathsf{refl}_A(x))}{\mathsf{idpeel}(e, N) : CM_1M_2E}\ (\mathsf{Id}E)$$

Definition 12 (@-formation rule) *For any i, $j \in \mathbb{N}$,*

$$\frac{A : \mathbf{type}_i \quad A\ true}{@_j : A}\ (@F)$$

Definition 13 (Type annotation rule)

$$\frac{t : A}{(t :: A) : A}\ (ann)$$

REFERENCES

Nicholas ASHER (2011), *Lexical Meaning in Context: A Web of Words*, Cambridge University Press, Cambridge.

Nicholas ASHER and Zhaohui LUO (2012), Formalisation of coercions in lexical semantics, in E. CHEMLA, V. HOMER, and G. WINTERSTEIN, editors, *Proceedings of Sinn und Bedeutung 17*, pp. 63–80, Paris, `http://semanticsarchive.net/sub2012/`.

David I. BEAVER (2001), *Presupposition and Assertion in Dynamic Semantics*, CSLI Publications, Stanford.

Daisuke BEKKI (2014), Representing anaphora with dependent types, in N. ASHER and S. SOLOVIEV, editors, *Logical Aspects of Computational Linguistics: 8th International Conference, LACL 2014, Proceedings*, volume 8535 of *Lecture Notes in Computer Science*, pp. 14–29, Springer, Heidelberg.

Daisuke BEKKI and Nicholas ASHER (2013), Logical polysemy and subtyping, in Y. MOTOMURA, A. BUTLER, and D. BEKKI, editors, *New Frontiers in Artificial Intelligence: JSAI-isAI 2012 Workshops, Revised Selected Papers*, volume 7856 of *Lecture Notes in Computer Science*, pp. 17–24, Springer, Heidelberg.

Daisuke BEKKI and Koji MINESHIMA (2017), Context-passing and underspecification in Dependent Type Semantics, in S. CHATZIKYRIAKIDIS and Z. LUO, editors, *Modern Perspectives in Type-Theoretical Semantics*, volume 98 of *Studies in Linguistics and Philosophy*, pp. 11–41, Springer, Heidelberg.

Daisuke BEKKI and Miho SATOH (2015), Calculating projections via type checking, in R. COOPER and C. RETORÉ, editors, *ESSLLI proceedings of the TYTLES workshop on Type Theory and Lexical Semantics ESSLLI2015*, Barcelona.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2014), Natural language inference in Coq, *Journal of Logic, Language and Information*, 23(4):441–480.

Robin COOPER (1983), *Quantification and Syntactic Theory*, Reidel, Dordrecht.

Rogelio DÁVILA-PÉREZ (1995), *Semantics and Parsing in Intuitionistic Categorial Grammar*, Ph.D. thesis, University of Essex.

Paul EGRÉ (2008), Question-embedding and factivity, *Grazer Philosophische Studien*, 77(1):85–125.

Delia Graff FARA (2001), Descriptions as predicates, *Philosophical Studies*, 102:1–42, originally published under the name "Delia Graff".

Chris FOX and Shalom LAPPIN (2005), *Foundations of Intensional Semantics*, Blackwell, Oxford.

Jonathan GINZBURG (1995a), Resolving questions, I, *Linguistics and Philosophy*, 18(5):459–527.

Jonathan GINZBURG (1995b), Resolving questions, II, *Linguistics and Philosophy*, 18(6):567–609.

Jeroen GROENENDIJK and Martin STOKHOF (1991), Dynamic predicate logic, *Linguistics and Philosophy*, 14(1):39–100.

Irene HEIM (1982), *The Semantics of Definite and Indefinite Noun Phrases*, Ph.D. thesis, University of Massachusetts, Amherst.

Irene HEIM (1983), On the projection problem for presuppositions, in M. BARLOW, D. FLICKINGER, and M. WESCOAT, editors, *Proceedings of the Second West Coast Conference on Formal Linguistics*, pp. 114–125, Stanford University Press, Stanford, CA.

Jaakko HINTIKKA (1962), *Knowledge and Belief: An Introduction to the Logic of the Two Notions*, Cornell University Press, Ithaca, NY.

Jaakko HINTIKKA (1969), Semantics for propositional attitudes, in J. W. DAVIS, D. J. HOCKNEY, and W. K. WILSON, editors, *Philosophical Logic*, volume 20 of *Synthese Library*, pp. 21–45, Reidel, Dordrecht.

William Alvin HOWARD (1980), The formulae-as-types notion of construction, in J. P. SELDIN and J. R. HINDLEY, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pp. 480–490, Academic Press, London.

Hans KAMP, Josef VAN GENABITH, and Uwe REYLE (2011), Discourse Representation Theory, in D. M. GABBAY and F. GUENTHNER, editors, *Handbook of Philosophical Logic*, volume 15, pp. 125–394, Springer, Heidelberg.

Lauri KARTTUNEN (1971), Implicative verbs, *Language*, 47(2):340–358.

Lauri KARTTUNEN and Stanley PETERS (1979), Conventional implicatures, in C. K. OH and D. A. DINNEEN, editors, *Syntax and Semantics 11: Presupposition*, pp. 1–56, Academic Press, New York, NY.

Jeffrey C. KING (2002), Designating propositions, *The Philosophical Review*, 111(3):341–371.

Eriko KINOSHITA, Koji MINESHIMA, and Daisuke BEKKI (2016), An analysis of selectional restrictions with Dependent Type Semantics, in *Proceedings of the 13th International Workshop on Logic and Engineering of Natural Language Semantics (LENLS13)*, pp. 100–113, Kanagawa.

Paul KIPARSKY and Carol KIPARSKY (1970), Fact, in M. BIERWISCH and K. E. HEIDOLPH, editors, *Progress in Linguistics*, pp. 143–173, de Gruyter Mouton, Berlin.

Emiel KRAHMER and Paul PIWEK (1999), Presupposition projection as proof construction, in H. BUNT and R. MUSKENS, editors, *Computing Meaning: Volume 1*, volume 73 of *Studies in Linguistics and Philosophy*, pp. 281–300, Kluwer Academic Publishers, Dordrecht.

Yusuke KUBOTA and Robert LEVINE (2017), Scope parallelism in coordination in Dependent Type Semantics, in M. OTAKE, S. KURAHASHI, Y. OTA, K. SATOH, and D. BEKKI, editors, *New Frontiers in Artificial Intelligence: JSAI-isAI 2015 Workshops, Revised Selected Papers*, volume 10091 of *Lecture Notes in Artificial Intelligence*, pp. 149–162, Springer, Heidelberg.

Susumu KUNO (1970), Some properties of non-referential noun phrases, in R. JAKOBSON and S. KAWAMOTO, editors, *Studies in General and Oriental Linguistics. Presented to S. Hattori on Occasion of his Sixtieth Birthday*, pp. 348–373, TEC, Tokyo.

Zhaohui LUO (2012a), Common nouns as types, in D. BÉCHET and A. DIKOVSKY, editors, *Logical Aspects of Computational Linguistics: 7th International Conference, LACL 2012, Proceedings*, volume 7351 of *Theoretical Computer Science and General Issues*, pp. 173–185, Springer, Heidelberg.

Zhaohui LUO (2012b), Formal semantics in modern type theories with coercive subtyping, *Linguistics and Philosophy*, 35(6):491–513.

Per MARTIN-LÖF (1984), *Intuitionistic Type Theory. Notes by G. Sambin*, Bibliopolis, Naples.

John-Jules Ch MEYER and Wiebe VAN DER HOEK (2004), *Epistemic Logic for AI and Computer Science*, Cambridge University Press, Cambridge.

Line MIKKELSEN (2005), *Copular Clauses: Specification, Predication and Equation*, John Benjamins, Amsterdam.

Line MIKKELSEN (2011), Copular clauses, in C. MAIENBORN, K. VON HEUSINGER, and P. PORTNER, editors, *Semantics: An International Handbook of Natural Language Meaning*, volume 2, pp. 1805–1829, de Gruyter Mouton, Berlin.

Richard MONTAGUE (1973), The proper treatment of quantification in ordinary English, in P. SUPPES, J. MORAVCSIK, and J. HINTIKKA, editors, *Approaches to Natural Language*, pp. 221–242, Kluwer Academic Publishers, Dordrecht.

Bengt NORDSTRÖM, Kent PETERSSON, and Jan M. SMITH (1990), *Programming in Martin-Löf's Type Theory: An Introduction*, Oxford University Press, Oxford.

Terence PARSONS (1993), On denoting propositions and facts, *Philosophical Perspectives*, 7:441–460.

Paul PIWEK and Emiel KRAHMER (2000), Presuppositions in context: constructing bridges, in P. BONZON, M. CAVALCANTI, and R. NOSSUM, editors, *Formal Aspects of Context*, volume 20 of *Applied Logic Series*, pp. 85–106, Kluwer Academic Publishers, Dordrecht.

Aarne RANTA (1994), *Type-Theoretical Grammar*, Oxford University Press, Oxford.

Christian RETORÉ (2014), The Montagovian generative lexicon ΛTy_n: a type theoretical framework for natural language semantics, in R. MATTHES and A. SCHUBERT, editors, *19th International Conference on Types for Proofs and Programs (TYPES 2013)*, volume 26 of *Leibniz International Proceedings in Informatics*, pp. 202–229, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, `http://drops.dagstuhl.de/opus/volltexte/2014/4633/`.

Bertrand RUSSELL (1919), *Introduction to Mathematical Philosophy*, George Allen & Unwin, London.

Mark STEEDMAN (2000), *The Syntactic Process*, MIT Press, Cambridge, MA.

Göran SUNDHOLM (1986), Proof Theory and Meaning, in D. M. GABBAY and F. GUENTHNER, editors, *Handbook of Philosophical Logic*, volume 3, pp. 471–506, Reidel, Dordrecht.

Göran SUNDHOLM (1989), Constructive generalized quantifiers, *Synthese*, 79(1):1–12.

Ribeka TANAKA (2014), A proof-theoretic approach to generalized quantifiers in dependent type semantics, in R. DE HAAN, editor, *Proceedings of the ESSLLI2014 Student Session*, pp. 140–151, Tübingen, `http://www.kr.tuwien.ac.at/drm/dehaan/stus2014/proceedings.pdf`.

Ribeka TANAKA, Koji MINESHIMA, and Daisuke BEKKI (2015), Resolving modal anaphora in Dependent Type Semantics, in T. MURATA, K. MINESHIMA,

and D. BEKKI, editors, *New Frontiers in Artificial Intelligence: JSAI-isAI 2014 Workshops, Revised Selected Papers*, pp. 83–98, Springer, Heidelberg.

Ribeka TANAKA, Yuki NAKANO, and Daisuke BEKKI (2014), Constructive generalized quantifiers revisited, in Y. NAKANO, K. SATOH, and D. BEKKI, editors, *New Frontiers in Artificial Intelligence: JSAI-isAI 2013 Workshops, Revised Selected Papers*, volume 8417 of *Lecture Notes in Computer Science*, pp. 115–124, Springer, Heidelberg.

Wataru UEGAKI (2016), Content nouns and the semantics of question-embedding, *Journal of Semantics*, 33(4):623–660.

Rob A. VAN DER SANDT (1992), Presupposition projection as anaphora resolution, *Journal of Semantics*, 9:333–377.

Zeno VENDLER (1972), *Res Cogitans: An Essay in Rational Psychology*, Cornell University Press, Ithaca, NY.