# Journal of Language Modelling

VOLUME 2 ISSUE 1
JUNE 2014

*Institute of Computer Science*
*Polish Academy of Sciences*
*Warsaw*

# Journal of Language Modelling

VOLUME 2 ISSUE 1
JUNE 2014

JOURNAL OF
LANGUAGE MODELLING

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

MANAGING EDITOR
*Adam Przepiórkowski* IPI PAN

SECTION EDITORS
*Elżbieta Hajnicz* IPI PAN
*Agnieszka Mykowiecka* IPI PAN
*Marcin Woliński* IPI PAN

STATISTICS EDITOR
*Łukasz Dębowski* IPI PAN


Published by IPI PAN
Instytut Podstaw Informatyki
Polskiej Akademii Nauk
*Institute of Computer Science*
*Polish Academy of Sciences*
ul. Jana Kazimierza 5, 01-248 Warszawa, Poland

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.

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

# EDITORIAL BOARD

*Steven Abney* University of Michigan, USA

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

*Chris Biemann* Technische Universität Darmstadt, GERMANY

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

*António Branco* University of Lisbon, PORTUGAL

*David Chiang* University of Southern California, Los Angeles, USA

*Greville Corbett* University of Surrey, UNITED KINGDOM

*Dan Cristea* University of Iași, ROMANIA

*Jan Daciuk* Gdańsk University of Technology, POLAND

*Mary Dalrymple* University of Oxford, UNITED KINGDOM

*Darja Fišer* University of Ljubljana, SLOVENIA

*Anette Frank* Universität Heidelberg, GERMANY

*Claire Gardent* CNRS/LORIA, Nancy, FRANCE

*Jonathan Ginzburg* Université Paris-Diderot, FRANCE

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

*Heiki-Jaan Kaalep* University of Tartu, ESTONIA

*Laura Kallmeyer* Heinrich-Heine-Universität Düsseldorf, GERMANY

*Jong-Bok Kim* Kyung Hee University, Seoul, KOREA

*Kimmo Koskenniemi* University of Helsinki, FINLAND

*Jonas Kuhn* Universität Stuttgart, GERMANY

*Alessandro Lenci* University of Pisa, ITALY

*Ján Mačutek* Comenius University in Bratislava, SLOVAKIA

*Igor Mel'čuk* University of Montreal, CANADA

*Glyn Morrill* Technical University of Catalonia, Barcelona, SPAIN

*Stefan Müller* Freie Universität Berlin, GERMANY

*Reinhard Muskens* Tilburg University, NETHERLANDS

*Mark-Jan Nederhof* University of St Andrews, UNITED KINGDOM

*Petya Osenova* Sofia University, BULGARIA

*David Pesetsky* Massachusetts Institute of Technology, USA

*Maciej Piasecki* Wrocław University of Technology, POLAND

*Christopher Potts* Stanford University, USA

*Louisa Sadler* University of Essex, UNITED KINGDOM

*Ivan A. Sag †* Stanford University, USA

*Agata Savary* Université François Rabelais Tours, FRANCE

*Sabine Schulte im Walde* Universität Stuttgart, GERMANY

*Stuart M. Shieber* Harvard University, USA

*Mark Steedman* University of Edinburgh, UNITED KINGDOM

*Stan Szpakowicz* School of Electrical Engineering
and Computer Science, University of Ottawa, CANADA

*Shravan Vasishth* Universität Potsdam, GERMANY

*Zygmunt Vetulani* Adam Mickiewicz University, Poznań, POLAND

*Aline Villavicencio* Federal University of Rio Grande do Sul,
Porto Alegre, BRAZIL

*Veronika Vincze* University of Szeged, HUNGARY

*Yorick Wilks* Florida Institute of Human and Machine Cognition, USA

*Shuly Wintner* University of Haifa, ISRAEL

*Zdeněk Žabokrtský* Charles University in Prague, CZECH REPUBLIC

# Evaluation of automatic updates of *Roget's Thesaurus*

*Alistair Kennedy*[1] *and Stan Szpakowicz*[2,1]
[1] School of Electrical Engineering and Computer Science
University of Ottawa, Ottawa, Ontario, Canada
[2] Institute of Computer Science
Polish Academy of Sciences, Warsaw, Poland

## ABSTRACT

Thesauri and similarly organised resources attract increasing interest of Natural Language Processing researchers. Thesauri age fast, so there is a constant need to update their vocabulary. Since a manual update cycle takes considerable time, automated methods are required. This work presents a tuneable method of measuring semantic relatedness, trained on *Roget's Thesaurus*, which generates lists of terms related to words not yet in the *Thesaurus*. Using these lists of terms, we experiment with three methods of adding words to the *Thesaurus*. We add, with high confidence, over 5500 and 9600 new word senses to versions of *Roget's Thesaurus* from 1911 and 1987 respectively.

We evaluate our work both manually and by applying the updated thesauri in three NLP tasks: selection of the best synonym from a set of candidates, pseudo-word-sense disambiguation and SAT-style analogy problems. We find that the newly added words are of high quality. The additions significantly improve the performance of *Roget's*-based methods in these NLP tasks. The performance of our system compares favourably with that of *WordNet*-based methods. Our methods are general enough to work with different versions of *Roget's Thesaurus*.

*Keywords: lexical resources, Roget's Thesaurus, WordNet, semantic relatedness, synonym selection, pseudo-word-sense disambiguation, analogy*

## 1 INTRODUCTION

Thesauri and other similarly organised lexical knowledge bases play a major role in applications of Natural Language Processing (NLP). While *Roget's Thesaurus*, whose original form is 160 years old, has been applied successfully, the NLP community turns most often to *WordNet* (Fellbaum 1998). *WordNet*'s intrinsic advantages notwithstanding, one of the reasons is that no other similar resource, including *Roget's Thesaurus*, has been publicly available in a suitable software package. It is, however, important to note that *WordNet* represents *one* of the methods of organising the English lexicon, and need not be the superior resource for every task. *Roget's Thesaurus* updated with the most recent vocabulary can become a competitive resource whose quality measures up to *WordNet*'s on a variety of NLP applications. In this paper, we describe and evaluate a few variations on an innovative method of updating the lexicon of *Roget's Thesaurus*.

Work on learning to construct or enhance a thesaurus by clustering related words goes back over two decades (Tsurumaru *et al.* 1986; Crouch 1988; Crouch and Yang 1992). Few methods use an existing resource in the process of updating that same resource. We employ *Roget's Thesaurus* in two ways when creating its updated versions. First, we construct a measure of semantic relatedness between terms, and tune a system to place a word in the *Thesaurus*. Next, we use the resource to "learn" how to place new words in the correct locations. This paper focusses on finding how to place a new word appropriately.

We evaluate our lexicon-updating methods on two versions of *Roget's Thesaurus*, with the vocabulary from 1911 and from 1987. Printed versions are periodically updated, but new releases – neither easily available to NLP researchers nor NLP-friendly – have had little effect on the community. The 1911 version of *Roget's Thesaurus* is freely available through Project Gutenberg.[1] We also work with the 1987 edition of *Penguin's Roget's Thesaurus* (Kirkpatrick 1987). An open Java API for the 1911 *Roget's Thesaurus* and its updated versions – including every addition we discuss in this paper – are available on the Web as the *Open Roget's Project*.[2] The API has been built on the work of Jarmasz (2003).

[1] `http://www.gutenberg.org/ebooks/22`

[2] `http://rogets.eecs.uottawa.ca/`

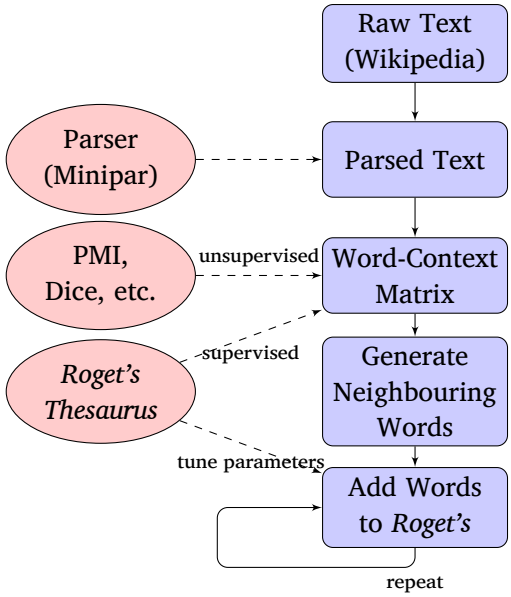



Figure 1:
The process of adding new words to *Roget's Thesaurus*

Figure 1 outlines the process of updating *Roget's Thesaurus*. We work with Wikipedia as a corpus and with the parser *MINIPAR* (Lin 1998a). Raw text is parsed, and a word–context matrix is constructed and re-weighted in both a supervised and an unsupervised manner. The nearest synonyms of each word in the matrix are generated and a location for them in *Roget's Thesaurus* is deduced using it as a source of tuning data. The last step can be applied iteratively to update the lexicon of *Roget's Thesaurus*.

This work makes six main contributions:

- apply the supervised measures of semantic relatedness from (Kennedy and Szpakowicz 2011) and (Kennedy and Szpakowicz 2012) to the updating of *Roget's Thesaurus*, and evaluate it carefully;
- propose and compare three methods of automatically adding words to *Roget's Thesaurus*;
- build the updated editions of the 1911 and 1987 versions of *Roget's Thesaurus*;
- create new datasets for pseudo-word-sense disambiguation and the selection of the best synonym;

- propose and evaluate a new method for solving SAT-style analogy problems;
- compare semantic similarity calculation with *Roget's Thesaurus* and *WordNet* on accuracy and on runtime.

### 1.1 *About Roget's Thesaurus*

In the early 1800s, Peter Mark Roget, a physician, began to categorise terms and phrases for his personal use in writing. The ensuing *Roget's Thesaurus*, first published in 1852, has gone through many revisions continuing to this day (Kendall 2008). A nine-level hierarchy makes up most of the structure of *Roget's Thesaurus*:

| | | | |
|---|---|---|---|
| 1 | Class | 6 | Part of Speech |
| 2 | Section | 7 | Paragraph |
| 3 | Sub-Section | 8 | Semicolon Group |
| 4 | Head Group | 9 | Words and Phrases |
| 5 | Head | | |

Eight classes are subdivided into Sections and Sub-Sections. There are around 1000 Heads – the main category in *Roget's Thesaurus*, corresponding to major concepts. Heads with opposing or complementary concepts form a Head Group. A Part of Speech (POS) groups all noun/verb/adjective/adverb realisations of the Head's concept. The closest counterpart of *WordNet*'s synsets is a Semicolon Group (SG). An SG contains closely related words (usually near-synonyms); a Paragraph contains related SGs. Note the division by part-of-speech quite low in the hierarchy, not at the very top as in *WordNet*. We define a *Roget's grouping* to be the set of words contained within an instance of any of these levels. A Section or even a Class is also a *Roget's grouping*, but usually we talk about words in the same POS, Paragraph or SG.

Figure 2 shows an example of a Head. Head #586 in the 1911 *Roget's Thesaurus* contains terms pertaining to language. A number before a word refers to a Head in which that word-sense may also be found. Although a thorough update of *Roget's Thesaurus* should include such *cross-references*, they are beyond the scope of this work.[3]

[3] They do not figure in any of the applications we consider here to test the quality of the updated versions of the *Thesaurus*.

Class 5: Intellect: communication of ideas
Section 3: Means of communicating ideas
Sub-Section: Conventional means
Head Group: 586 Language
Head: 586 Language

N. *language*; *595* phraseology; *608* speech; tongue, lingo, vernacular; mother tongue, vulgar tongue, native tongue; household words; King's English, Queen's English; *589* dialect.

*confusion of tongues*, Babel, pasigraphie; *sign 576* pantomime; onomatopoeia; betacism, mimmation, myatism, nunnation; pasigraphy.

*lexicology*, philology, glossology, glottology; linguistics, chrestomathy; paleology, paleography; comparative grammar.

*literature*, letters, polite literature, belles lettres, muses, humanities, literae humaniores, republic of letters, dead languages, classics; genius of language; *scholar 516* scholarship.

VB. *592 express by words*.

ADJ. *lingual*, linguistic; dialectic; vernacular, current; bilingual; diglot, hexaglot, polyglot; literary.

Figure 2: Head 586: Language from the 1911 *Roget's Thesaurus*

### 1.2 *Where to add new words to Roget's Thesaurus*

The number of Heads and POSs per Head have changed little between the 1911 and 1987 versions of *Roget's Thesaurus*. We can aim to add new words in three different ways:

- in an existing SG,
- in a new SG in an existing Paragraph,
- in a new SG in a new Paragraph.

Evaluation of a new semantic distance measure should then be useful at identifying words in the same POS, Paragraph and SG.

## 2 PREVIOUS WORK ON UPDATING THESAURI

There have been few attempts to expand the lexicon of *Roget's Thesaurus* thus far. Cassidy (2000) added manually a few hundred words to the 1911 edition of *Roget's Thesaurus*. Kennedy and Szpakowicz (2007) disambiguated hypernym instances in the 1987 *Roget's Thesaurus*. Both projects augmented *Roget's Thesaurus*, but did not offer insight into how to update the lexicon automatically.

Other related work includes mapping word senses between *Roget's Thesaurus*, *WordNet* and *LDOCE* (Procter 1978). The contexts where a word appears, whether it is words in the same Paragraph, *WordNet* synset or an *LDOCE* definition, are used to deduce which words are likely to be related (Kwong 1998a,b; Nastase and Szpakowicz 2001).

### 2.1 *Updating WordNet*

The automatic expansion of *WordNet*'s lexicon has been attempted several times. Snow *et al.* (2006) extracted thousands of new words from a corpus for possible inclusion in *WordNet* (though that expansion never materialised in practice due to its low accuracy). Many of the new terms were proper nouns found in a corpus by a machine learning system (Snow *et al.* 2005) which was used to discover IS-A relations using dependency paths generated by *MINIPAR* (Lin 1998b).

Pantel (2005) created semantic vectors for each word in *WordNet* by disambiguating contexts which appeared with different senses of a word. The building of semantic vectors is described in (Pantel 2003). *WordNet*'s hierarchy was used to propagate contexts where words may appear throughout the network. A word sense was then represented by contexts from its semantic vector not shared with its parents. Pantel did not attempt to place new words into the resource, only evaluated the method on existing words. This technique was only examined for nouns. It presumably applied to verbs as well, but could not be tried on adjectives or adverbs, for which there was no usable hypernym hierarchy.

A folksonomy is a Web service which allows users to annotate Web sites (among other things) with strings of their choice. One such folksonomy was *Delicious* where users categorised Web pages. HYPERNYM/HYPONYM relations can be extracted from folksonomies by identifying tags subsuming other tags. Zheng *et al.* (2008) describe how to use folksonomies to discover instances of hypernymy and so help put new words into *WordNet*.

Not directly applicable but relevant to our work is semi-automatic enhancement of *WordNet* with sentiment and affect information. Esuli and Sebastiani (2006) used machine learning to build *SentiWordNet* by labelling synsets in *WordNet* 2.0 as objective, positive or negative. In *WordNet Affect* (Strapparava and Valitutti 2004), synsets got one or more labels, often related to emotion. An initial set of words marked

with emotions was built manually. Next, those emotions were propagated to other synsets via *WordNet* relations. This work was based on *WordNet Domains* (Magnini and Cavagliá 2000), a framework which allows a user to augment *WordNet* by adding domain labels to synsets. No new words were added, but these projects highlight some of the more successful experiments with enhancing *WordNet*.

There is a reasonable amount of work on mining hypernym relations from text, which could then be used to update *WordNet*. This includes using set patterns (Hearst 1992; Sombatsrisomboon *et al.* 2003) or discovering new patterns using a few seed sets of hypernyms (Morin and Jacquemin 1999). Languages other than English for which hypernym mining has been attempted include Swedish (Rydin 2002), Dutch (Sang 2007) and Japanese (Shinzato and Torisawa 2004). There also has been research on hierarchically related verbs (Girju *et al.* 2003, 2006).

### 2.2 *Wordnets in other languages*

There has been much work on building wordnets for languages other than English, loosely coordinated by the Global Wordnet Association.[4] One strategy is to take the Princeton *WordNet* (Fellbaum 1998) as a starting point. That was the mode of operation in the *EuroWordNet* project (Vossen 1998), an early initiative meant to build wordnets for several European languages. One of its offshoots is *BalkaNet*.[5]

The other wordnet-building strategy is to avoid the influence of Princeton *WordNet*. Polish *WordNet* (Piasecki *et al.* 2009) is one such resource built from scratch. Its development was supported, among others, by *WordNet Weaver*, a tool which helps increase the vocabulary of a wordnet. The tool implements a two-phase algorithm. Phase I identifies a network vicinity in which to place a new word, while phase II connects possible candidate synsets. Phase II is semi-automatic: it is the linguists who decide what additions are ultimately made to the growing Polish *WordNet*.

[4] See `http://globalwordnet.org/wordnets-in-the-world/` for an up-to-date list of available wordnets.

[5] "The Balkan WordNet aims at the development of a multilingual lexical database comprising of individual WordNets for the Balkan languages." (`http://www.dblab.upatras.gr/balkanet/`)

Lemnitzer *et al.* (2008) discuss adding semantic relationships between nouns and verbs to *GermaNet*, a German wordnet. Those were verb–object relationships believed to be useful in applications such as text summarisation or anaphora resolution. Sagot and Fišer (2011) present an automatic, language-independent method (tested on Slovene and French) of extending a wordnet by "recycling" freely available bilingual resources such as machine-readable dictionaries and on-line encyclopaedias.

## 3 MEASURING SEMANTIC RELATEDNESS

Distributional measures of semantic relatedness (MSRs) use a word's context to help determine its meaning. Words which frequently appear in similar contexts are assumed to have similar meaning. Such MSRs usually re-weight contexts by considering some measure of their importance, usually the association between a context and the terms it contains. One of the most successful measures is Pointwise Mutual Information (PMI). PMI increases the weight of contexts where a word appears regularly but other words do not, and decreases the weight of contexts where many words may appear. Essentially, it is unsupervised feature weighting.

Kennedy and Szpakowicz (2011, 2012) discussed introducing supervision into the process of context re-weighting. Their method identifies contexts where pairs of words known to be semantically related frequently appear, and then uses a measure of association to re-weight these contexts by how often they contain closely related words. The method, very general, can work with any thesaurus as a source of known synonym pairs and with measures of association other than PMI. Here, this measure will help update *Roget's Thesaurus*. This section describes in general how this method is applied.

### 3.1 *Building a word–context matrix for semantic relatedness*

We used Wikipedia[6] as a source of data and parsed it with *MINIPAR* (Lin 1998a). The choice of dependency triples instead of all neighbouring words favours contexts which most directly affect a word's meaning. Examples of triples are $\langle time, mod, unlimited \rangle$ and

[6] A dump of August 2010.

$\langle time, conj, motion \rangle$: "time" appears in contexts with the modifier "unlimited" and in a conjunction with "motion". Some 900 million dependency triples generated by parsing Wikipedia took up ≈20GB.

Three matrices were built, one each for nouns, verbs and adjectives/adverbs.[7] For each word–relation–word triple $\langle w_1, r, w_2 \rangle$ we generated two word–context pairs $(w_1, \langle r, w_2 \rangle)$ and $(w_2, \langle w_1, r \rangle)$. Words $w_1$ and $w_2$ could be of any part of speech. All relations $r$ were considered, with the direction of $r$ retained. When $w_1$ or $w_2$ was an individual term, it had to be a noun, verb, adjective or adverb, written in lower case (*MINIPAR* only leaves proper nouns capitalised).

With these constraints we used all of the Wikipedia dump when building the matrices for verbs and adjectives/adverbs, but only 50% for nouns. This limit was chosen both because it was the most data which could be held in a system with 4GB of RAM and because the leftover data could be used in later evaluation.

Very infrequent words and contexts tend to be unreliable, and often appear because of spelling errors. We established thresholds for how often a word or context needs to appear. We measured the quality of synonyms generated for a set of randomly selected words which appear with different frequencies in the matrix. Next, in a series of straightforward experiments, we selected a cutoff after which the quality of the synonyms does not appear to improve: 35 for nouns and for adjectives, 10 for verbs. Also, an entry must appear in a context at least twice for the context to count. Table 1 shows the counts of words and contexts in each matrix before and after the cutoff. Non-zero entries are cells with positive values. While the reduction of the matrix dimensionally was quite large, the decrease in the number of non-zero entries was very small. So, we lost little information, but created a much denser and more informative matrix.

### 3.2 *Measures of semantic relatedness*

We explored two complementary methods of re-weighting the word-context matrix. An unsupervised method measures association between words and contexts; a supervised method uses known pairs of synonyms in *Roget's Thesaurus* to determine which contexts have a

[7] We have decided to work with *MINIPAR*'s labelling system, which does not distinguish between adjectives and adverbs.

Table 1: Counts of the rows, columns and non-zero entries for each matrix

| POS | Matrix | Words | Contexts | Non-zero entries | % non-zero |
|---|---|---|---|---|---|
| Noun ($\geq$ 35) | Full | 359 380 | 2 463 001 | 30 994 968 | 0.0035% |
| | Cutoff | 43 834 | 1 050 178 | 28 296 890 | 0.0615% |
| | (*% of full*) | (*12.2%*) | (*42.6%*) | (*91.3%*) | |
| Verb ($\geq$ 10) | Full | 9 294 | 2 892 002 | 26 716 709 | 0.0994% |
| | Cutoff | 7141 | 1 423 665 | 25 239 485 | 0.2483% |
| | (*% of full*) | (*76.8%*) | (*49.3%*) | (*94.5%*) | |
| Adj/Adv ($\geq$ 35) | Full | 104 074 | 817 921 | 9 116 741 | 0.0107% |
| | Cutoff | 17 160 | 360 436 | 8 379 637 | 0.1355% |
| | (*% of full*) | (*16.5%*) | (*44.1%*) | (*91.9%*) | |

higher tendency to contain pairs of known synonyms (Kennedy and Szpakowicz 2011, 2012). Supervision can be conducted on each individual context, or on groups of contexts with a syntactic relation in common. It was found that supervision at the context level worked best for nouns and verbs, while grouping contexts by relation worked best for adjectives (Kennedy and Szpakowicz 2012).

Both supervised and unsupervised methods employ measures of association; Kennedy and Szpakowicz (2012) found that in all cases PMI was the most successful. These two kinds of methods can actually be complementary. It is possible to use the supervised method of matrix re-weighting and then apply the unsupervised method on top of it. This was generally found to yield the best results; so this is how we report the results.

To evaluate this work, we created a random set of 1000 nouns, 600 verbs and 600 adjectives and generated lists of neighbouring words for each of them.[8] Those words were left out of the training process. We then measured the precision – how many neighbouring words appeared in the same SG, Paragraph or POS – in the 1987 *Roget's Thesaurus*. Precision was measured at several recall points: the top 1, 5, 10, 20, 50 and 100 words retrieved from the 1987 *Thesaurus*.

Table 2 shows the results for the unsupervised baseline, using PMI weighting and the results for the combined supervised methods using synonyms from either the 1911 or the 1987 version of *Roget's*

[8] There were not enough adverbs to construct such a set. Adverbs will be left for future work.

| Year | POS | Group | Top 1 | Top 5 | Top 10 | Top 20 | Top 50 | Top 100 |
|---|---|---|---|---|---|---|---|---|
| – | N. | SG | 0.358 | 0.236 | 0.179 | 0.130 | 0.084 | 0.059 |
| | | Para | 0.560 | 0.469 | 0.412 | 0.352 | 0.279 | 0.230 |
| | | POS | 0.645 | 0.579 | 0.537 | 0.490 | 0.423 | 0.374 |
| | V. | SG | 0.302 | 0.206 | 0.162 | 0.126 | 0.086 | 0.065 |
| | | Para | 0.513 | 0.445 | 0.407 | 0.358 | 0.304 | 0.264 |
| | | POS | 0.582 | 0.526 | 0.487 | 0.444 | 0.396 | 0.357 |
| | Adj. | SG | 0.345 | 0.206 | 0.156 | 0.115 | 0.069 | 0.046 |
| | | Para | 0.562 | 0.417 | 0.363 | 0.304 | 0.231 | 0.185 |
| | | POS | 0.600 | 0.480 | 0.431 | 0.368 | 0.295 | 0.247 |
| 1911 | N. | SG | 0.358 | *0.225* | *0.175* | 0.132 | 0.084 | 0.058 |
| | | Para | 0.568 | 0.472 | 0.418 | **0.361** | **0.286** | **0.234** |
| | | POS | 0.659 | **0.588** | **0.548** | **0.501** | **0.431** | **0.382** |
| | V. | SG | 0.310 | 0.207 | 0.163 | 0.124 | 0.086 | 0.064 |
| | | Para | **0.550** | 0.456 | 0.414 | 0.362 | 0.307 | **0.268** |
| | | POS | 0.605 | 0.533 | **0.500** | **0.455** | **0.401** | **0.362** |
| | Adj. | SG | 0.343 | **0.209** | 0.157 | 0.114 | 0.069 | 0.046 |
| | | Para | 0.563 | 0.422 | 0.365 | 0.304 | **0.232** | 0.184 |
| | | POS | 0.602 | 0.484 | 0.431 | 0.368 | 0.296 | 0.247 |
| 1987 | N. | SG | 0.359 | *0.229* | 0.177 | **0.134** | 0.085 | 0.059 |
| | | Para | 0.564 | 0.471 | **0.419** | **0.365** | **0.285** | **0.234** |
| | | POS | 0.651 | 0.584 | **0.549** | **0.501** | **0.430** | **0.381** |
| | V. | SG | 0.308 | 0.211 | 0.167 | 0.127 | 0.087 | 0.064 |
| | | Para | 0.525 | **0.457** | **0.417** | 0.362 | 0.305 | **0.266** |
| | | POS | 0.588 | **0.537** | **0.499** | **0.453** | 0.399 | **0.360** |
| | Adj. | SG | 0.343 | 0.208 | 0.158 | 0.115 | 0.069 | 0.046 |
| | | Para | 0.565 | **0.421** | 0.365 | 0.304 | 0.232 | 0.184 |
| | | POS | 0.603 | 0.483 | 0.431 | 0.367 | 0.296 | 0.247 |

Table 2: Evaluation results for the combined measure with PMI. Significant improvement over unsupervised PMI in **bold**, significantly worse results in *italics*

*Thesaurus* as training data. Statistically significant improvement over the baseline appears in bold, while significantly worse results are italicised; we applied Student's t-test. With a few small exceptions, we found that the supervised system performs better. The number of times the scores were better, unchanged, or worse can be found in Table 3. In general, we concluded that the combination of supervised and unsupervised context weighting created a superior MSR, better suited to

Table 3:
The number of statistically improved/unaffected/ decreased results for both sources of training data

| Resource | Nouns | Verbs | Adjectives | All |
|---|---|---|---|---|
| 1911 *Roget's* | 8/8/2 | 6/12/0 | 2/16/0 | 16/36/2 |
| 1987 *Roget's* | 9/8/1 | 7/11/0 | 1/17/0 | 17/36/1 |

updating *Roget's Thesaurus* than the unsupervised method alone. We used the supervised method of generating lists of related words when adding new terms to the *Thesaurus*.

## 4 PLACING NEW WORDS IN *ROGET'S THESAURUS*

In this section, we evaluate a variety of systems for adding new words to *Roget's Thesaurus*. The baseline method places a word in the same POS, Paragraph and Semicolon Group as its closest neighbour in the *Thesaurus*. We improve on this baseline using multiple words to deduce a better location or better locations.

### 4.1 *Methods of adding new words*

We took advantage of the hierarchy of *Roget's Thesaurus* to select the best place to add words. We found first the POS, then the Paragraph, then the SG.[9] We refer to the word to be added to *Roget's Thesaurus* as the *target* word. A word already in the *Thesaurus* may be an *anchor*, acting as a "magnet" for a given target. For every target word $t$, we generated a list of nearest neighbours $NN(t)$, along with similarity scores, and identified anchors using $NN(t)$.

We experimented with three methods, evaluated against the following baseline: the target $t$ is placed in the same POS, Paragraph and SG as $w_i$, where $w_i$ is the first word in $NN(t)$ found in *Roget's Thesaurus*. Since $w_i$ may be polysemous, $t$ can go into multiple locations in *Roget's Thesaurus*. Often $w_i$ will be $w_1$ if the first neighbour of $t$ is found in the *Thesaurus*. For the values in Table 4, this baseline has been calculated using the MSRs built with combined weighting, trained with the 1911 or the 1987 *Thesaurus*. The results show one number for the count of POSs, Paragraphs and SGs where the target $t$ was placed and the precision of placing the word into the POSs, Paragraphs and SGs.

[9] Identifying the POS effectively gives us the correct Head as well.

The first method is to apply a nearest-neighbour model. $X$ nearest neighbours from $NN(t)$ are identified for each target word $t$. If $W$ of these $X$ words appear in the same *Roget's grouping*, the target word is placed there. It is a weakness that this method considers – somewhat unrealistically – the same number of neighbours for every target word.

In the second method, scores replace rank. Words with scores of $Y$ or higher are identified. If $W$ of them are in the same *Roget's grouping*, the target word is placed there. This allows for varying numbers of neighbours, but similarity scores partially depend on the target word, so the same score between two different word pairs may indicate different degrees of similarity. A very frequent word which appears in many contexts may have more highly related neighbours than a word which appears in few contexts. Such a frequent word may thus have inordinately many synonyms.

The third method considers relative scores. It assumes that the first similar word $w_1$ is very closely related to $t$, then takes all synonyms within $Z$% of the similarity score for $w_1$. This means that if $w_i$ has a score of within $Z$% of $w_1$, then it can be used as an anchor of $t$ for determining the correct *Roget's grouping*. Once again, if $W$ of these words in the same *Roget's grouping* have a relative score of $Z$% or higher, then the target word can be placed there as well.

We also considered how to optimise the measures. In placing words into a *Roget's grouping*, the method has two parameters to optimise, $W$ and one of $X$, $Y$ or $Z$. One possibility is to base F-measure on the precision with which words are placed in *Roget's Thesaurus* and recall on the number of words from the test set which could actually be placed. Another possibility of counting recall would be to identify the number of places where a word appears in the *Thesaurus* and see in how many of them it *was* placed. This measure has some problems.

For one, rare senses are not well represented by the vectors in the word–context matrix, so synonyms for only the most dominant senses will be found. Also, an even balance of precision and recall is not appropriate for this task. Adding incorrect words could be quite detrimental, so we assume that identifying the POS must weight precision more highly than recall. We set a 0.33 ratio of recall to precision (an F0.33 measure rather than F1). Once the POS has been identified, the Paragraph and SG will be identified using the F1 measure. The

Table 4: Baseline for identifying the POS of a word on the tuning and test data

| Year | POS | Data | Words | P | R | F0.33 |
|---|---|---|---|---|---|---|
| 1987 | Noun | *Tuning* | *1000* | *0.281* | *0.486* | *0.293* |
| | | Test | 1000 | 0.295 | 0.487 | 0.307 |
| | Verb | *Tuning* | *600* | *0.204* | *0.468* | *0.216* |
| | | Test | 600 | 0.245 | 0.455 | 0.257 |
| | Adjective | *Tuning* | *600* | *0.250* | *0.460* | *0.262* |
| | | Test | 600 | 0.232 | 0.435 | 0.244 |
| 1911 | Noun | *Tuning* | *817* | *0.232* | *0.296* | *0.237* |
| | | Test | 840 | 0.267 | 0.344 | 0.273 |
| | Verb | *Tuning* | *542* | *0.167* | *0.271* | *0.174* |
| | | Test | 538 | 0.196 | 0.297 | 0.203 |
| | Adjective | *Tuning* | *489* | *0.246* | *0.288* | *0.249* |
| | | Test | 497 | 0.201 | 0.262 | 0.206 |

choice of F0.33 is somewhat arbitrary, but favouring precision over recall should mostly bring advantages. A high-precision system is, in theory, more likely to place words in the correct *Roget's grouping* at the cost of lower recall. Any method of adding new words to *Roget's Thesaurus*, however, could be run iteratively and thus make up for the lower recall. Rather than attempting to add a lot of words in one pass, our method will add fewer words in each of multiple passes.

When using this method to actually add new words, sometimes it is necessary to create new Paragraphs or SGs. If a POS is identified but no Paragraph, then a new Paragraph will be created. Likewise, if a Paragraph but not an SG can be identified, then the word is placed in a new SG in the selected Paragraph.

The methods were tuned on the same dataset as that used to evaluate the MSR in Section 3. For evaluation, we constructed a test set equal in size to the tuning set. We evaluated all methods on the task of identifying the correct POS to place a target word $t$. The best method is then applied to the task of placing a word in the appropriate Paragraph and SG.

### 4.2 *Baseline*

Table 4 shows the results of the baseline experiments, measured for the 1911 and 1987 versions of *Roget's Thesaurus*. The former did not contain all the words for evaluation that the latter did – hence

| | | 1987 | | 1911 | |
|---|---|---|---|---|---|
| Parameter | POS | $X/Y/Z$ | $W$-POS | $X/Y/Z$ | $W$-POS |
| $X$ | Noun | 26 | 10 | 10 | 4 |
| | Verb | 22 | 7 | 6 | 3 |
| | Adjective | 19 | 6 | 8 | 3 |
| $Y$ | Noun | .08 | 15 | .07 | 14 |
| | Verb | .09 | 9 | .13 | 2 |
| | Adjective | .13 | 3 | .10 | 4 |
| $Z$ | Noun | .82 | 4 | .93 | 2 |
| | Verb | .89 | 3 | .98 | 2 |
| | Adjective | .82 | 3 | .91 | 2 |

Table 5: Optimal values for parameters $X$ (the number of nearest neighbours), $Y$ (the minimal relatedness score) and $Z$ (the relative score)

| Year | POS | Data | Words | P | R | F0.33 |
|---|---|---|---|---|---|---|
| 1987 | Noun | *Tuning* | *1000* | *0.746* | *0.267* | *0.633* |
| | | Test | 1000 | 0.758 | 0.262 | 0.637 |
| | Verb | *Tuning* | *600* | *0.565* | *0.285* | *0.514* |
| | | Test | 600 | 0.536 | 0.252 | 0.482 |
| | Adjective | *Tuning* | *600* | *0.658* | *0.273* | *0.577* |
| | | Test | 600 | 0.590 | 0.233 | 0.512 |
| 1911 | Noun | *Tuning* | *817* | *0.613* | *0.171* | *0.488* |
| | | Test | 840 | 0.659 | 0.182 | 0.522 |
| | Verb | *Tuning* | *542* | *0.484* | *0.131* | *0.381* |
| | | Test | 538 | 0.471 | 0.097 | 0.340 |
| | Adjective | *Tuning* | *489* | *0.571* | *0.184* | *0.472* |
| | | Test | 497 | 0.503 | 0.141 | 0.400 |

Table 6: Precision, Recall and F0.33-measure when optimising for $X$, the number of nearest neighbours

the differences in word counts. The results show a small advantage of adding words to the 1987 *Thesaurus* over the 1911 version.

### 4.3 *Tuning parameters for adding new words*

Table 5 shows the parameters, optimised for F0.33, for the three non-baseline methods. Tables 6–8 present the results on the tuning and test data.

When optimising for the $X$ nearest neighbours (Table 6), the results show a large improvement over the baseline (Table 4). The results for nouns were actually better on the test dataset than on tuning data,

Table 7: Precision, Recall and F0.33-measure when optimising for $Y$, the minimal relatedness score

| Year | POS | Data | Words | P | R | F0.33 |
|---|---|---|---|---|---|---|
| 1987 | Noun | *Tuning* | *1000* | *0.596* | *0.182* | *0.486* |
| | | Test | 1000 | 0.507 | 0.160 | 0.417 |
| | Verb | *Tuning* | *600* | *0.477* | *0.078* | *0.316* |
| | | Test | 600 | 0.573 | 0.062 | 0.313 |
| | Adjective | *Tuning* | *600* | *0.529* | *0.122* | *0.396* |
| | | Test | 600 | 0.421 | 0.103 | 0.322 |
| 1911 | Noun | *Tuning* | *817* | *0.420* | *0.120* | *0.336* |
| | | Test | 840 | 0.367 | 0.110 | 0.297 |
| | Verb | *Tuning* | *542* | *0.211* | *0.096* | *0.189* |
| | | Test | 538 | 0.234 | 0.063 | 0.184 |
| | Adjective | *Tuning* | *489* | *0.480* | *0.084* | *0.326* |
| | | Test | 497 | 0.274 | 0.066 | 0.209 |

but somewhat worse for verbs and adjectives. As with the baseline, the results were better for the 1987 *Roget's Thesaurus* than the 1911 version. Generally about one third to half of the words found in the top $X$ needed to be present in the same *Roget's grouping* in order to accurately select the correct grouping.

Table 7 shows optimising word placement with scores $Y$ or higher. The optimal scores were noticeably lower than when we optimised for $X$ nearest neighbours (Table 6). The minimum score $Y$ appeared to be lower for nouns than for verbs or adjectives, though more words were required in order to identify the *Roget's grouping* positively. This method is not as successful as simply selecting the $X$ nearest neighbours. For verbs added to the 1911 *Roget's Thesaurus*, there was actually no improvement over the baseline (Table 4). This is the least successful method of the three.

Table 8 reports on optimising for the relative score $Z$. We found that most neighbouring words had to be within 80–90% of the closest neighbour in terms of score. This improved the results noticeably over a simple selection of a hard score cut-off (Table 7). Nonetheless, we did not improve on simply taking the $X$ nearest neighbours (Table 6). For determining relatedness, it would appear, rank is often a feature more important than score. With this in mind, we applied the nearest-neighbour function using $X$ to find the best parameters for identifying the POS, Paragraph and SG. The parameter $W$ shown in Table 5 was

| Year | POS | Data | Words | P | R | F0.33 |
|---|---|---|---|---|---|---|
| 1987 | Noun | *Tuning* | *1000* | *0.643* | *0.190* | *0.519* |
| | | Test | 1000 | 0.595 | 0.215 | 0.506 |
| | Verb | *Tuning* | *600* | *0.468* | *0.147* | *0.384* |
| | | Test | 600 | 0.492 | 0.163 | 0.410 |
| | Adjective | *Tuning* | *600* | *0.512* | *0.215* | *0.450* |
| | | Test | 600 | 0.463 | 0.200 | 0.409 |
| 1911 | Noun | *Tuning* | *817* | *0.468* | *0.200* | *0.413* |
| | | Test | 840 | 0.542 | 0.219 | 0.473 |
| | Verb | *Tuning* | *542* | *0.438* | *0.118* | *0.344* |
| | | Test | 538 | 0.389 | 0.091 | 0.293 |
| | Adjective | *Tuning* | *489* | *0.478* | *0.145* | *0.389* |
| | | Test | 497 | 0.434 | 0.129 | 0.351 |

Table 8:
Precision, Recall and F0.33-mesure when optimising for $Z$, the relative score

| Year | POS | $X$ | $W$-POS | $W$-Para | $W$-SG |
|---|---|---|---|---|---|
| 1987 | Noun | 26 | 10 | 5 | 2 |
| | Verb | 22 | 7 | 4 | 3 |
| | Adjective | 19 | 6 | 4 | 2 |
| 1911 | Noun | 10 | 4 | 3 | 3 |
| | Verb | 6 | 3 | 2 | 2 |
| | Adjective | 8 | 3 | 2 | 2 |

Table 9:
Optimal parameters for $X$ (the number of nearest neighbours) and $W$ (neighbours needed to insert a word into a *Roget's grouping*) at the POS, Paragraph and SG levels

for the POS level. We have three versions, $W$-POS, $W$-Para and $W$-SG for the POS, Paragraph and SG respectively.

Table 9 shows the optimal values of $X$, $W$-POS, $W$-Para and $W$-SG. The same value of $X$ was used for identifying groupings at the POS, Paragraph and SG levels. There is a bit of variance in the measures. The values of $W$-POS, $W$-Para and $W$-SG decrease as the groupings become smaller. To identify the correct SG, only 2 or 3 words were used. For the 1911 *Roget's Thesaurus*, the same number of words were used to identify the Paragraph as the SG. More words could be used to identify the POS for the 1987 *Thesaurus* than for the 1911 version.

Tables 10 and 11 show the precision, recall and F1 measure at the POS, Paragraph and SG level for the 1987 and 1911 *Thesauri*. The results show clearly that the F1 measure is highest when identifying the Paragraph level; this is largely because the POS level is optimised for the F0.33 measure. Once again, the scores for the 1987 version

Table 10: Identifying best POS, Paragraph and SG using optimised values for $X$, $W$-POS, $W$-Para and $W$-SG, using the F1 measure for evaluation on the **1987** *Roget's Thesaurus*

| | Data | RG | P | R | F1 |
|---|---|---|---|---|---|
| Noun | *Tuning* | *POS* | *306/410 (0.746)* | *267/1000 (0.267)* | *0.393* |
| | *Tuning* | *Para* | *225/402 (0.560)* | *189/267 (0.708)* | *0.625* |
| | *Tuning* | *SG* | *104/664 (0.157)* | *92/189 (0.487)* | *0.237* |
| | Test | POS | 304/401 (0.758) | 262/1000 (0.262) | 0.389 |
| | Test | Para | 234/416 (0.562) | 196/262 (0.748) | 0.642 |
| | Test | SG | 101/659 (0.153) | 93/196 (0.474) | 0.232 |
| Verb | *Tuning* | *POS* | *227/402 (0.565)* | *171/600 (0.285)* | *0.379* |
| | *Tuning* | *Para* | *186/413 (0.450)* | *137/171 (0.801)* | *0.577* |
| | *Tuning* | *SG* | *34/129 (0.264)* | *32/137 (0.234)* | *0.248* |
| | Test | POS | 185/345 (0.536) | 151/600 (0.252) | 0.343 |
| | Test | Para | 148/339 (0.437) | 114/151 (0.755) | 0.553 |
| | Test | SG | 18/103 (0.175) | 17/114 (0.149) | 0.161 |
| Adj | *Tuning* | *POS* | *227/345 (0.658)* | *164/600 (0.273)* | *0.386* |
| | *Tuning* | *Para* | *182/312 (0.583)* | *136/164 (0.829)* | *0.685* |
| | *Tuning* | *SG* | *75/381 (0.197)* | *63/136 (0.463)* | *0.276* |
| | Test | POS | 193/327 (0.590) | 140/600 (0.233) | 0.334 |
| | Test | Para | 152/294 (0.517) | 116/140 (0.829) | 0.637 |
| | Test | SG | 59/351 (0.168) | 51/116 (0.440) | 0.243 |

tend to be better than those for the 1911 version. Most of the time it is possible to identify the correct POS with at least 40% accuracy. The recall for the 1987 *Thesaurus* was 0.233 or higher at the POS level. This is important, because it indicates how many new word additions to the *Thesaurus* can be expected. For the 1911 *Thesaurus*, the results tend to be much lower, with scores from 0.097 to 0.182 on the test set. The number for verbs is very low; for nouns and adjectives it is better, but still lower than the corresponding results for the 1987 thesaurus.

### 4.4 *Adding words to the Thesaurus*

We now show how the method described in Section 4.3 adds words to *Roget's Thesaurus*. In practice, a few small modifications were needed. First, we only let a word be placed in a POS if it was not already present in either that POS or in another POS within the same Head Group. This reduced the possibility of entering antonyms, which may be distributionally similar, into the same POS. Within each POS, we let a word be placed only in one Paragraph. We also did not allow

| | Data | RG | P | R | F1 |
|---|---|---|---|---|---|
| Noun | *Tuning* | *POS* | *157/256 (0.613)* | *140/817 (0.171)* | *0.268* |
| | *Tuning* | *Para* | *89/163 (0.546)* | *83/140 (0.593)* | *0.568* |
| | *Tuning* | *SG* | *31/62 (0.500)* | *29/83 (0.349)* | *0.411* |
| | Test | POS | 162/246 (0.659) | 153/840 (0.182) | 0.285 |
| | Test | Para | 83/155 (0.535) | 78/153 (0.510) | 0.522 |
| | Test | SG | 29/55 (0.527) | 28/78 (0.359) | 0.427 |
| Verb | *Tuning* | *POS* | *76/157 (0.484)* | *71/542 (0.131)* | *0.206* |
| | *Tuning* | *Para* | *55/136 (0.404)* | *53/71 (0.746)* | *0.525* |
| | *Tuning* | *SG* | *24/86 (0.279)* | *24/53 (0.453)* | *0.345* |
| | Test | POS | 57/121 (0.471) | 52/538 (0.097) | 0.160 |
| | Test | Para | 39/112 (0.348) | 35/52 (0.673) | 0.459 |
| | Test | SG | 22/76 (0.289) | 19/35 (0.543) | 0.378 |
| Adj | *Tuning* | *POS* | *109/191 (0.571)* | *90/489 (0.184)* | *0.278* |
| | *Tuning* | *Para* | *80/188 (0.426)* | *71/90 (0.789)* | *0.553* |
| | *Tuning* | *SG* | *23/107 (0.215)* | *22/71 (0.310)* | *0.254* |
| | Test | POS | 79/157 (0.503) | 70/497 (0.141) | 0.220 |
| | Test | Para | 46/148 (0.311) | 42/70 (0.600) | 0.409 |
| | Test | SG | 14/91 (0.154) | 13/42 (0.310) | 0.206 |

Table 11: Identifying best POS, Paragraph and SG using optimised values for $X$, $W$-POS, $W$-Para and $W$-SG, using the F1 measure for evaluation on the **1911** *Roget's Thesaurus*

adding the same word to multiple SGs within the same Paragraph or indeed to multiple Paragraphs in the same POS.

Once a new word has been added to *Roget's Thesaurus*, it can be used as an anchor to help add subsequent words. We built two updated versions of each *Thesaurus*, one with a single pass to update the *Thesaurus*, another with five updating passes. We considered each word in each matrix, excluding stop words,[10] to be a target and generated a list of the nearest 100 neighbours for each of these words.[11] It was from these lists that we attempted to add new words to the *Thesaurus*.

Several measures are of interest when adding new words to the *Thesaurus*. The first is the number of times a target word has sufficient $X$ and $W$ values to be placed in *Roget's Thesaurus*, regardless of whether it was already present. The second measure is the total num-

[10] We applied a 980-element union of five stop lists first used in Jarmasz (2003): Oracle 8 ConText, SMART, Hyperwave, a list from the University of Kansas and a list from Ohio State University.

[11] Only the top $X$ of those 100 helped identify the best place for a new word.

ber of words added to the *Thesaurus*. The third measure is the number of unique words added. These two are likely to be similar since most often a target word is only added to a single location in the *Thesaurus*. The fourth measure counts new words whose derivational form already exists in the *Thesaurus*. The fifth measure counts new words which have no derivationally related words in the *Thesaurus*. The last measure is the number of Heads where a new word was added. The results for all five passes can be seen in Table 12.

In addition to the five passes of adding new words, we experimented with random addition. All process parameters are the same, up to the point when our system determines a location where it believes a word belongs. Before checking whether that word already appears at this location, it is swapped for a random word. The counts appear in Table 13. Since the random word is selected after a location has been decided, it is very rare for this word already to be in that Head Group. As a result, the number of attempted placements is very close to the total number of words added, much closer than for the counts from Table 12.

Ultimately three updated version each of the 1911 and 1987 versions of the *Thesaurus* were created, those updated with one pass, five passes and one random pass – X1, X5 and R in Table 14. The updated versions are referred to as 1911X1, 1911X5, 1911R, 1987X1, 1987X5 and 1987R. The new thesauri have been evaluated manually (Section 5) and through selected NLP applications (Section 6).

Another statistic to consider is the total number of words, SGs and Paragraphs added to each version of *Roget's Thesaurus*, shown in Table 14. Overall, some 5500 new words were added to 1911X5 and 9600 to 1987X5. In the 1911 *Thesaurus*, approximately two thirds of the new words were placed in a new SG, while about a quarter were added to a new Paragraph. For the 1987 *Thesaurus*, a little under half of the new words were placed in new SGs, while around one fifth were added to new Paragraphs.

## 5 MANUAL EVALUATION

To determine the quality of the additions reliably, one needs manual evaluation. In the next subsection, we describe several possibilities and explain how we chose our evaluation method.

| P | Year | POS | Matches | Total Words | Unique Words | Derived Words | New Words | Heads Affected |
|---|---|---|---|---|---|---|---|---|
| 1 | 1987 | Nouns | 6755 | 1510 | 1414 | 175 | 98 | 206 |
| | | Verbs | 2870 | 893 | 735 | 52 | 45 | 129 |
| | | Adj | 3053 | 858 | 713 | 15 | 10 | 183 |
| | 1911 | Nouns | 3888 | 1259 | 1193 | 148 | 68 | 274 |
| | | Verbs | 1069 | 407 | 378 | 22 | 19 | 133 |
| | | Adj | 1430 | 539 | 480 | 18 | 16 | 198 |
| 2 | 1987 | Nouns | 8388 | 774 | 742 | 37 | 14 | 139 |
| | | Verbs | 4335 | 747 | 653 | 23 | 16 | 92 |
| | | Adj | 4412 | 612 | 549 | 4 | 4 | 114 |
| | 1911 | Nouns | 5315 | 762 | 719 | 65 | 13 | 164 |
| | | Verbs | 1530 | 247 | 238 | 14 | 14 | 71 |
| | | Adj | 2083 | 287 | 262 | 6 | 5 | 95 |
| 3 | 1987 | Nouns | 9213 | 499 | 478 | 16 | 6 | 88 |
| | | Verbs | 5303 | 600 | 543 | 16 | 14 | 61 |
| | | Adj | 5275 | 532 | 463 | 7 | 2 | 80 |
| | 1911 | Nouns | 6109 | 549 | 520 | 35 | 11 | 100 |
| | | Verbs | 1761 | 147 | 142 | 6 | 6 | 36 |
| | | Adj | 2393 | 205 | 191 | 5 | 4 | 57 |
| 4 | 1987 | Nouns | 9767 | 384 | 378 | 11 | 2 | 60 |
| | | Verbs | 6068 | 523 | 496 | 11 | 9 | 49 |
| | | Adj | 5926 | 451 | 404 | 6 | 6 | 55 |
| | 1911 | Nouns | 6652 | 417 | 395 | 20 | 5 | 76 |
| | | Verbs | 1898 | 106 | 105 | 0 | 0 | 21 |
| | | Adj | 2571 | 139 | 129 | 1 | 0 | 35 |
| 5 | 1987 | Nouns | 10210 | 330 | 324 | 12 | 2 | 49 |
| | | Verbs | 6689 | 464 | 422 | 6 | 3 | 39 |
| | | Adj | 6509 | 424 | 382 | 3 | 1 | 38 |
| | 1911 | Nouns | 7026 | 295 | 288 | 22 | 10 | 54 |
| | | Verbs | 1979 | 76 | 74 | 0 | 0 | 14 |
| | | Adj | 2710 | 119 | 115 | 1 | 0 | 22 |

Table 12: New words added after the $1^{st}$, $2^{nd}$, $3^{rd}$, $4^{th}$ and $5^{th}$ pass (P)

Table 13: Random words added after one pass

| Year | POS | Matches | Total Words | Unique Words | Derived Words | New Words | Heads Affected |
|---|---|---|---|---|---|---|---|
| 1987 | Nouns | 6755 | 6189 | 5007 | 3923 | 3593 | 306 |
| | Verbs | 2870 | 2238 | 1366 | 734 | 715 | 186 |
| | Adj | 3053 | 2631 | 1670 | 1547 | 1488 | 278 |
| 1911 | Nouns | 3888 | 3718 | 3203 | 2736 | 2554 | 379 |
| | Verbs | 1069 | 946 | 759 | 468 | 465 | 195 |
| | Adj | 1430 | 1349 | 1051 | 952 | 926 | 276 |

Table 14: New Paragraphs, SGs and words in the updated versions of *Roget's Thesaurus*

| Resource | New Paragraphs | New SGs | New Words |
|---|---|---|---|
| 1911X1 | 633 | 1442 | 2209 |
| 1911X5 | 1851 | 3864 | 5566 |
| 1911R | 1477 | 3803 | 6018 |
| 1987X1 | 653 | 1356 | 3261 |
| 1987X5 | 2063 | 4466 | 9601 |
| 1987R | 1672 | 3731 | 11058 |

### 5.1 *Methods considered*

The first evaluation method would test how well people can identify newly added words. Given a set of Paragraphs from *Roget's Thesaurus*, the annotator would be asked to identify which words she thought were added automatically and which were originally in the *Thesaurus*. The percentage of times the annotator correctly identifies newly added words can be used to evaluate the additions. If a word already in the *Thesaurus* were as likely to be picked as one newly added, then the additions would be indistinguishable – an ideal outcome. We could also perform a "placebo test": the annotator gets a Paragraph where no words have been added, and decides whether to remove any words at all. A drawback is that the annotator may be more likely to select words whose meaning she does not know, especially in the 1911 *Thesaurus*, where there are many outdated words. Even the 1987 version has many words infrequently used today.

The second method of manual evaluation we considered was to ask the annotator to assign a new word to the correct location in the *Thesaurus*. A weighted edit-distance score could then tell how many steps the system's placement is from that location. We would also mea-

| Score | *Roget's* Paragraph |
|---|---|
| (word fits in this SG) | *Head 25: Agreement, noun*<br><br>fitness, aptness;<br>relevancy;<br>pertinence, pertinencey;<br>sortance;<br>case in point;<br>**aptitude, coaptation, propriety, applicability, admissibility, commensurability, *compatibility*;**<br>cognation. |

Figure 3: Example of the annotator task for adding a word to a Paragraph

sure how often the annotator needed to create a new Paragraph or SG for the word, and how many SGs and Paragraphs were automatically created but should not have been. Such a method would be labour-intensive: the annotator would need to read an entire Head before deciding how far a word is from its correct location. Larger Heads, where most new words are added, could contain thousands of words. Identifying whether there is an SG more appropriate for a given word could also take a fair bit of effort. It might not be feasible to annotate enough data to perform a meaningful evaluation.

The strategy we finally adopted combines elements of the two preceding methods. The first step of this evaluation exercise is to decide whether new words added to an existing SG or a new SG in an existing Paragraph are in the correct location. The annotator is given the name of the Head, the part of speech and the text of the Paragraph where the word has been added. The new term is specially highlighted, and other terms in its SG are in bold. The annotator is asked to decide whether the new word is in the correct SG, wrong SG but correct Paragraph, wrong Paragraph but correct Head, or incorrect Head. Figure 3 shows a sample question.

The second evaluation step determines whether a word added to a new Paragraph is in the correct Head. As context, we provide the first word in every Paragraph in the same POS. It is too onerous to determine precisely in which SG or Paragraph a new word would belong, because some POSs are very large. Instead, we only ask whether the word is in the correct Head. A sample question appears in Figure 4.

Figure 4: Example of the annotator task for adding a word to a POS

| Score | *Roget's* Paragraph |
|---|---|
| (closely related) | *Head 25: Agreement, noun*<br><br>agreement.. / conformity.. / fitness.. / adaption.. / ***consent***; |

We manually evaluated only the additions to the 1911 *Roget's Thesaurus*. As Paragraph size, we allowed at most 250 characters, thus limiting the number of words the annotators had to look at. The evaluation was completed by the first author and four volunteers. We chose enough samples to guarantee a 5% confidence interval at a 95% confidence level.[12] We also included a high baseline and a low baseline: words already present in the *Thesaurus*[13] and words randomly added to it. There are enough samples from the baselines to guarantee a 5% confidence interval at a 95% confidence level if the samples from all three parts of speech are combined, though individually the confidence interval exceeds 5%.

Every new word in 1911X1 appears in 1911X5,[14] so a percentage of the samples needed to evaluate 1911X5 can be selected from the samples used to evaluate 1911X1. We thus must evaluate only a selection of the words from 1911X5 not present in 1911X1. We randomly selected words from the sample set for 1911X1 to make up the rest of the samples for the 1911X5 evaluation.

Random selection was made from each annotator's dataset: 40 tests for adding words to existing Paragraphs and 40 tests for adding words to new Paragraphs. These data points were added to each annotator's test sets so that there would be an overlap of 200 samples for each experiment, on which to calculate inter-annotator agreement. The positive examples are words already present in *Roget's Thesaurus*. The negative examples are words randomly placed in the *Thesaurus*.

[12] `http://www.macorr.com/sample-size-calculator.htm`

[13] They are referred to as "pre-existing" in Tables 15–16, in Figures 5–6 and in the discussion in Section 5.2

[14] We remind the reader that X1 and X5 denote updating with one pass and five passes respectively.

### 5.2 *Manual annotation results*

Tables 15 and 16 show the combined manual annotation results for new words added to existing Paragraphs and for new Paragraphs. A number of interesting observations can be taken from Table 15. The results are summarised in Figure 5. In the case of pre-existing examples, around 60% of the time the annotators could correctly determine when a word belonged in the SG in which it was found. The annotators agreed on the correct Head approximately 80–90% of the time. One reason why annotators might believe the words belonged in a different grouping was that many of the words were difficult to understand. A high number of words which the annotators could not label fell into the pre-existing category. For the randomly assigned words, 70–80% of the time the annotators correctly stated that those words did not belong in that Head. For nouns there were numerous cases when the annotators could not answer. It would appear that the meaning of words pre-existing in the *Thesaurus*, and of those randomly added, is harder to determine than the meaning of automatically added words.

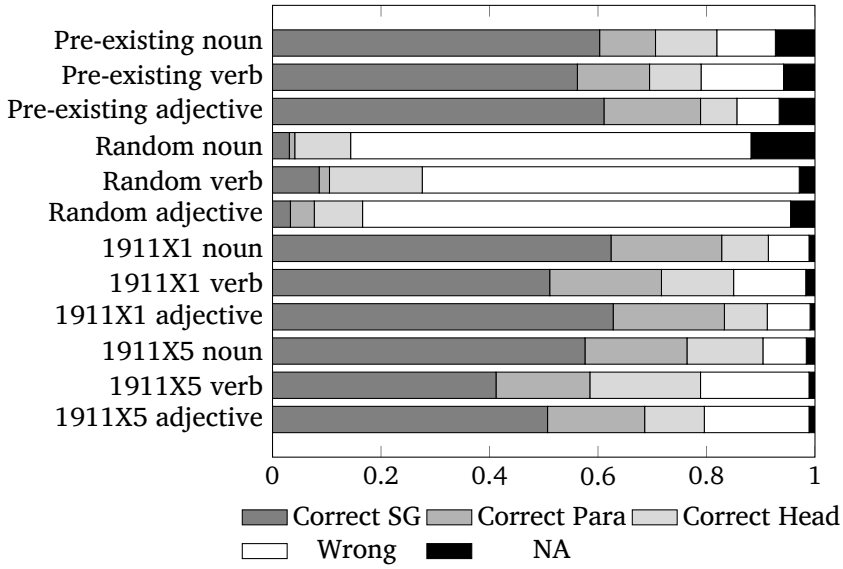
We now turn to the quality of additions. The distribution of 1911X1 scores in Table 15 is very close to that of the distribution for words pre-existing in *Roget's Thesaurus*. This suggests that after one pass the added words are nearly indistinguishable from those already in the *Thesaurus*. This is very good news: it confirms that our process of updating the lexicon has succeeded. The distribution of 1911X5 scores suggests that those additions were less reliable. The scores are worse than for 1911X1, but still much closer to the pre-existing baseline than the random baseline. Multiple passes increase the error, but not by much.

The results are a bit different when it comes to inserting words into new Paragraphs. These results are summarised in Figure 6. Once again the high and low baselines appeared to be fairly easy for the annotators, who usually got around 80% of the questions right. Also, a solid majority of the unknown words appeared in these two groups. The additions to 1911X1 showed high scores, too, comparable to the high baseline, sometimes even exceeding it slightly. It may be that for this baseline the annotators were unaware of the sense of some words, so they mistakenly labelled those words as incorrect.

Table 15: Manual evaluation results for words added to previously existing Paragraphs

| Task | POS | Correct SG | Correct Para | Correct Head | Wrong Head | N/A |
|---|---|---|---|---|---|---|
| Pre-existing Words | Noun | 117 (*.600*) | 20 (*.103*) | 22 (*.113*) | 21 (*.108*) | 15 (*.077*) |
| | Verb | 59 (*.562*) | 14 (*.133*) | 10 (*.095*) | 16 (*.152*) | 6 (*.057*) |
| | Adj. | 55 (*.611*) | 16 (*.178*) | 6 (*.067*) | 7 (*.078*) | 6 (*.067*) |
| Random Words | Noun | 6 (*.031*) | 2 (*.010*) | 20 (*.103*) | 144 (*.738*) | 23 (*.118*) |
| | Verb | 9 (*.086*) | 2 (*.019*) | 18 (*.171*) | 73 (*.695*) | 3 (*.029*) |
| | Adj. | 3 (*.033*) | 4 (*.044*) | 8 (*.089*) | 71 (*.789*) | 4 (*.044*) |
| 1911X1 | Noun | 159 (*.624*) | 52 (*.204*) | 22 (*.086*) | 19 (*.075*) | 3 (*.012*) |
| | Verb | 92 (*.511*) | 37 (*.206*) | 24 (*.133*) | 24 (*.133*) | 3 (*.017*) |
| | Adj. | 135 (*.628*) | 44 (*.205*) | 17 (*.079*) | 17 (*.079*) | 2 (*.009*) |
| 1911X5 | Noun | 181 (*.576*) | 59 (*.188*) | 44 (*.140*) | 25 (*.080*) | 5 (*.016*) |
| | Verb | 107 (*.412*) | 45 (*.173*) | 53 (*.204*) | 52 (*.200*) | 3 (*.012*) |
| | Adj. | 147 (*.507*) | 52 (*.179*) | 32 (*.110*) | 56 (*.193*) | 3 (*.010*) |

Figure 5: Evaluation on words added to previously existing Paragraphs in the 1911 *Roget's Thesaurus*



| Task | POS | Correct Head | Wrong Head | N/A |
|---|---|---|---|---|
| Pre-existing Words | Noun | 158 ( *.810)* | 33 ( *.169)* | 4 ( *.021)* |
| | Verb | 87 ( *.829)* | 17 ( *.162)* | 1 ( *.010)* |
| | Adj | 75 ( *.833)* | 14 ( *.156)* | 1 ( *.011)* |
| Random Words | Noun | 18 ( *.092)* | 151 ( *.774)* | 26 ( *.133)* |
| | Verb | 17 ( *.162)* | 83 ( *.790)* | 5 ( *.048)* |
| | Adj | 13 ( *.144)* | 74 ( *.822)* | 3 ( *.033)* |
| 1911X1 | Noun | 189 ( *.859)* | 27 ( *.123)* | 4 ( *.018)* |
| | Verb | 50 ( *.833)* | 10 ( *.167)* | 0 ( *.000)* |
| | Adj | 48 ( *.873)* | 7 ( *.127)* | 0 ( *.000)* |
| 1911X5 | Noun | 207 ( *.674)* | 94 ( *.306)* | 6 ( *.020)* |
| | Verb | 64 ( *.533)* | 55 ( *.458)* | 1 ( *.008)* |
| | Adj | 61 ( *.616)* | 37 ( *.374)* | 1 ( *.010)* |

Table 16: Manual evaluation results for words added to new Paragraphs

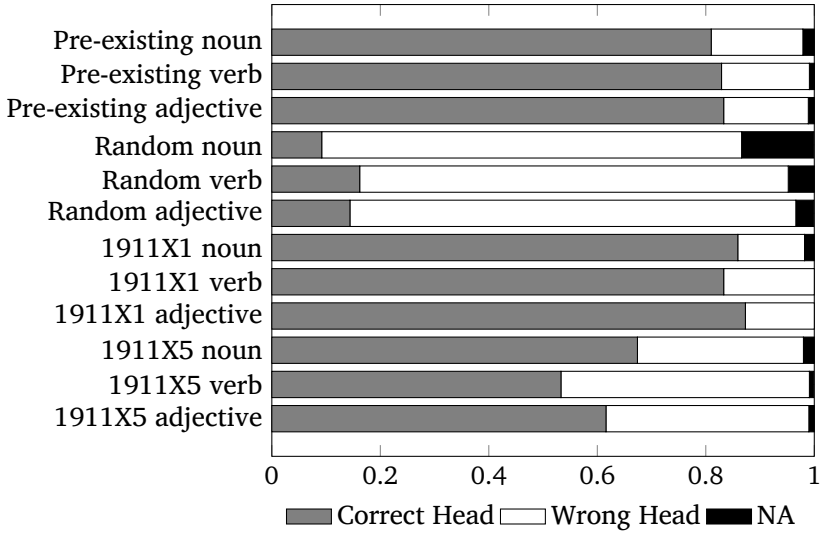



Figure 6: Evaluation on words added to new Paragraphs in the 1911 *Roget's Thesaurus*

The 1911X5 results – multi-pass update – clearly fall a fair distance from the scores for 1911X1. It would appear that multiple passes introduce considerable error into the *Thesaurus*, when words are placed into new Paragraphs. This is in stark contrast to the result of adding words to existing Paragraphs, when the drop in scores between 1911X1 and 1911X5 was relatively small.

### 5.3 *Inter-annotator agreement*

Each annotator was given 200 examples which reoccurred between the annotations. Inter-annotator agreement was measured on these overlaps, using Krippendorff's $\alpha$ (Krippendorff 2004), a measure designed to work with various kinds of data, including nominal, ordinal and interval annotations. We used ordinal in our experiments. The value of $\alpha$ was calculated for adding words both to existing Paragraphs and to new Paragraphs. When adding words to an existing Paragraph, we obtained a score of $\alpha = 0.340$; when adding words to new Paragraphs, the score was $\alpha = 0.358$. Such scores are often considered a "fair" amount of agreement (Landis and Koch 1977).

## 6 AUTOMATIC EVALUATION

We now examine how the various versions of *Roget's Thesaurus*, as well as *WordNet* 3.0, perform on several NLP applications. The problems we selected are designed to evaluate *Roget's Thesaurus* on a diverse cross-section of NLP tasks: synonym identification, pseudo-word-sense disambiguation and SAT-style analogy problems. We use *WordNet* 3.0 and all available versions of the *Thesaurus*: 1911, 1911X1, 1911X5, 1911R, 1987, 1987X1, 1987X5 and 1987R. Although the updated versions of *Roget's Thesaurus* are larger than the original, and new words have been added with relatively high precision, there is no *a priori* guarantee that they will give higher scores on any NLP applications. Before we harness these resources into NLP applications, we will very briefly compare the structure of *Roget's Thesaurus* to that of *WordNet*.[15]

A major difference between *WordNet* and *Roget's Thesaurus* is that the former is built around a hypernym hierarchy of arbitrary depth. Words appear at all levels, rather than only at the bottom level, as in

[15] For a detailed presentation of *WordNet*, see (Fellbaum 1998).

*Roget's Thesaurus*. Words are grouped into synsets. Synsets are similar to SGs in the *Thesaurus*, but are often smaller and contain only close synonyms. Synsets are linked by a variety of explicitly named semantic relations, while in the *Thesaurus* the SGs in a Paragraph are loosely related by a variety of possible implicit relations.

### 6.1 *Synonym identification*

Synonym identification is a means of evaluating the quality of newly added words in *Roget's Thesaurus*. In this problem one is given a term $q$ and seeks its best synonym $s$ in a set of words $C$. The system from Jarmasz and Szpakowicz (2003b, 2004) identifies synonyms using the *Thesaurus* as the lexical resource. This method relies on a simple function which counts the number of edges in the *Thesaurus* between $q$ and words in $C$. In Equation 1, 18 is the highest possible distance in the *Thesaurus*, so the closest words have the highest scores (*edgesBetween* simply counts the edges). We treat a word $X$ as a lexeme: a set of word senses $x \in X$.

$$edgeScore(X, Y) = \max_{x \in X,\, y \in Y} [18 - edgesBetween(x, y)] \tag{1}$$

The best synonym is selected in two steps. First, we find a set of terms $B \subseteq C$ with the maximum relatedness between $q$ and each word sense $x \in C$ (Equation 2).

$$B = \left\{ x \mid \operatorname*{argmax}_{x \in C} edgeScore(x, q) \right\} \tag{2}$$

Next, we take the set of terms $A \subseteq B$ where each $a \in A$ has the largest number of shortest paths between $a$ and $q$ (Equation 3).

$$A = \left\{ x \mid \operatorname*{argmax}_{x \in B} numberOfShortestPaths(x, q) \right\} \tag{3}$$

The correct synonym $s$ has been selected if $s \in A$ and $|A| = 1$. Often the sets $A$ and $B$ will both contain one item, but if $s \in A$ and $|A| > 1$, there is a tie. If $s \notin A$, the selected synonyms are incorrect. If an n-word phrase $c \in C$ is found, its words $c_1, c_2 ..., c_n$ are considered in turn; the $c_i$ closest to $q$ is chosen to represent $c$. A sought word can be of any part of speech, though only some *WordNet*-based methods allow for adjectives or adverbs, and none can measure distance between different parts of speech. In these problems, we do not consider a word and its morphological variant to be the same.

We generated synonym selection problems specifically for words newly added to *Roget's Thesaurus*. We took all words which appeared either in 1987X5 or in 1911X5, but were not present in the original 1987 or 1911 versions, and used them as query words $q$ for the new problems. We then found in *WordNet* synsets which contain at least one of $q$'s synonyms found in the original (not updated) version of the *Thesaurus*. We completed the problem by finding in the original *Thesaurus* three detractors from $q$'s co-hyponym synsets. This was done for nouns and for verbs, but not for adjectives, for which *WordNet* does not have a strong hypernym hierarchy.

Four different versions of this problem were generated for the 1911 and 1987 *Roget's Thesauri* using nouns and verbs. The linking structure for adjectives in *WordNet* precludes the creation of a data set in this manner. We present the final scores as precision and recall. The precision excludes questions where $q$ is not in *Roget's Thesaurus*, and recall is the score over the entire data set. Precision is thus the proportion of correct guesses out of the questions attempted, while recall is the proportion of correct guesses out of the maximum number of attempted questions. This method of evaluating such work was proposed by Turney (2006).

Table 17 shows the results for nouns and verbs added to both the 1987 and the 1911 versions of *Roget's Thesaurus*. The results are quite similar for all four data sets. Obviously, a precision and recall of 0 is attained for the original versions of the *Thesaurus*. The randomly updated versions did poorly as well. Versions updated after one pass had recall between 18% and 26%, while the versions updated in 5 passes had 40% or more. The random baseline is 25% if all of the questions can be answered. The thesauri updated in 5 passes significantly beat this baseline.[16] The thesauri updated in one pass tended not to show statistically significant improvement, though many problems were unsolvable ($q$ was absent from 1911X1 or 1987X1).

The recall improvement for *Roget's Thesaurus* updated in 5 passes was significantly better (at $p < 0.05$) than for the *Thesaurus* updated in one pass. In turn, the *Thesaurus* updated in one pass was significantly better than the original *Thesaurus* (again at $p < 0.05$). The exception was the 1911 verb data set, for which the improvement could only

[16] Significance was established with Student's T-test with $p < 0.05$.

| | Resource | Correct | Wrong | Ties | N/A | Precision | Recall |
|---|---|---|---|---|---|---|---|
| | 1911 | 0 | 98 | 0 | 98 | 0 | 0 |
| 1911 | 1911X1 | 18 | 70 | 10 | 44 | 40.13 | 22.11 |
| Nouns | 1911X5 | 30 | 45 | 23 | 0 | 39.63 | 39.63 |
| | 1911R | 3 | 93 | 2 | 88 | 39.98 | 4.08 |
| | 1911 | 0 | 27 | 0 | 27 | 0 | 0 |
| 1911 | 1911X1 | 6 | 20 | 1 | 13 | 46.42 | 24.07 |
| Verbs | 1911X5 | 11 | 14 | 2 | 0 | 44.44 | 44.44 |
| | 1911R | 0 | 27 | 0 | 26 | 0 | 0 |
| | 1987 | 0 | 57 | 0 | 57 | 0 | 0 |
| 1987 | 1987X1 | 11 | 38 | 8 | 18 | 38.03 | 26.02 |
| Nouns | 1987X5 | 18 | 29 | 10 | 0 | 39.77 | 39.77 |
| | 1987R | 0 | 56 | 1 | 52 | 10.03 | 0.88 |
| | 1987 | 0 | 36 | 0 | 36 | 0 | 0 |
| 1987 | 1987X1 | 5 | 27 | 4 | 20 | 41.67 | 18.52 |
| Verbs | 1987X5 | 12 | 15 | 9 | 0 | 44.91 | 44.91 |
| | 1987R | 1 | 35 | 0 | 29 | 14.29 | 2.78 |

Table 17: Evaluation of identifying synonyms from *WordNet*

be measured as significant at $p < 0.065$. This is largely because the dataset was fairly small. Another observation is that the randomly updated *Thesaurus* only once had a significant improvement over the original *Thesaurus*, in the case of the 1911 noun data set.

These results suggest that the words newly added to *Roget's Thesaurus* are close to the correct location. The newly added words and their synonyms were closer than the newly added words and their cohyponyms. Generally the precision measure showed words added to the 1911X1 and 1987X1 thesauri to be approximately as accurate as, if not slightly more accurate than, those added in passes 2–5. The randomly updated *Thesaurus* did not perform as well, usually falling below the 25% baseline on the precision measure. The results for nouns added to the 1911 *Thesaurus* are a noticeable exception. In the other datasets at most one question was answered correctly by the randomly updated *Thesaurus*, but in this case there were three correct answers. It should be noted, however, that the evaluated sample was very small, so this is likely to have been a coincidence.

## 6.2 Pseudo-word-sense disambiguation

Pseudo-word-sense disambiguation (PWSD) is a somewhat contrived task, meant to evaluate the quality of a word-sense disambiguation (WSD) system. The set-up for this task is to take two words and merge them into a *pseudo-word*. A WSD system, then, has the goal of identifying which of the two words actually belongs in a given context in which the whole pseudo-word appears. We have had a chance to create a very large dataset for PWSD. This is an opportunity to consider *WordNet* and the versions of *Roget's Thesaurus* in PWSD, and to compare them not only for accuracy but also for runtime.

We used PWSD instead of real WSD for two main reasons. Firstly, as far as we know, there is no WSD data set annotated with *Roget's* word senses and so one would have to be built from scratch. Worse still, to compare WSD systems built using *Roget's Thesaurus* and *WordNet* we would need a dataset labeled with senses from both. Secondly, PWSD gives us a fast way of building a dataset which can be used to evaluate the WSD systems based on the *Thesaurus* and on *WordNet*.

A common variation on this task is to make triples out of a noun and two verbs, then determine which of the verbs takes the noun as its object. The aim is to create a kind of verb disambiguation system which incorporates the edge count distance between nouns. In theory, this measure can help indicate how well a system identifies contexts (verb object) in which a verb appears. That can be useful in real WSD. Others who have worked on variations of PWSD include Gale *et al.* (1992); Schütze (1998); Lee (1999); Dagan *et al.* (1999); Rooth *et al.* (1999); Clark and Weir (2002); Weeds and Weir (2005); Zhitomirsky-Geffet and Dagan (2009). The methodology we followed was similar to that of Weeds and Weir.

The data set was constructed in four steps.

1. Parse Wikipedia with *MINIPAR* (Lin 1998a).
2. Select all object relations and count the frequency of each noun–verb pair $\langle n, v \rangle$.
3. Separate the noun–verb pairs into a training set (80%) and a test set (20%).
4. For each pair $\langle n, v \rangle$ in the test set find another verb $v'$ with the same frequency as $v$, such that $\langle n, v' \rangle$ appears neither in the training set nor the test set; replace $\langle n, v \rangle$ with the test triple $\langle n, v, v' \rangle$.

This creates two data sets. One is a training set of noun–verb pairs $\langle n, v \rangle$. The other is a test set made up of noun–verb–verb triples $\langle n, v, v' \rangle$. Examples of such triples are ⟨*task, assign, rock*⟩ and ⟨*data, extract, anticipate*⟩. We selected $v'$ so that its frequency is $v$'s frequency $\pm$ 1. We also ensured that the pair $\langle n, v' \rangle$ does not appear anywhere in the training or test data. To reduce noise and decrease the overall size of the dataset, we removed from both the test and training set all noun–verb object pairs which appeared less than five times. This produced a test set of 3327 triples and a training set of 464,303 pairs. We only used half of Wikipedia to generate this data set, the half not used in constructing the noun matrix.

We employed *edgeScore* (Equation 1) for all versions of *Roget's Thesaurus*. The methods implemented in the *WordNet::Similarity* software package (Pedersen *et al.* 2004) determine how close two words are in *WordNet*. These methods are J&C (Jiang and Conrath 1997), Res (Resnik 1995), Lin (Lin 1998a), W&P (Wu and Palmer 1994), L&C (Leacock and Chodorow 1998), H&SO (Hirst and St-Onge 1998), Path (counts edges between synsets), Lesk (Banerjee and Pedersen 2002), and finally Vector and Vector Pair (Patwardhan *et al.* 2003). The measure most similar to the *edgeScore* method is the Path measure in *WordNet*. J&C, Res, Lin, W&P, L&C and Path can only measure relatedness between nouns and verbs, because they only make use of hypernym links. H&SO uses all available *WordNet* relations in finding a path between two words. The Lesk and Vector methods use glosses and so might be just as easily implemented using a dictionary. They need not take advantage of *WordNet's* hierarchical structure.

To perform the PWSD task for each triple $\langle n, v, v' \rangle$, we found in the training corpus $k$ nouns which were the closest to $n$. Every such noun $m$ got a vote: the number of occurrences of the pair $\langle m, v \rangle$ minus the number of occurrences of $\langle m, v' \rangle$. Any value of $k$ could potentially be used. This means comparing each noun $n$ in the test data to every noun $m$ in the training set if these nouns share a common verb $v$ or $v'$. Such a computation is feasible in *Roget's Thesaurus*, but it takes a very long time for any *WordNet*-based measure.[17] To ensure that a fair value is

[17] We ran these experiments on an IBM ThinkCenter with a 3.4 GHz Intel Pentium 4 processor and 1.5GB 400 MHz DDR RAM.

Table 18: Pseudo-word-sense disambiguation error rates and run-time

| Method | Error Rate | $p$-value | Change | Time in seconds |
|---|---|---|---|---|
| 1911 | 0.257 | – | – | 58 |
| 1911X1 | 0.252 | 0.000 | +1.9% | 59 |
| 1911X5 | 0.246 | 0.000 | +4.3% | 60 |
| 1911R | 0.258 | 0.202 | −0.6% | 58 |
| 1987 | 0.252 | – | – | 135 |
| 1987X1 | 0.250 | 0.152 | +0.8% | 135 |
| 1987X5 | 0.246 | 0.010 | +2.3% | 134 |
| 1987R | 0.252 | 0.997 | 0.0% | 134 |
| J&C | 0.253 | – | – | 23 208 |
| Resnik | 0.258 | – | – | 23 112 |
| Lin | 0.251 | – | – | 19 840 |
| W&P | 0.245 | – | – | 38 721 |
| L&C | 0.241 | – | – | 23 445 |
| H&SO | 0.257 | – | – | 2 452 188 |
| Path | 0.241 | – | – | 22 720 |
| Lesk | 0.255 | – | – | 47 625 |
| Vector | 0.263 | – | – | 32 753 |
| Vct Pair | 0.272 | – | – | 74 803 |

selected, we divided the test set into 30 sets. We use 29 folds to find the optimal value of $k$ and apply it to the $30^{\text{th}}$ fold.

The score for the PWSD task is typically measured as an error rate where $T$ is the number of test cases (Equation 4).

$$\textit{Error rate} = \frac{1}{T}\left(\#\ \textit{incorrect choices} + \frac{\#\ \textit{ties}}{2}\right) \tag{4}$$

Table 18 shows the results of this experiment. The improvement on 1911X1 and 1911X5 over the original 1911 version of the *Thesaurus* was statistically significant at $p < 0.05$, according to Student's T-test. The improvement on the updated 1987 version was not statistically significant for 1987X1 with $p \approx 0.15$, but it was significant for 1987X5. The 1911X5 version gave results comparable to the 1987 version. The *Roget's*-based methods were comparable to the best *WordNet*-based methods.

When it comes to the values of $k$, $k = 1$ was always found to be the optimal value on this dataset. So, the best way to perform PWSD

is to select the *nearest* noun taken as the object of either $v$ or $v'$.

The CPU usage was perhaps the most pronounced difference with *Roget's*-based methods, which ran in a tiny fraction of the time which *WordNet*-based methods required. H&SO took around 28 days to run, so this measure simply is not an option for large-scale semantic relatedness problems. Even Lin, the fastest *WordNet*-based method, took around 5.5 hours, over 340 times longer than than the method based on the 1911 *Thesaurus*.

For all systems, a total of 193192 word pairs must be compared. We also examined the number of necessary comparisons between word senses. If one resource contains a larger number of senses of each word it is measuring distance on, then it will necessarily have to perform many more comparisons. The method based on the 1987 *Thesaurus* required nearly 120 million comparisons. The method based on the 1911 *Thesaurus* needed 14.7 million comparisons. For the *WordNet*-based methods only 3.5 million comparisons were necessary. Clearly the implementation of *Roget's Thesaurus* has a very strong advantage when it comes to runtime.

### 6.3 *SAT analogies*

The last class of problems to which we applied *Roget's Thesaurus* were analogy problems in the style of Scholastic Aptitude Tests (SAT). In an SAT analogy task, one is given a *target pair* $\langle A, B \rangle$ and then from a list of possible candidates selects the pair $\langle C, D \rangle$ most similar to the target pair. Ideally, the relation between the pair $\langle A, B \rangle$ should be the same as the relation between the pair $\langle C, D \rangle$. For example:

| Target pair | *word, language* |
|---|---|
| Candidates | *paint, portrait* |
| | *poetry, rhythm* |
| | ***note, music*** |
| | *tale, story* |
| | *week, year* |

*Roget's Thesaurus* performs well on problems of selecting synonyms and pseudo-word-sense disambiguation, but it is not clear just

Table 19: Scores in the analogy problem solved by matching kinds of relations (P = precision, R = recall, WN = *WordNet*)

| System | Correct | Ties | Incorrect | Filtered | P | R | F1 |
|---|---|---|---|---|---|---|---|
| 1911 | 14 | 21 | 39 | 300 | 0.307 | 0.061 | 0.102 |
| 1911X1 | 15 | 23 | 39 | 297 | 0.321 | 0.066 | 0.110 |
| 1911X5 | 15 | 27 | 39 | 293 | 0.330 | 0.072 | 0.118 |
| 1911R | 14 | 21 | 39 | 300 | 0.307 | 0.061 | 0.102 |
| 1987 | 18 | 85 | 81 | 190 | 0.271 | 0.133 | 0.179 |
| 1987X1 | 19 | 85 | 81 | 189 | 0.273 | 0.135 | 0.181 |
| 1987X5 | 21 | 85 | 81 | 187 | 0.278 | 0.139 | 0.185 |
| 1987R | 18 | 86 | 80 | 190 | 0.271 | 0.133 | 0.179 |
| WN 3.0 | 20 | 4 | 12 | 338 | 0.600 | 0.058 | 0.105 |

how well it will do on tasks of identifying analogies. That is because relations in the *Thesaurus* are unlabelled. We explore two methods of solving such problems with both the *Thesaurus* and *WordNet*. The first method attempts to identify a few kinds of relations in the *Thesaurus* and then apply them to identifying analogies. The second method uses edge distance between the pairs $\langle A, B\rangle$–$\langle C, D\rangle$ and $\langle A, C\rangle$–$\langle B, D\rangle$ as a heuristic for guessing whether two word pairs contain the same relation.

The dataset contains 374 analogy problems extracted from real SAT tests and practice tests (Turney 2005). A problem contains a *target pair* $\langle A, B\rangle$ and several pairs to choose from: $test_i = \langle X_i, Y_i\rangle$, $i = 1..5$. In evaluation, we consider seven scores: correct, ties, incorrect, filtered out, precision, recall and equal-weighted F-score. We define precision and recall in the same way as in Section 6.1. In the case of an $n$-way tie, the correct answer counts as $1/n$ towards the precision and recall. We consider recall as the most important measure, because it evaluates each method over the entire data set.

### 6.3.1 Matching relations

Unlike *WordNet*, *Roget's Thesaurus* contains no explicitly labelled semantic relations, but certain implicit relations can be inferred from its structure. Near-synonyms tend to appear in the same SG. Near-antonyms usually appear in different Heads in the same Head Group. One can also infer a hierarchical relation between two words if (1) they are in the same Paragraph and one of them is in the first SG, or (2) they are in the same POS and one of them is in the first SG of the

first Paragraph. So, three relations can be deduced from the *Thesaurus*. Two words can be near-synonymous, near-antonymous or hierarchically related. From *WordNet*, we allow words to be related by any of the explicit semantic relations. We also apply hypernymy/hyponymy transitively.

Using these semantic relations, the analogy problem is solved by identifying a candidate analogy which contains the same relation as the target pair. There will be no solution if no relation can be found for the target pair. This experiment is interesting in that it helps test whether narrower semantic relations in *WordNet* are more useful or less useful than the broader relations in *Roget's Thesaurus*. Table 19 shows the results; "Filtered" shows the number of pairs which were not scored because no relation could be established between the words in the target or candidate pairs.

The *WordNet*-based method has high precision, but recall is low compared to that of the *Roget's*-based versions. Interestingly, the precision and recall both increase as more words are added to the 1911 and 1987 versions of *Roget's*. We consider recall as more important in this evaluation, so it is clear that the most updated versions of *Roget's Thesaurus* outperform *WordNet* by a fair margin. Although the original 1911 version gave a lower F-score than *WordNet*, all other versions performed better. The existence of very specific semantic relations in *WordNet* did give it an edge in precision, but *WordNet* was only able to answer a few questions. This suggests that the relations between pairs in analogy tests are not only of the type encountered in *WordNet*. While the broader relations identified in the *Thesaurus* appear to be less reliable and give lower precision, the recall is much higher.

### 6.3.2 Edge distance

The second method of solving analogy problems uses edge distance between words as a heuristic. Analogy problems have been solved in this way using Equation 5 proposed by Turney (2006).

$$score(\langle A,\ B\rangle : \langle X_i,\ Y_i\rangle) = \frac{1}{2}\big(sim_a(A,\ X_i) + sim_a(B,\ Y_i)\big) \tag{5}$$

The highest-scoring pair $\langle X_i,\ Y_i\rangle$ is guessed as the correct analogy. This method assumes that $A$ and $X_i$ should be closely related and

so should $B$ and $Y_i$. An illustrative example is $\langle carpenter,\ wood\rangle$ and $\langle mason,\ stone\rangle$.

In Equation 5, $sim_a$ is the attributional similarity. We replaced it with an edge distance measure $r$, either *edgeScore* (Equation 1) or one of the measures built on *WordNet*. Because *edgeScore* only returns even numbers between 0 and 18, it tends to give many ties. We used a formula with a tie breaker based on the edge distance between $A$ and $B$ and between $X_i$ and $Y_i$:[18]

$$score(\langle A,B\rangle, \langle X_i, Y_i\rangle) = r(A,X_i) + r(B,Y_i) + \frac{1}{|r(A,B) - r(X_i,Y_i)| + 1} \qquad (6)$$

The last term of the sum in Equation 6 acts as a tie-breaker which favours candidates $\langle X_i,\ Y_i\rangle$ with an edge distance similar to the target $\langle A,\ B\rangle$. We include another constraint: $A$ and $X_i$ must be in the same part of speech, and so do $B$ and $Y_i$. Only one sense of each of $A$, $B$, $X_i$ and $Y_i$ can be used in the calculation of Equation 6. For example, the same sense of $A$ is used when calculating $r(A,\ X_i)$ and $r(A,\ B)$.

We applied Equation 6 to the 374 analogy problems using all versions of *Roget's Thesaurus* and the *WordNet*-based edge distance measures. The results are shown in Table 20. The "Filtered" column shows how many SAT problems could not be solved because at least one of the words needed was absent in either the *Thesaurus* or *WordNet*. Unfortunately, expanding *Roget's Thesaurus* did not reduce the number of filtered results. That said, both precision and recall increased when more words were added to the *Thesaurus*. Overall, we found that in absolute numbers the updated 1987X5 *Roget's Thesaurus* performed better than any other resource examined. Even the updated versions of the 1911 *Thesaurus* performed on par with the best *WordNet*-based systems. We must note, however, that none of the improvements of the 1987X5-based method over any given *WordNet* method are statistically significant.

[18] We owe this formula to a personal communication with Dr. Vivi Nastase. It was also used in (Kennedy and Szpakowicz 2007)

| System | Correct | Ties | Misses | Filtered | P | R | F1 |
|---|---|---|---|---|---|---|---|
| 1911 | 98 | 11 | 214 | 51 | 0.319 | 0.276 | 0.296 |
| 1911X1 | 98 | 17 | 208 | 51 | 0.329 | 0.284 | 0.305 |
| 1911X5 | 97 | 20 | 206 | 51 | 0.330 | 0.285 | 0.306 |
| 1911R | 97 | 12 | 218 | 47 | 0.313 | 0.274 | 0.292 |
| 1987 | 101 | 35 | 232 | 6 | 0.318 | 0.313 | 0.316 |
| 1987X1 | 102 | 38 | 228 | 6 | 0.324 | 0.319 | 0.322 |
| 1987X5 | 102 | 39 | 227 | 6 | 0.325 | 0.320 | 0.323 |
| 1987R | 103 | 34 | 233 | 4 | 0.320 | 0.316 | 0.318 |
| Path | 85 | 5 | 166 | 118 | 0.342 | 0.234 | 0.278 |
| J&C | 80 | 0 | 176 | 118 | 0.312 | 0.214 | 0.254 |
| Resnik | 91 | 16 | 149 | 118 | 0.385 | 0.263 | 0.313 |
| Lin | 82 | 3 | 171 | 118 | 0.325 | 0.222 | 0.264 |
| W&P | 90 | 1 | 165 | 118 | 0.354 | 0.242 | 0.287 |
| L&C | 91 | 4 | 161 | 118 | 0.363 | 0.249 | 0.295 |
| H&SO | 96 | 39 | 212 | 27 | 0.321 | 0.298 | 0.309 |
| Lesk | 113 | 0 | 234 | 27 | 0.326 | 0.302 | 0.313 |
| Vector | 113 | 0 | 234 | 27 | 0.326 | 0.302 | 0.313 |
| Vector Pair | 106 | 0 | 241 | 27 | 0.305 | 0.283 | 0.294 |

Table 20: Scores in the analogy problem solved using semantic distance functions (P = precision, R = recall)

# 7 CONCLUSION

## 7.1 *Summary*

We have described a method of automatically updating *Roget's Thesaurus* with new words. The process has two main steps: lists of semantically related words are generated, and next those lists are used to find a place for a new word in the *Thesaurus*. We have enhanced both steps by leveraging the structure of *Roget's Thesaurus*.

When creating lists of related words, we have evaluated a technique for measuring semantic relatedness which enhances distributional methods using lists of known synonyms. We have shown this to have a statistically significant effect on the quality of measures of semantic relatedness.

In the second step, the actual addition of new words to *Roget's Thesaurus*, we generated a list of neighbouring words and used them as anchors to identify where in the *Thesaurus* to place a new word. This process benefits from tuning on the actual *Thesaurus*. The task here is to find words which will be good anchors for determining where to

place a new term. We experimented with three methods of finding anchors, using the rank, the relatedness score and a relative relatedness score. We found that rank worked best. The process of adding new words to *Roget's Thesaurus* is hierarchical. First the Part of Speech in the *Thesaurus* is identified, then the Paragraph, then the Semicolon Group. A new Paragraph or Semicolon Group can be created if needed.

A manual evaluation of our methodology found that added words were almost indistinguishable from words already present in the *Thesaurus*. Even after multiple passes the words seemed to find fairly accurate placing in an existing Paragraph. When adding words to a new Paragraph, after one pass the words were highly accurate, but the accuracy fell after additional passes. In total, some 5500 words were added to the 1911 version and some 9600 words to the 1987 version.

We also performed an application-based evaluation to compare the original and updated versions of *Roget's Thesaurus* and, when possible, *WordNet*. The tasks were synonym identification, pseudo-word-sense disambiguation and SAT-style analogy problems. On all tasks the updates to the *Thesaurus* showed a noticeable improvement. In our evaluations, *Roget's Thesaurus* also performed as well as, or better than, *WordNet*. In particular, it could perform calculations many times faster than the *WordNet::Similarity* software package (Pedersen *et al.* 2004).

Most of our experiments show that the 1987 version of *Roget's Thesaurus* outperforms the 1911 version. There are two reasons. First, for our measure of semantic relatedness, the evaluation was conducted on words in the same *Roget's grouping* from the 1987 version. Since the structure of the *Thesaurus* is used to train our MSR, it is natural that scores are higher when training and evaluation are done on the same version. The second reason is simply that the 1987 version is larger. When adding new words to *Roget's Thesaurus*, a larger thesaurus gives more potential anchor words to help find an appropriate placement for a new word. For our task-based evaluation, the applications we chose will naturally benefit from a larger thesaurus as well.

### 7.2 *Future work*

The supervised measure of semantic relatedness provides an interesting method of re-weighting contexts. Recent work has shown that similar techniques make it possible to find a weighted mapping be-

tween the context space in two different languages (Kennedy and Hirst 2012). Methods of this kind could be used to emphasise similarities between words based on sentiment, emotion or formality, rather than simply on synonymy. Using emotionally related words as a source of training data could enable the creation of a measure of semantic relatedness which favours words of the same emotional class over other, nearer, synonyms conveying a different emotion.

Perhaps other more complex methods of adding new words to *Roget's Thesaurus* can be considered. For example, mixing rank and score (maybe using machine learning) might lead to an even more accurate method. Other methods of identifying where in the *Thesaurus* to place a word could also be considered. In particular, Pantel's (2005) method could potentially be modified to work for *Roget's Thesaurus*.

Our method only adds individual words to *Roget's Thesaurus*. It should be possible to expand it into adding multi-word phrases. Many dependency parsers can identify noun phrases and so can be used to create distributional vectors for such phrases. Adding multi-word phrases to verb or adjective *Roget's groupings* may be possible by identifying n-grams which are frequent in a text. Two problems arise. One is determining whether high frequency alone is a good enough reason to add a multi-word phrase. The second is how to represent such multi-word phrases. It could be possible to represent them by vectors of word–relation pairs for syntactically related words in the same sentence, but outside of the phrase being considered. The meaning of a phrase may also be deduced by composing the distributional vectors of its individual words. There is ongoing, and very interesting, research in this area (Mitchell and Lapata 2008; Turney 2012).

A problem which we have not tackled yet is that of adding cross-references: if the same word appears in two places in *Roget's Thesaurus*, then often a cross-reference links the two occurrences. Making use of these cross-references could be a considerable undertaking, because it requires, amongst other things, some form of effective word-sense disambiguation.

The manual annotation has only been conducted on the 1911 version of *Roget's Thesaurus*, because it is the only version which can be released to the public, and because the annotation experiment has been very time-consuming. In the interest of completeness, the updates to the 1987 version could be evaluated similarly. We expect that those

updates should actually be more accurate, because the 1911 version is both older and smaller. This would be in line with the automatic evaluation from Section 4, but it is yet to be confirmed manually.

It should be possible to adapt our methods of placing words in *Roget's Thesaurus* to work for *WordNet*. Instead of identifying words in the same POS, then Paragraph, then SG, word groupings could be created from *WordNet*'s hypernym hierarchy. We envisage two ways of doing this. The first would be to pick a relatively high level within the hierarchy and classify each word into one or more of the synsets at that level, much as we did with the POS level. A synset could be represented by all the words in the transitive closure of its hyponym synsets. Next, the word would be propagated down the hierarchy – as we do with Paragraphs and SGs – until it can go no further, and then added to the synset there.

This method could not (yet) be applied to adjectives, and would only take one kind of relation into account when placing a word in *WordNet*. Another option is to create a neighbourhood of words for each synset, based on a variety of relations. A word could then be placed in a larger grouping of multiple synsets before the particular synset it belongs to is determined. If no synset can be picked, then a new synset can be created with some sort of ambiguous link joining it to the other synsets in its neighbourhood. A hybrid of these two methods is also possible. Our first method could be enhanced by using not only a synset's terms, but also its close neighbours. This would expand the set of anchor words at the cost of introducing words common to multiple synsets.

It should also be possible to port our method to thesauri and wordnets in other languages. The main problem might be our method's reliance on a dependency parser. Such parsers are not available yet for many languages. Nonetheless, it could be possible to replicate much of the relevant functionality of a dependency parser using a part-of-speech tagger – and taggers are quite widely available. For example, one may assume that a noun can only be modified by other nouns or adjectives in its vicinity, and so only use those terms in constructing a distributional vector.

Another direction which this kind of research could take would be to test the methods on adding words in a particular domain. Most of the words in *Roget's Thesaurus* are from everyday English, as op-

posed to, say, medical terms. The nearest synonyms of such technical words will be other technical words. This could make it more difficult to actually add domain-specific terms to *Roget's Thesaurus*. That said, the trainable measure of semantic relatedness from Kennedy and Szpakowicz (2011, 2012) could be built using words of a particular domain. If domain-specific and everyday words could be grouped as near-synonyms, then an MSR could be trained for adding domain-specific terms to *Roget's Thesaurus*.

Similar to adding domain-specific words is the challenge of adding brand new coinage to *Roget's Thesaurus*. Very new words may not have close synonyms in the *Thesaurus*, which is why we add words in multiple passes. It would be interesting to investigate how many passes are required before, say, the word "iPhone" is added to the *Thesaurus*. Closely related phrases like "mobile phone" or "smart phone" would need to already be present. Other terms, such as "cellular network", "texting" or "Apple", could also be useful in choosing where to place a word like "iPhone".

Finally, note that we have only applied *Roget's Thesaurus* to three NLP tasks, to demonstrate value in both its structure and language coverage. Many other applications of the *Thesaurus* are possible. Some obvious ones include real word-sense disambiguation and lexical substitution. *Roget's Thesaurus* has already been used in the construction of lexical chains (Morris and Hirst 1991; Jarmasz and Szpakowicz 2003a). Lexical chains might be applied to summarisation or text segmentation. Since the *Thesaurus* contains a large number of opposing concepts, it may be possible to apply it to lexical entailment as well.

NLP researchers are always on the hunt for newer and larger data sets on which to train and evaluate their experiments. Many of these experiments will require measuring semantic distance among huge sets of words. In the coming years, the trend towards analyzing big data will drive the need for fast semantic relatedness calculation. *Roget's Thesaurus* is uniquely suited for that.

## REFERENCES

Satanjeev BANERJEE and Ted PEDERSEN (2002), An Adapted Lesk Algorithm for Word Sense Disambiguation Using WordNet, in *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics – CICLing 2002*, pp. 136–145, Mexico City, Mexico.

Patrick J. CASSIDY (2000), An Investigation of the Semantic Relations in the Roget's Thesaurus: Preliminary Results, in *Proceedings of the Conference on Intelligent Text Processing and Computational Linguistics – CICLing 2000*, pp. 181–204, Mexico City, Mexico.

Stephen CLARK and David WEIR (2002), Class-based Probability Estimation Using a Semantic Hierarchy, *Computational Linguistics*, 28(2):187–206.

Carolyn J. CROUCH (1988), A Cluster-based Approach to Thesaurus Construction, in *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval – SIGIR 1988*, pp. 309–320, Grenoble, France.

Carolyn J. CROUCH and Bokyung YANG (1992), Experiments in Automatic Statistical Thesaurus Construction, in *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval – SIGIR 1992*, pp. 77–88, Copenhagen, Denmark.

DAGAN, IDO, Lillian LEE, and Fernando PEREIRA (1999), Similarity-based Models of Word Co-occurrence Probabilities, *Machine Learning*, 34(1-3):43–69.

Andrea ESULI and Fabrizio SEBASTIANI (2006), SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining, in *Proceedings of the 5th Conference on Language Resources and Evaluation – LREC 2006*, pp. 417–422, Genoa, Italy.

Christiane FELLBAUM, editor (1998), *WordNet: an Electronic Lexical Database*, MIT Press, Cambridge, MA, USA.

William A. GALE, Kenneth W. CHURCH, and David YAROWSKY (1992), A Method for Disambiguating Word Senses in a Large Corpus, *Computers and the Humanities*, 26:415–439.

Roxana GIRJU, Adriana BADULESCU, and Dan MOLDOVAN (2003), Learning Semantic Constraints for the Automatic Discovery of Part–Whole Relations, in *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics – HLT-NAACL 2003*, pp. 1–8, Edmonton, Canada.

Roxana GIRJU, Adriana BADULESCU, and Dan MOLDOVAN (2006), Automatic Discovery of Part–Whole Relations, *Computational Linguistics*, 32(1):83–136.

Marti A. HEARST (1992), Automatic Acquisition of Hyponyms from Large Text Corpora, in *Proceedings of the 14th International Conference on Computational Linguistics – COLING 1992*, pp. 539–545, Nantes, France.

Graeme HIRST and David ST-ONGE (1998), Lexical Chains as Representation of Context for the Detection and Correction of Malapropisms, in Christiane FELLBAUM, editor, *WordNet: An Electronic Lexical Database*, pp. 305–322, MIT Press, Cambridge, MA, USA.

Mario JARMASZ (2003), *Roget's Thesaurus as a Lexical Resource for Natural Language Processing*, Master's thesis, University of Ottawa, Canada.

Mario JARMASZ and Stan SZPAKOWICZ (2003a), Not as Easy As It Seems: Automating the Construction of Lexical Chains Using Roget's Thesaurus, in *16th Conference of the Canadian Society for Computational Studies of Intelligence – AI 2003, Halifax, Canada*, number 2671 in Lecture Notes in Computer Science, pp. 544–549, Springer, Berlin/Heidelberg, Germany.

Mario JARMASZ and Stan SZPAKOWICZ (2003b), Roget's Thesaurus and Semantic Similarity, in *Proceedings of the Conference on Recent Advances in Natural Language Processing – RANLP 2003*, pp. 212–219, Borovets, Bulgaria.

Mario JARMASZ and Stan SZPAKOWICZ (2004), Roget's Thesaurus and Semantic Similarity, in N. NICOLOV, K. BONTCHEVA, G. ANGELOVA, and R. MITKOV, editors, *Recent Advances in Natural Language Processing III: Selected Papers from RANLP 2003*, volume 260 of *Current Issues in Linguistic Theory*, pp. 111–120, John Benjamins, Amsterdam, The Netherlands/Philadelphia, PA, USA.

Jay J. JIANG and David W. CONRATH (1997), Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy, in *Proceedings of the 10th International Conference on Research on Computational Linguistics – ROCLING X*, pp. 19–33, Taipei, Taiwan.

Joshua C. KENDALL (2008), *The Man Who Made Lists : Love, Death, Madness, and the Creation of Roget's Thesaurus*, G.P.Putnam's Sons, New York, NY, USA.

Alistair KENNEDY and Graeme HIRST (2012), Measuring Semantic Relatedness Across Languages, in *xLiTe: Cross-Lingual Technologies, workshop collocated with the Conference on Neural Information Processing Systems – NIPS 2012*, Lake Tahoe, NV, USA.

Alistair KENNEDY and Stan SZPAKOWICZ (2007), Disambiguating Hypernym Relations for Roget's Thesaurus, in *Proceedings of the 10th International Conference on Text, Speech and Dialogue – TSD 2007, Pilsen, Czech Republic*, number 4629 in Lecture Notes in Artificial Intelligence, pp. 66–75, Springer, Berlin/Heidelberg, Germany.

Alistair KENNEDY and Stan SZPAKOWICZ (2011), A Supervised Method of Feature Weighting for Measuring Semantic Relatedness, in *Proceedings of the Canadian Conference on Artificial Intelligence – AI 2011, St. John's, Canada*, number 6657 in Lecture Notes in Artificial Intelligence, pp. 222–233, Springer, Berlin/Heidelberg, Germany.

Alistair KENNEDY and Stan SZPAKOWICZ (2012), Supervised Distributional Semantic Relatedness, in *Proceedings of the 15th International Conference on Text, Speech and Dialogue – TSD 2012, Brno, Czech Republic*, number 7499 in Lecture Notes in Artificial Intelligence, pp. 207–214, Springer, Berlin/Heidelberg, Germany.

Betty KIRKPATRICK, editor (1987), *Roget's Thesaurus of English Words and Phrases*, Longman, Harlow, UK.

Klaus KRIPPENDORFF (2004), *Content Analysis: An Introduction to Its Methodology*, Sage Publications Inc., Los Angeles, CA, USA, 2nd edition.

Oi Yee KWONG (1998a), Aligning WordNet with Additional Lexical Resources, in *Proceedings of the COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, pp. 73–79, Montréal, Canada.

Oi Yee KWONG (1998b), Bridging the Gap Between Dictionary and Thesaurus, in *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics – ACL 1998*, pp. 1487–1489, Montréal, Canada.

Robin J. LANDIS and G. G. KOCH (1977), The Measurement of Observer Agreement for Categorical Data, *Biometrics*, 33:159–174.

Claudia LEACOCK and Martin CHODOROW (1998), Combining Local Context and WordNet Sense Similarity for Word Sense Disambiguation, in Christiane FELLBAUM, editor, *WordNet: An Electronic Lexical Database*, pp. 265–284, MIT Press, Cambridge, MA, USA.

Lillian LEE (1999), Measures of Distributional Similarity, in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics – ACL 1999*, pp. 25–32, College Park, MD, USA.

Lothar LEMNITZER, Holger WUNSCH, and Piklu GUPTA (2008), Enriching GermaNet with Verb–Noun Relations – a Case Study of Lexical Acquisition, in *Proceedings of the 6th International Conference on Language Resources and Evaluation – LREC 2008*, Marrakech, Morocco.

Dekang LIN (1998a), Automatic Retrieval and Clustering of Similar Words, in *Proceedings of the 17th International Conference on Computational Linguistics – COLING 1998*, pp. 768–774, Montréal, Canada.

Dekang LIN (1998b), Dependency-Based Evaluation of MINIPAR, in *Proceedings of the Workshop on the Evaluation of Parsing Systems at the 1st International Conference on Language Resources and Evaluation – LREC 1998*, Granada, Spain.

Bernardo MAGNINI and Gabriela CAVAGLIÁ (2000), Integrating Subject Field Codes into WordNet, in *Proceedings of the 2nd International Conference on Language Resources and Evaluation – LREC 2000*, pp. 1413–1418, Athens, Greece.

Jeff MITCHELL and Mirella LAPATA (2008), Vector-based models of semantic composition, in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies – ACL 2008: HLT*, pp. 236–244, Columbus, OH, USA.

Emmanuel MORIN and Christian JACQUEMIN (1999), Projecting Corpus-Based Semantic Links on a Thesaurus, in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics – ACL 1999*, pp. 389–396, College Park, MD, USA.

Jane MORRIS and Graeme HIRST (1991), Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text, *Computational Linguistics*, 17(1):21–48.

Vivi NASTASE and Stan SZPAKOWICZ (2001), Word Sense Disambiguation in Roget's Thesaurus Using WordNet, in *Proceedings of the NAACL Workshop on WordNet and Other Lexical Resources*, pp. 12–22, Pittsburgh, PA, USA.

Patrick PANTEL (2003), *Clustering by Committee*, Ph.D. thesis, University of Alberta, Canada.

Patrick PANTEL (2005), Inducing Ontological Co-occurrence Vectors, in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics – ACL 2005*, pp. 125–132, Ann Arbor, MI, USA.

Siddharth PATWARDHAN, Satanjeev BANERJEE, and Ted PEDERSEN (2003), Using Measures of Semantic Relatedness for Word Sense Disambiguation, in *Proceedings of the 4th International Conference on Intelligent Text Processing and Computational Linguistics – CICLing-2003*, pp. 241–257, Mexico City, Mexico.

Ted PEDERSEN, Siddhart PATWARDHAN, and Jason MICHELIZZI (2004), Wordnet::Similarity - Measuring the Relatedness of Concepts, in *Proceedings of the 19th National Conference on Artificial Intelligence – AAAI 2004*, pp. 1024–1025, San Jose, CA, USA.

Maciej PIASECKI, Bartosz BRODA, Michał MARCIŃCZUK, and Stan SZPAKOWICZ (2009), The WordNet Weaver: Multi-criteria Voting for Semi-automatic Extension of a Wordnet, in *Proceedings of the 22nd Canadian Conference on Artificial Intelligence – AI 2009, Kelowna, Canada*, number 5549 in Lecture Notes in Artificial Intelligence, pp. 237–240, Springer, Berlin/Heidelberg, Germany.

Paul PROCTER (1978), *Longman Dictionary of Contemporary English*, Longman Group Ltd., Essex, UK.

Philip RESNIK (1995), Using Information Content to Evaluate Semantic Similarity, in *Proceedings of the 14th International Joint Conference on Artificial Intelligence – IJCAI 1995*, pp. 448–453, Montréal, Canada.

Mats ROOTH, Stefan RIEZLER, Detlef PRESCHER, Glenn CARROLL, and Franz BEIL (1999), Inducing a Semantically Annotated Lexicon via EM-based Clustering, in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics – ACL 1999*, pp. 104–111, College Park, MD, USA.

Sara RYDIN (2002), Building a Hyponymy Lexicon with Hierarchical Structure, in *Proceedings of the ACL-02/SIGLEX Workshop on Unsupervised Lexical Acquisition – ULA 2002*, pp. 26–33, Philadelphia, PA, USA.

Benoît SAGOT and Darja FIŠER (2011), Extending wordnets by learning from multiple resources, in *Proceedings of the 5th Language and Technology Conference – LTC 2011*, pp. 526–530, Poznań, Poland.

Erik Tjong Kim SANG (2007), Extracting Hypernym Pairs from the Web, in *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics – ACL 2007 (Interactive Poster and Demonstration Sessions)*, pp. 165–168, Prague, Czech Republic.

Hinrich SCHÜTZE (1998), Automatic Word Sense Discrimination, *Computational Linguistics*, 24(1):97–123.

Keiji SHINZATO and Kentaro TORISAWA (2004), Acquiring Hyponymy Relations from Web Documents, in *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics – HLT-NAACL 2004*, pp. 73–80, Boston, MA, USA.

Rion SNOW, Daniel JURAFSKY, and Andrew Y. NG (2005), Learning Syntactic Patterns for Automatic Hypernym Discovery, in Lawrence K. SAUL, Yair WEISS, and Léon BOTTOU, editors, *Advances in Neural Information Processing Systems 17*, pp. 1297–1304, MIT Press, Cambridge, MA, USA.

Rion SNOW, Daniel JURAFSKY, and Andrew Y. NG (2006), Semantic Taxonomy Induction from Heterogenous Evidence, in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics – COLING/ACL 2006*, Sydney, Australia.

Ratanachai SOMBATSRISOMBOON, Yutaka MATSUO, and Mitsuru ISHIZUKA (2003), Acquisition of Hypernyms and Hyponyms from the WWW, in *Procedings of the 2nd International Workshop on Active Mining – AM 2003 (in Conjunction with the International Symposium on Methodologies for Intelligent Systems)*, pp. 7–13, Maebashi City, Japan.

Carlo STRAPPARAVA and Alessandro VALITUTTI (2004), WordNet-Affect: an Affective Extension of WordNet, in *Proceedings of the 4th International Conference on Language Resources and Evaluation – LREC 2004*, pp. 1083–1086, Lisbon, Portugal.

Hiroaki TSURUMARU, Toru HITAKA, and Sho YOSHIDA (1986), An Attempt to Automatic Thesaurus Construction from an Ordinary Japanese Language Dictionary, in *Proceedings of the 11th Conference on Computational Linguistics – COLING 1986*, pp. 445–447, Bonn, Germany.

Peter TURNEY (2005), Measuring Semantic Similarity by Latent Relational Analysis, in *Proceedings of the 19th International Joint Conference on Artificial Intelligence – IJCAI-05*, pp. 1136–1141, Edinburgh, Scotland.

Peter TURNEY (2006), Similarity of Semantic Relations, *Computational Linguistics*, 32(3):379–416.

Peter TURNEY (2012), Domain and Function: A Dual-Space Model of Semantic Relations and Compositions, *Journal of Artificial Intelligence Research*, 44:533–585.

Piek VOSSEN, editor (1998), *EuroWordNet: a Multilingual Database with Lexical Semantic Networks*, Kluwer Academic Publishers, Norwell, MA, USA.

Julie WEEDS and David WEIR (2005), Co-occurrence Retrieval: A Flexible Framework for Lexical Distributional Similarity, *Computational Linguistics*, 31(4):439–475.

Zhibiao WU and Martha PALMER (1994), Verb Semantics and Lexical Selection, in *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics – ACL 1994*, pp. 133–138, Las Cruces, NM, USA.

Hao ZHENG, Xian WU, and Yong YU (2008), Enriching WordNet with Folksonomies, in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining – PAKDD 2008*, number 5012 in Lecture Notes in Artificial Intelligence, pp. 1075–1080, Springer, Berlin/Heidelberg, Germany.

Maayan ZHITOMIRSKY-GEFFET and Ido DAGAN (2009), Bootstrapping distributional feature vector quality, *Computational Linguistics*, 35(3):435–461.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*
http://creativecommons.org/licenses/by/3.0/

# Bimorphisms and synchronous grammars

*Stuart M. Shieber*
School of Engineering and Applied Sciences
Harvard University, Cambridge MA, USA

## ABSTRACT

We tend to think of the study of language as proceeding by characterizing the strings and structures of a language, and we think of natural-language processing as using those structures to build systems of utility in manipulating the language. But many language-related problems are more fruitfully viewed as requiring the specification of a relation between two languages, rather than the specification of a single language.

In this paper, we provide a synthesis and extension of work that unifies two approaches to such language relations: the automata-theoretic approach based on tree transducers that transform trees to their counterparts in the relation, and the grammatical approach based on synchronous grammars that derive pairs of trees in the relation. In particular, we characterize synchronous tree-substitution grammars and synchronous tree-adjoining grammars in terms of bimorphisms, which have previously been used to characterize tree transducers. In the process, we provide new approaches to formalizing the various concepts: a metanotation for describing varieties of tree automata and transducers in equational terms; a rigorous formalization of tree-adjoining and tree-substitution grammars and their synchronous counterparts, using trees over ranked alphabets; and generalizations of tree-adjoining grammar allowing multiple adjunction.

*Keywords:*
*synchronous grammars,*
*tree transducers,*
*tree-adjoining grammars,*
*tree-substitution grammars*

## 1 INTRODUCTION

We tend to think of the study of language as proceeding by characterizing the strings and structures of a language, and we think of natural-language processing as using those structures to build systems of utility in manipulating the language. But many language-related problems are more fruitfully viewed as requiring the specification of a *relation* between two languages, rather than the specification of a single language. The paradigmatic case is machine translation, where the translation relation between the source and target natural languages is itself the goal to be characterized. Similarly, the study of semantics involves a relation between a natural language and a language of semantic representation (phonological form and logical form in one parlance). Computational interpretation of text, as in question-answering or natural-language command and control systems, requires computing that relation in the direction from natural language to semantic representation, and tactical generation in the opposite direction. Sentence paraphrase and compression can be thought of as computing a relation between strings of a single natural language. Similar examples abound.

The modelling of these relations has been a repeated area of study throughout the history of computational linguistics, proceeding in phases that have alternated between emphasizing automata-theoretic tools and grammatical tools. On the automata-theoretic side, the early pioneering work of Rounds (1970) on *tree transducers* was intended to formalize aspects of transformational grammars, and led to a long development of the formal-language theory of tree transducers. Grammatical approaches are based on the idea of synchronizing the grammars of the related languages. We use the general term *synchronous grammars* for the idea (Shieber and Schabes 1990), though early work in formalizing programming-language compilation uses the more domain-specific term *syntax-directed transduction* or *translation* (Lewis and Stearns 1968; Aho and Ullman 1969), and a variety of specific systems – inversion transduction grammars (Wu 1996, 1997), head transducers (Alshawi *et al.* 2000), multitext grammars (Melamed 2003, 2004) – forgo the use of the term. The early work on the synchronous grammar approach for natural-language application involved synchronizing tree-adjoining grammars (TAG). A recent

resurgence of interest in automata-theoretic approaches in the machine translation community (Graehl and Knight 2004; Galley *et al.* 2004) has led to more powerful types of transducers (Maletti *et al.* 2009) and a far better understanding of the computational properties of and relationships among different transducer types (Maletti *et al.* 2009). Synchronous grammars have also seen a rise in application in areas such as machine translation (Nesson *et al.* 2006; DeNeefe and Knight 2009), linguistic semantics (Nesson and Shieber 2006; Han and Hedberg 2008), and sentence compression (Yamangil and Shieber 2010).

As these various models were developed, the exact relationship among them had been unclear, with a large number of seemingly unrelated formalisms being independently proposed or characterized. In particular, the grammatical approach to tree relations found in synchronous grammar formalisms and the automata-theoretic approach of tree transducers have been viewed as contrasting approaches.

A reconciliation of these two approaches was initiated in two pieces of earlier work (Shieber 2004, 2006), which the present paper unifies, simplifies, and extends. That work proposed to use the formal-language-theoretic device of bimorphisms (Arnold and Dauchet 1982), previously little known outside the formal-language-theory community, as a means for unifying the two approaches and clarifying the interrelations. It investigated the formal properties of synchronous tree-substitution grammars (STSG) and synchronous tree-adjoining grammars (STAG) from this perspective, showing that both formalisms, along with traditional tree transducers, can be thought of as varieties of bimorphisms. This earlier work has already been the basis for further extensions, such as the synchronous context-free tree grammars of Nederhof and Vogler (2012).

The present paper includes all of the results of the prior two papers, with notations made consistent, presentations clarified and expanded, and proofs simplified, and therefore supersedes those papers. It provides a definitive presentation of the formal foundations for TSG, TAG, and their synchronous versions, improving on the earlier presentations. To our knowledge, it provides the most consistent definition of TAG and STAG available, and the only one to use trees over ranked rather than unranked alphabets. It also, in passing, provides a characterization of transducers in terms of equational systems using

a uniform metagrammar notation, a new characterization of the relation between tree-adjoining grammar derivation and derived trees, and a new simpler and more direct proof of the equivalence of tree-adjoining languages and the output languages of monadic macro tree transducers, formal contributions that may have independent utility. Finally, it extends the prior results to cover more linguistically appropriate variants of synchronous tree-adjoining grammars, in particular incorporating multiple adjunction.

After some preliminaries (Section 2), we present a set of known results relating context-free languages, tree homomorphisms, tree automata, and tree transducers to extend them for the tree-adjoining languages (Section 3), presenting these in terms of restricted kinds of functional programs over trees, using a simple grammatical notation for describing the programs. We review the definition of tree-substitution and tree-adjoining grammars (Section 4) and synchronous versions thereof (Section 5). We prove the equivalence between STSG and a variety of bimorphism (Section 6).

The grammatical presentation of transducers as functional programs allows us to easily express generalizations of the notions: monadic macro tree homomorphisms, automata, and transducers, which bear (at least some of) the same interrelationships that their traditional simpler counterparts do (Section 7). Finally, we use this characterization to place the synchronous TAG formalism in the bimorphism framework (Section 7.3), further unifying tree transducers and other synchronous grammar formalisms. We show that these methods generalize to TAG allowing multiple adjunction as well (Section 8).[1]

The present work, being based on and synthesizing work from some ten years ago, is by no means the last word in the general area. Indeed, since publication of the earlier articles, the connections among synchronous grammars, transducers, and bimorphisms have been considerably further clarified. The relation between bimorphisms and tree transducers has benefitted from a notion of extended top-down tree transducers, which have been shown to be strongly equivalent to the $B(LC, LC)$ bimorphism class we discuss below (Maletti 2008). Koller

[1] Much of the content in Sections 2–7 of this paper is based on material in previous papers (Shieber 2004, 2006), and is used by permission.

and Kuhlmann (2011) provide an elegant generalization of monolingual and synchronous systems in terms of interpreted regular tree grammars (IRTG), in spirit quite close to the idea here of reconstructing synchronous grammars as bimorphism-like formal systems. Their IRTG can be used for CFG, TSG, TAG, and synchronous versions of various sorts. Of especial interest are the formalizations of Büchse *et al.* (2012, 2014), which modify the definitions of TAG to incorporate state information at substitution and adjunction sites. This modification eliminates much of the inelegance of the formalization here that accounts for our having to couch the various equivalences we show in terms of weak rather than strong generative capacity. The presentation below should be helpful in understanding the background to these works as well.

## 2 PRELIMINARIES

We start by defining the terminology and notations that we will use for strings, trees, and the like.

### 2.1 *Basics*

We will notate sequences with angle brackets, e.g., $\langle a, b, c\rangle$, or where no confusion results, simply as $abc$, with the empty string written $\epsilon$.

We follow much of the formal-language-theory literature (and in particular, the tree transducer literature) in defining trees over *ranked* alphabets, in which the symbols decorating the nodes are associated with fixed arities. (By contrast, formal work in computational linguistics typically uses unranked trees.) Trees will thus have nodes labeled with elements of a RANKED ALPHABET, a set of symbols $\mathcal{F}$, each with a non-negative integer RANK or ARITY assigned to it, determining the number of children for nodes so labeled. To emphasize the arity of a symbol, we will write it as a parenthesized superscript, for instance $f^{(n)}$ for a symbol $f$ of arity $n$. Analogously, we write $\mathcal{F}^{(n)}$ for the set of symbols in $\mathcal{F}$ with arity $n$. Symbols with arity zero ($\mathcal{F}^{(0)}$) are called NULLARY symbols or CONSTANTS. The set of nonconstants is written $\mathcal{F}^{(\geq 1)}$.

To express incomplete trees, trees with "holes" waiting to be filled, we will allow leaves to be labeled with variables, in addition to nullary symbols. The set of TREES OVER A RANKED ALPHABET $\mathcal{F}$

AND VARIABLES $\mathcal{X}$, notated $\mathcal{T}(\mathcal{F}, \mathcal{X})$, is the smallest set such that

**Nullary symbols at leaves** $f \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ for all $f \in \mathcal{F}^{(0)}$;

**Variables at leaves** $x \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ for all $x \in \mathcal{X}$;

**Internal nodes** $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ for all $f \in \mathcal{F}^{(n)}$, $n \geq 1$, and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

Where convenient, we will blur the distinction between the leaf and internal node notation for a nullary symbol $f$, allowing $f()$ as synonymous for the leaf node $f$.

We abbreviate $\mathcal{T}(\mathcal{F}, \emptyset)$, where the set of variables is empty, as $\mathcal{T}(\mathcal{F})$, the set of GROUND TREES over $\mathcal{F}$. We will also make use of the set of $n$ numerically ordered variables $\mathcal{X}_n = \{x_1, \ldots, x_n\}$, and write $x$, $y$, $z$ as synonyms for $x_1$, $x_2$, $x_3$, respectively.

Trees can also be viewed as mappings from TREE ADDRESSES, sequences of integers, to the labels of nodes at those addresses. The address $\epsilon$ is the address of the root, 1 the address of the first child, 12 the address of the second child of the first child, and so forth. We write $q \prec p$ to indicate that tree address $q$ is a proper prefix of $p$, and $p - q$ for the sequence obtained from $p$ by removing prefix $q$ from the front. For instance, $1213 - 12 = 13$.

We will use the notation $t/p$ to pick out the subtree of the node at address $p$ in the tree $t$, that is, (using $\cdot$ for the insertion of an element on a sequence)

$$\begin{aligned} t/\epsilon &= t \\ f(t_1, \ldots, t_n)/(i \cdot p) &= t_i/p \qquad \text{for } 1 \leq i \leq n \quad . \end{aligned}$$

The notation $t@p$ picks out the label of the node at address $p$ in the tree $t$, that is, the root label of $t/p$.

Replacing the subtree of $t$ at address $p$ by a tree $t'$, written $t[p \mapsto t']$ is defined as

$$\begin{aligned} t[\epsilon \mapsto t'] &= t' \\ f(t_1, \ldots, t_n)[(i \cdot p) \mapsto t'] &= f(t_1, \ldots, t_i[p \mapsto t'], \ldots, t_n) \\ &\qquad \text{for } 1 \leq i \leq n \quad . \end{aligned}$$

The HEIGHT of a tree $t$, notated $height(t)$, is defined as follows:

$$height(x) = 0 \qquad \text{for } x \in \mathcal{X}$$
$$height(f(t_1, \ldots, t_n)) = 1 + \max_{i=1}^{n} height(t_i) \qquad \text{for } f \in \mathcal{F}^{(n)}$$

We can use trees with variables as CONTEXTS in which to place other trees. A tree in $\mathcal{T}(\mathcal{F}, \mathcal{X}_n)$ will be called a context, typically denoted with the symbol $C$. The notation $C[t_1, \ldots, t_n]$ for $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F})$ denotes the tree in $\mathcal{T}(\mathcal{F})$ obtained by substituting for each $x_i$ the corresponding $t_i$.

More formally, for a context $C \in \mathcal{T}(\mathcal{F}, \mathcal{X}_n)$ and a sequence of $n$ trees $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F})$, the SUBSTITUTION OF $t_1, \ldots, t_n$ INTO $C$, notated $C[t_1, \ldots, t_n]$, is defined inductively as follows:

$$(f(u_1, \ldots, u_m))[t_1, \ldots, t_n] = f(u_1[t_1, \ldots, t_n], \ldots, u_m[t_1, \ldots, t_n])$$
$$x_i[t_1, \ldots, t_n] = t_i$$

## 2.2 *A grammatical metanotation*

We will use a grammatical notation akin to BNF to specify, among other constructs, equations defining functional programs of various sorts. As an introduction to this notation, here is a grammar defining trees over a ranked alphabet and variables (essentially identically to the definition given above):

$$\begin{aligned} f^{(n)} &\in \mathcal{F}^{(n)} \\ x \in \mathcal{X} &::= x_1 \mid x_2 \mid \cdots \\ t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) &::= f^{(m)}(t_1, \ldots, t_m) \\ &\mid \quad x \end{aligned}$$

The notation allows definition of classes of expressions (e.g., $\mathcal{F}^{(n)}$) and specifies metavariables over them ($f^{(n)}$). These classes can be primitive ($\mathcal{F}^{(n)}$) or defined ($\mathcal{X}$), even inductively in terms of other classes or themselves ($\mathcal{T}(\mathcal{F}, \mathcal{X})$). We use the metavariables and subscripted variants on the right-hand side to represent an arbitrary element of the corresponding class. Thus, the elements $t_1, \ldots, t_m$ stand for arbitrary trees in $\mathcal{T}(\mathcal{F}, \mathcal{X})$, and $x$ an arbitrary variable in $\mathcal{X}$. Because numerically subscripted versions of $x$ appear explicitly and individually enumerated as instances of $\mathcal{X}$ (on the right hand side of the rule defining variables), numerically subscripted variables (e.g., $x_1$) on the right-

hand side of all rules are taken to refer to the specific elements of $\mathcal{X}$ (for instance, in the definition (1) of tree transducers), whereas otherwise subscripted elements within the metanotation (e.g., $x_i$, $t_1$, $t_m$) are taken as metavariables.

## 3 TREE TRANSDUCERS, HOMOMORPHISMS, AND AUTOMATA

We review the formal definitions of tree transducers and related constructions for defining tree languages and relations, making use of the grammatical metanotation to define them as functional program classes.

### 3.1 *Tree transducers*

The variation in tree transducer formalisms is extraordinarily wide and the literature vast. For present purposes, we restrict attention to simple nondeterministic tree transducers operating top-down, which transform trees by replacing each node with a subtree as specified by the label of the node and the state of the transduction at that node.

Informally, a TREE TRANSDUCER (specifically a NONDETERMINISTIC TOP-DOWN TREE TRANSDUCER ($\downarrow T\,T$)) specifies a nondeterministic computation from $\mathcal{T}(\mathcal{F})$ to $\mathcal{T}(\mathcal{G})$ defined such that the symbol at the root of the input tree and a current state determines an output context in which the recursive images of the subtrees are placed. Formally, we can define a transducer as a kind of functional program, that is, a set of equations characterized by the following grammar for equations $Eqn$. (The set of states is conventionally notated $Q$, with members notated $q$. One of the states is distinguished as the INITIAL STATE of the transducer.)

$$
\begin{aligned}
q &\in Q \\
f^{(n)} &\in \mathcal{F}^{(n)} \\
g^{(n)} &\in \mathcal{G}^{(n)} \\
x \in \mathcal{X} &::= x_1 \mid x_2 \mid \cdots \\
Eqn &::= q(f^{(n)}(x_1, \ldots, x_n)) \doteq \tau^{(n)} \\
\tau^{(n)} \in \mathcal{R}^{(n)} &::= g^{(m)}({\tau^{(n)}}_1, \ldots, {\tau^{(n)}}_m) \\
&\quad \mid \quad q_j(x_i) \quad \text{where } 1 \leq i \leq n
\end{aligned}
\tag{1}
$$

Intuitively speaking, the expressions in $\mathcal{R}^{(n)}$ are right-hand-side terms using variables limited to the first $n$.

Given this formal description of the set of equations $Eqn$, a tree transducer is defined as a tuple $\langle Q, \mathcal{F}, \mathcal{G}, \Delta, q_0 \rangle$ where[2]

- $Q$ is a finite set of STATES;
- $\mathcal{F}$ is a ranked alphabet of INPUT SYMBOLS;
- $\mathcal{G}$ is a ranked alphabet of OUTPUT SYMBOLS;
- $\Delta \subseteq Eqn$ is a finite set of EQUATIONS;
- $q_0 \in Q$ is a distinguished INITIAL STATE.

Conventional nomenclature refers to the equations as TRANSITIONS, by analogy with transitions in string automata. We use both terms interchangeably. To make clear the distinction between these equations and other equalities used throughout the paper, we use the special equality symbol $\doteq$ for these equations.

The equations define a derivation relation as follows. Given a tree transducer $\langle Q, \mathcal{F}, \mathcal{G}, \Delta, q_0 \rangle$ and two trees $t \in \mathcal{T}(\mathcal{F} \cup \mathcal{G} \cup Q)$ and $t' \in \mathcal{T}(\mathcal{F} \cup \mathcal{G} \cup Q)$, tree $t$ DERIVES $t'$ IN ONE STEP, notated $t \doteq t'$ if and only if there is an equation $u \doteq u' \in \Delta$ with $u \in \mathcal{T}(\mathcal{F} \cup Q, \mathcal{X}_n)$ and $u' \in \mathcal{T}(\mathcal{G} \cup Q, \mathcal{X}_n)$, and a tree $C \in \mathcal{T}(\mathcal{F} \cup \mathcal{G} \cup Q, \mathcal{X}_1)$ in which the variable $x_1$ occurs exactly once and trees $u_1, \ldots, u_n \in \mathcal{T}(\mathcal{F} \cup \mathcal{G})$, such that

$$t = C[u[u_1, \ldots, u_n]]$$

and

$$t' = C[u'[u_1, \ldots, u_n]] \quad .$$

We abuse notation by using the same symbol for the transition equations and the one-step derivation relation they define, and will further extend the abuse to cover the derivation relation's reflexive transitive closure.

The TREE RELATION defined by a $\downarrow TT$ $\langle Q, \mathcal{F}, \mathcal{G}, \Delta, q_0 \rangle$ is the set of all tree pairs $\langle s, t \rangle \in \mathcal{T}(\mathcal{F}) \times \mathcal{T}(\mathcal{G})$ such that $q_0(s) \doteq t$. By virtue of nondeterminism in the equations, multiple equations for a given state $q$ and symbol $f$, tree transducers define true relations rather than merely functions.

[2] We assume without loss of generality that $\mathcal{F}$, $\mathcal{G}$, and $Q$ are disjoint so that their union can itself be taken to be a well-formed ranked alphabet. The elements of the set $Q$ are taken to be ranked symbols of arity 1.

By way of example, the equation grammar above allows the definition of the following set of equations defining a tree transducer:[3]

$$\begin{aligned} q(f(x)) &\doteq g(q'(x), q(x)) \\ q(a) &\doteq a \end{aligned}$$

$$\begin{aligned} q'(f(x)) &\doteq f(q'(x)) \\ q'(a) &\doteq a \end{aligned}$$

This transducer allows for the following derivation:

$$\begin{aligned} q(f(f(a))) &\doteq g(q'(f(a)), q(f(a))) \\ &\doteq g(f(q'(a)), g(q'(a), q(a))) \\ &\doteq g(f(a), g(a, a)) \end{aligned}$$

### 3.2 *Subvarieties of transducers*

Important subvarieties of the basic transducers can be defined by restricting the trees $\tau$ that form the right-hand sides of equations, the elements of $\mathcal{R}^{(n)}$ used.

Recall that each equation is of the form

$$q(f^{(n)}(x_1, \ldots, x_n)) \doteq \tau^{(n)} \qquad .$$

A transducer is

- LINEAR if for each such equation defining the transducer, $\tau$ is linear, that is, no variable is used more than once;
- COMPLETE if $\tau$ contains every variable in $\mathcal{X}_n$ at least once;
- $\epsilon$-FREE if $\tau \notin \mathcal{X}_n$;
- SYMBOL-TO-SYMBOL if $height(\tau) = 1$; and
- a DELABELING if $\tau$ is complete, linear, and symbol-to-symbol.

[3] We will, in general, leave off the explicit specification of the set of states, input and output ranked alphabet, and initial state when providing example transducers, in the expectation that the sets of states and symbols can be inferred from the equations, and the initial state determined under a convention that it is the state defined in the textually first equation.

Note also that we avail ourselves of consistent renaming of the variables $x_1$, $x_2$, and so forth, where convenient for readability.

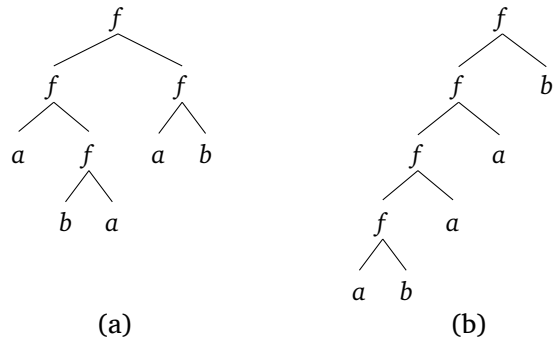



Figure 1:
Local rotation computed by a nonlinear tree transducer. Trees (a) and (b) are in the tree relation of the transducer defined in Section 3.3.

### 3.3 *Nonlinearity deprecated*

The following rules specify a transducer that recursively "rotates" subtrees of the form $f(t_1, f(t_2, t_3))$ to the tree $f(f(t_1, t_2), t_3)$, failing if the required pattern is not found.

$$\begin{aligned} q(f(x,y)) &\doteq f(f(q(x), q_1(y)), q_2(y)) \\ q_1(f(x,y)) &\doteq q(x) \\ q_2(f(x,y)) &\doteq q(y) \\ q(a) &\doteq a \\ q(b) &\doteq b \end{aligned}$$

The tree $f(f(a, f(b,a)), f(a,b))$ is transduced to $f(f(f(f(a,b),a),a),b)$ (as depicted graphically in Figure 1) according to the following derivation:

$$\begin{aligned} & q(f(f(a, f(b,a)), f(a,b))) \\ & \quad \doteq \quad f(f(q(f(a, f(b,a))), q_1(f(a,b))), q_2(f(a,b))) \\ & \quad \doteq \quad f(f(f(f(q(a), q_1(f(b,a))), q_2(f(b,a))), q(a)), q(b)) \\ & \quad \doteq \quad f(f(f(f(a, q(b)), q(a)), a), b) \\ & \quad \doteq \quad f(f(f(f(a,b),a),a),b) \end{aligned}$$

A variant transducer can allow $f$ subtrees to remain unchanged (rather than failing) when the second argument is not itself an $f$ tree. We add a (nondeterministic) equation to allow nonrotation,

$$q(f(x,y)) \doteq f(q(x), q'(y)) \qquad ,$$

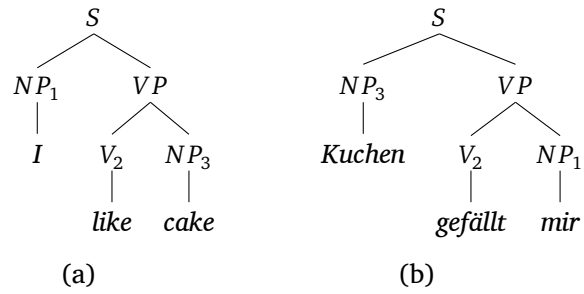



Figure 2: Example of local rotation in language translation divergence. Corresponding nodes are marked with matched subscripts.

which puts the proper constraint on its second subtree $y$ through the new state $q'$ defined by

$$q'(a) \doteq a$$
$$q'(b) \doteq b \quad .$$

This allows, for instance, the "already rotated" tree in Figure 1(b) to transduce to itself.

Note that intrinsic use is made in these examples of the ability to duplicate variables on the right-hand sides of rewrite rules. Transducers without such duplication are *linear*. Linear tree transducers are incapable of performing local rotations of this sort.

Local rotations are typical of natural-language applications. For instance, many of the kinds of translation divergences between languages, such as that exemplified in Figure 2, manifest such rotations. Similarly, semantic bracketing paradoxes can be viewed as necessitating rotations. Thus, linear tree transducers are insufficient for natural-language modeling purposes.

Nonlinearity per se, the ability to make copies during transduction, is not the kind of operation that is characteristic of natural-language phenomena. Furthermore, nonlinear transducers are computationally problematic. The following nonlinear transducer generates a tree that doubles in both width and depth.

$$q(f(x)) \doteq g(f(f(q(x))), f(f(q(x))))$$
$$q(g(x,y)) \doteq g(q(x), q(y))$$
$$q(a) \doteq a$$

For instance, the tree $f(a)$ transduces to

$$g(f(f(a)), f(f(a)))$$

which in turn transduces to

$$\begin{aligned} &g(g(f(f(g(f(f(a)),f(f(a))))),\\ &\quad f(f(g(f(f(a)),f(f(a))))),\\ &\quad g(f(f(g(f(f(a)),f(f(a))))),\\ &\quad f(f(g(f(f(a)),f(f(a)))))) \end{aligned} \quad .$$

Notice that the number of $a$'s in the $i$-th iteration is $2^{2^i-1}$. The size of this transducer's output is exponential in the size of its input. (The existence of such a transducer constitutes a simple proof of the lack of composition closure of tree transducers, as the exponential of an exponential grows faster than exponential.)

In summary, nonlinearity seems inappropriate on computational and linguistic grounds, yet is required for tree transducers to express the kinds of simple local rotations that are typical of natural-language transductions. By contrast, STSG, as described in Section 6, is intrinsically a linear formalism but can express rotations straightforwardly.

### 3.4 *Tree automata and homomorphisms*

Two subcases of tree transducers are especially important. First, tree transducers that implement a partial identity function over their domain are TREE AUTOMATA. These are delabeling tree transducers that preserve the label and the order of arguments. Because they compute only the identity function, tree automata are of interest for the domains over which they are defined, not the mappings they compute. This domain forms a tree language, the tree language recognized by the automaton. The tree languages so recognized are the REGULAR TREE LANGUAGES (or RECOGNIZABLE TREE LANGUAGES). Though the regular tree languages are a superset of the tree languages defined by context-free grammars (the local tree languages), the string languages defined by their yield are coextensive with the context-free languages. We take tree automata to be quadruples by dropping one of the redundant alphabets from the corresponding tree transducer quintuple.

Second, TREE HOMOMORPHISMS are deterministic tree transducers with only a single state, hence essentially stateless. The replacement of a node by a subtree thus proceeds deterministically and independently of its context. Consequently, a homomorphism $h : \mathcal{T}(\mathcal{F}) \rightarrow$

$\mathcal{T}(\mathcal{G})$ is specified by its kernel, a function $\hat{h} : \mathcal{F} \rightarrow \mathcal{T}(\mathcal{G}, \mathcal{X}_\infty)$ such that $\hat{h}(f)$ is a context in $\mathcal{T}(\mathcal{G}, \mathcal{X}_{arity(f)})$ for each symbol $f \in \mathcal{F}$. The kernel $\hat{h}$ is extended to the homomorphism $h$ by the following recurrence:

$$h(f(t_1, \ldots, t_n)) = \hat{h}(f)[h(t_1), \ldots, h(t_n)]$$

that is, $\hat{h}(f)$ acts as a context in which the homomorphic images of the subtrees are substituted.

As with transducers (see Section 3.2), further restrictions can be imposed to generate the subclasses of linear, complete, $\epsilon$-free, symbol-to-symbol, and delabeling tree homomorphisms.

The import of these two subcases of tree transducers lies in the fact that the tree relations defined by certain tree transducers have been shown to be also characterizable by composition from these simplified forms, via an alternate and quite distinct formalization, to which we now turn.

### 3.5 *The bimorphism characterization of tree transducers*

Tree transducers can be characterized directly in terms of equations defining a simple kind of functional program, as above. Bimorphisms constitute an elegant alternative characterization of tree transducers in terms of a constellation of elements of the various subtypes of transducers – homomorphisms and automata – we have introduced.

A BIMORPHISM is a triple $\langle L, h_{in}, h_{out} \rangle$ consisting of a regular tree language $L$ (or, equivalently, a tree automaton) and two tree homomorphisms $h_{in}$ and $h_{out}$ (connoting the input and output respectively). The tree relation $\mathcal{L}$ defined by a bimorphism is the set of tree pairs that are generable from elements of the tree language by the homomorphisms, that is,

$$\mathcal{L}(\langle L, h_{in}, h_{out} \rangle) = \{\langle h_{in}(t), h_{out}(t) \rangle \mid t \in L\} \quad .$$

Depending on the type of tree homomorphisms used in the bimorphism, different classes of tree relations are defined. We can limit attention to bimorphisms in which the input or output homomorphisms are restricted to a certain type: linear ($L$), complete ($C$), $\epsilon$-free ($F$), symbol-to-symbol ($S$), delabeling ($D$), or unrestricted ($M$). We will write $B(I, O)$ where $I$ and $O$ characterize a subclass of homomorphisms for the set of bimorphisms for which the input homomorphism is in the

subclass indicated by $I$ and the output homomorphism is in the subclass indicated by $O$. For example, $B(D, M)$ is the set of bimorphisms for which the input homomorphism is a delabeling but the output homomorphism can be arbitrary.

The tree relations definable by bottom-up tree transducers (closely related to the top-down transducers we use here) turn out to be exactly this class $B(D, M)$. (See the survey by Comon *et al.* (2008, Section 6.5) and works cited therein.) The bimorphism notion thus allows us to characterize certain tree transductions purely in terms of tree automata and tree homomorphisms.

As an example, we consider the rotation transducer of Section 3.3, expressed as a bimorphism. The tree relation for the bimorphism expresses an abstract specification of where the rotations are to occur, picking out such cases with a special symbol $R$ of arity 3, its arguments being the three subtrees participating in the rotation.

$$\begin{aligned} q(A) &\doteq A \\ q(B) &\doteq B \\ q(R(x,y,z)) &\doteq R(q(x), q(y), q(z)) \end{aligned}$$

The input homomorphism maps these trees onto trees prior to rotation.

$$\begin{aligned} q(A) &\doteq a \\ q(B) &\doteq b \\ q(R(x,y,z)) &\doteq f(q(x), f(q(y), q(z))) \end{aligned}$$

Notice that the trees rooted in $R$ map onto a tree configuration that should be rotated.

The output homomorphism maps each tree onto the corresponding post-rotation tree:

$$\begin{aligned} q(A) &\doteq a \\ q(B) &\doteq b \\ q(R(x,y,z)) &\doteq f(f(q(x), q(y)), q(z)) \end{aligned}$$

Again, to allow the option of nonrotating configurations, we can add to the control trees nodes labeled $F$ that should map onto configurations that cannot be rotated. (New equations are marked with $\Leftarrow$.)

The new $q'$ state guarantees this constraint on the $F$ trees.

$$\begin{aligned} q(A) &\doteq A \\ q(B) &\doteq B \\ q(F(x,y)) &\doteq F(q(x), q'(y)) \quad \Leftarrow \\ q(R(x,y,z)) &\doteq R(q(x), q(y), q(z)) \\ q'(A) &\doteq A \quad \Leftarrow \\ q'(B) &\doteq B \quad \Leftarrow \end{aligned}$$

The input homomorphism maps the new $F$ states onto $f$ trees

$$\begin{aligned} q(A) &\doteq a \\ q(B) &\doteq b \\ q(F(x,y)) &\doteq f(q(x), q(y)) \quad \Leftarrow \\ q(R(x,y,z)) &\doteq f(q(x), f(q(y), q(z))) \end{aligned}$$

as does the output homomorphism.

$$\begin{aligned} q(A) &\doteq a \\ q(B) &\doteq b \\ q(F(x,y)) &\doteq f(q(x), q(y)) \quad \Leftarrow \\ q(R(x,y,z)) &\doteq f(f(q(x), q(y)), q(z)) \end{aligned}$$

## 4 TREE-SUBSTITUTION AND TREE-ADJOINING GRAMMARS

Tree-adjoining grammars (TAG) and tree-substitution grammars (TSG) are grammar formalisms based on tree rewriting, rather than the string rewriting of the Chomsky hierarchy formalisms. Grammars are composed of a set of elementary trees, which are combined according to simple tree operations. In the case of TAG, these operations are substitution and adjunction, in the case of TSG, substitution alone. Synchronous variants of these formalisms extend the base formalism with the synchronization idea presented in earlier work (Shieber 1994). In particular, grammars are composed of pairs of elementary trees, and certain pairs of nodes, one from each tree in a pair, are linked to indi-

cate that operations incorporating trees from a single elementary pair must occur at the linked nodes.

We review here the definition of tree-substitution and tree-adjoining grammars, and their synchronous variants. Since TSG can be thought of as a subset of TAG, we first present TAG, describing the restriction to TSG thereafter. Our presentation of TAG differs slightly from traditional ones in ways that simplify the synchronous variants and the later bimorphism constructions.

### 4.1 *Tree-adjoining grammars*

A tree-adjoining grammar is composed of a set of elementary trees, such as those depicted in Figure 4, that are combined by operations of substitution and adjunction. Traditional presentations of TAG, with which we will assume familiarity, take the symbols in elementary and derived trees to be unranked; nodes labeled with a given nonterminal symbol may have differing numbers of children. (Joshi and Schabes (1997) present a good overview.) For example, foot nodes of auxiliary trees and substitution nodes have no children, whereas the similarly labeled root nodes must have at least one. Similarly, two nodes with the same label but differing numbers of children may match for the purpose of allowing an adjunction (as the root nodes of $\alpha_1$ and $\beta_1$ in Figure 4). In order to integrate TAG with tree transducers, however, we move to a ranked alphabet, which presents some problems and opportunities. (In some ways, the ranked alphabet definition of TAGs is slightly more elegant than the traditional one.)

We will thus take the nodes of TAG trees to be labeled with symbols from a ranked alphabet $\mathcal{F}$; a given symbol then has a fixed arity and a fixed number of children. However, in order to maintain information about which symbols may match for the purpose of adjunction and substitution, we take the elements of $\mathcal{F}$ to be explicitly formed as pairs of an unranked label $e$ and an arity $n$. (For notational consistency, we will use $e$ for unranked and $f$ for ranked symbols.) We will notate these elements, abusing notation, as $e^{(n)}$, and make use of a function $|\cdot|$ to unrank symbols in $\mathcal{F}$, so that $|e^{(n)}| = e$.

To handle foot nodes, for each non-nullary symbol $e^{(i)} \in \mathcal{F}^{(\geq 1)}$, we will associate a new nullary symbol $e_*$, which one can take to be the pair of $e$ and $*$; the set of such symbols will be notated $\mathcal{F}_*$. Similarly, for substitution nodes, $\mathcal{F}_\downarrow$ will be the set of nullary symbols $e_\downarrow$

for all $e^{(i)} \in \mathcal{F}^{(\geq 1)}$. These additional symbols, since they are nullary, will necessarily appear only at the frontier of trees. We will extend the function $|\cdot|$ to provide the unranked symbol associated with these symbols as well, so $|e_{\downarrow}| = |e_{*}| = e$.

A TAG grammar (which we will define more precisely shortly) is based then on a set $P$ of elementary trees, a finite subset of $\mathcal{T}(\mathcal{F} \cup \mathcal{F}_{\downarrow} \cup \mathcal{F}_{*})$, divided into the auxiliary and initial trees depending on whether they do or do not possess a foot node, respectively. In order to allow reference to a particular tree in the set $P$, we associate with each tree a unique name, conventionally notated with a subscripted $\alpha$ or $\beta$ for initial and auxiliary trees respectively. We will abuse notation by using the name and the tree that it names interchangably, and use primed and subscripted variants of $\alpha$ and $\beta$ as variables over initial and auxiliary trees, with $\gamma$ serving for elementary trees in general.

Traditionally in TAG grammars, substitutions are obligatory at substitution nodes (those with labels from $\mathcal{F}_{\downarrow}$) and adjunctions are optional at nodes with labels from $\mathcal{F}$. This presents two problems. First, the optionality of adjunction makes it tricky to provide a canonical fixed-length specification of what trees operate at the various nodes in the tree; such a specification will turn out to be helpful in our definitions of derivation for TAG and synchronous TAG. (This is not a problem for substitution, as the obligatoriness of substitution means that there will be exactly as many trees substituting in as there are substitution nodes.) Second, it is standard within TAG to provide further constraints that disallow adjunction at certain nodes. So far, we have no provision for such NONADJOINING CONSTRAINTS. To address these problems, we use a TAG formalism slightly modified from traditional presentations, one that loses no expressivity in weak generative capacity but is easier for analysis purposes.

First, we make all adjunction obligatory, in the sense that if a node in a tree allows adjunction, an adjunction must occur there. To get the effect of optional adjunction, for instance at a node labeled $B$, we add to the grammar a NONADJUNCTION TREE $\text{NA}_B$, a vestigial auxiliary tree of a single node $B_{*}$, which has no adjunction sites and therefore does not itself modify any tree that it adjoins into. These nonadjunction trees thus found the recursive structure of derivations.[4]

[4] In traditional TAG, all adjunction is optional; adding nonadjunction trees

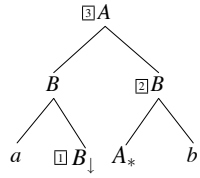



Figure 3:
Sample TAG tree marked with diacritics to show the permutation of operable nodes. Note that the node at address 1 is left out of the set of operable sites; it is thus a nonadjoining node.

Second, now that it is determinate whether an operation must occur at a node, the number of children of a node in a derivation tree is determined by the elementary tree $\gamma$ at that node; it is just the number of adjunction or substitution sites in $\gamma$, the OPERABLE SITES, which we will notate $\overline{\gamma}$. We take $\overline{\gamma}$ to be the set of adjunction and substitution nodes in the tree, that is, all nodes in the tree with the exception of the foot node. (Below, we will allow for nodes to be left out from the set of operable sites, and in Section 8, we generalize this to allow multiple adjunctions at a single site.)

All that is left is to provide a determinate ordering of operable sites in an elementary tree, that is, a permutation $\pi$ on the operable sites $\overline{\gamma}$ (or equivalently, their addresses). This permutation can be thought of as specified as part of the elementary tree itself. For example, the tree in Figure 3, which requires operations at the nodes at addresses $\epsilon$, 12, and 2, may be associated with the permutation $\langle 12, 2, \epsilon \rangle$. The permutation can be marked on the tree itself with numeric diacritics $\boxed{i}$, as shown in the figure.

A nonadjoining constraint on a node can now be implemented merely by removing the node from the operable sites of a tree, and hence from the tree's associated permutation. In the graphical depictions, nonadjoining nodes are those non-substitution nodes that bear no numeric diacritic.

Formally, we define $\mathcal{E}(\mathcal{F})$, the ELEMENTARY TREES over a ranked alphabet $\mathcal{F}$, to be all pairs ${}_{\square}\gamma = \langle \gamma, \pi \rangle$ where $\gamma \in \mathcal{T}(\mathcal{F} \cup \mathcal{F}_{\downarrow} \cup \mathcal{F}_{*})$ and $\pi$ is a permutation of a subset of the nodes in $\gamma$. As above, we use the notation $\overline{\gamma}$ to specify the operable sites of $\gamma$, that is, the domain of $\pi$. The operable sites $\overline{\gamma}$ must contain all substitution nodes in $\gamma$.

---

for all elements of $\mathcal{F}$ is consistent with that practice. Our approach, however, opens the possibility of leaving out nonadjunction trees for one or more symbols, thereby implementing a kind of global obligatory adjunction constraint, less expressive than those variants of TAG that have node-based obligatory adjunction constraints, but more so than the purely adjunction-optional approach.

We further require that the tree $\gamma$ whose root is labeled $f$ contain at most one node labeled with $|f|_* \in \mathcal{F}_*$ and no other nodes labeled in $\mathcal{F}_*$; this node is its foot node, and its address is notated $\text{foot}(\beta)$. The foot node is not an element of $\overline{\gamma}$. Trees with a foot node are AUXILIARY TREES; those with no foot node are INITIAL TREES. The set $\mathcal{E}(\mathcal{F})$ is the set of all possible such elementary trees.

The notation ${}_{\square}\gamma$ is used to indicate an elementary tree, the box as a mnemonic for the box diacritics labeling the permutation. We use similar notations for the particular cases where the elementary tree is initial (${}_{\square}\alpha$) or auxiliary (${}_{\square}\beta$). For convenience, for an elementary tree ${}_{\square}\gamma$, we will use $\gamma$ for its tree component when no confusion results, and will conflate the tree properties of an elementary tree ${}_{\square}\gamma$ and its tree component $\gamma$.

A TAG grammar is then a triple $\langle \mathcal{F}, P, S \rangle$, where $\mathcal{F}$ is a ranked alphabet; $P$ is the set of ELEMENTARY TREES, a finite subset of $\mathcal{E}(\mathcal{F})$; and $S \in \mathcal{F}_{\downarrow}$ is a distinguished INITIAL SYMBOL. We further partition the set $P$ into the set $I$ of initial trees in $P$ and the set $A$ of auxiliary trees in $P$. A simple TAG grammar is depicted in Figure 4; $\alpha_1$ and $\alpha_2$ are initial trees, and $\beta_1$ and $\beta_2$ are auxiliary trees.

### 4.2 *The substitution and adjunction operations*

We turn now to the operations used to derive more complex trees from the elementary trees. It is convenient to notationally distinguish derived trees that have the *form* of an initial or auxiliary tree, that is, (respectively) lacking or bearing a foot node. We use the bolded symbols $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ for derived trees in $\mathcal{T}(\mathcal{F} \cup \mathcal{F}_{\downarrow} \cup \mathcal{F}_*)$ without and with foot nodes, respectively, again using $\boldsymbol{\gamma}$ when being agnostic as to the form.

The trees are combined by two operations, SUBSTITUTION and ADJUNCTION. Under substitution, a node labeled $e_{\downarrow}$ (at address $p$) in a tree $\boldsymbol{\gamma}$ can be replaced by an initial-form tree $\boldsymbol{\alpha}$ with the corresponding label $f$ at the root when $|f| = e$. The resulting tree, the substitution of $\boldsymbol{\alpha}$ at $p$ in $\boldsymbol{\gamma}$, is

$$\boldsymbol{\gamma}[\text{SUBST}_p\ \boldsymbol{\alpha}] \equiv \boldsymbol{\gamma}[p \mapsto \boldsymbol{\alpha}] \qquad .$$

Under adjunction, an internal node of $\boldsymbol{\gamma}$ at $p$ labeled $f \in \mathcal{F}$ is *split apart*, replaced by an auxiliary-form tree $\boldsymbol{\beta}$ rooted in $f'$ when $|f| = |f'|$. The

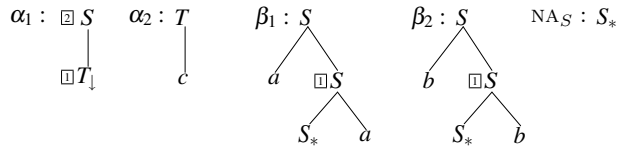



Figure 4:
Sample TAG for the copy language $\{wcw \mid w \in \{a,b\}^*\}$. The initial symbol is $S_\downarrow$.

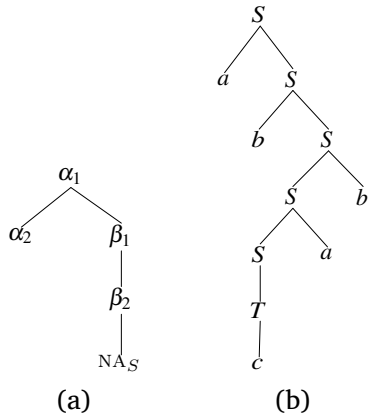



Figure 5:
Derivation and derived trees for the sample grammar of Figure 4: (a) derivation tree, (b) corresponding derived tree.

resulting tree, the adjunction of $\boldsymbol{\beta}$ at $p$ in $\boldsymbol{\gamma}$, is

$$\boldsymbol{\gamma}[\mathrm{ADJ}_p\ \boldsymbol{\beta}] \equiv \boldsymbol{\gamma}[p \mapsto \boldsymbol{\beta}[\mathrm{foot}(\boldsymbol{\beta}) \mapsto \boldsymbol{\gamma}/p]] \qquad .$$

This definition (by requiring $f$ to be in $\mathcal{F}$, not $\mathcal{F}_*$ or $\mathcal{F}_\downarrow$) is consistent with the standard convention, without loss of expressivity, that adjunction is disallowed at foot nodes and substitution nodes.

For uniformity, we will notate these operations with a single operator $\mathrm{OP}_p$ defined as follows:

$$\boldsymbol{\gamma}[\mathrm{OP}_p\ \boldsymbol{\gamma}'] \equiv \begin{cases} \boldsymbol{\gamma}[\mathrm{SUBST}_p\ \boldsymbol{\gamma}'] & \text{if } \boldsymbol{\gamma}@p \in \mathcal{F}_\downarrow \\ \boldsymbol{\gamma}[\mathrm{ADJ}_p\ \boldsymbol{\gamma}'] & \text{otherwise} \end{cases}$$

### 4.3 *Derivation trees and the derivation relation*

A derivation tree $D$ records the operations over the elementary trees used to derive a given derived tree. Each node in the derivation tree specifies an elementary tree ${}_\square\gamma$, with the node's child subtrees $D_i$ recording the derivations for trees that are adjoined or substituted into that tree at the corresponding operable nodes.

A DERIVATION for a grammar $G = \langle \mathcal{F}, P, S \rangle$ is a tree whose nodes are labeled with elementary trees, that is, a tree $D$ in $\mathcal{T}(P)$. We here

interpret $P$ itself as a ranked alphabet, where for each ${}_{\square}\gamma = \langle \gamma, \pi \rangle \in P$, we take its arity to be $arity({}_{\square}\gamma) \equiv |\pi|$. This requirement enforces the constraint that nodes in a derivation tree labeled with ${}_{\square}\gamma$ will have exactly the right number of children to specify the subtrees to be used at each of the operable sites in ${}_{\square}\gamma$. We add an additional constraint:

**Labels match:** For each node in $D$ labeled with ${}_{\square}\gamma = \langle \gamma, \pi \rangle$, and for all $i$ where $1 \leq i \leq arity({}_{\square}\gamma)$, the root node of the $i$-th child of ${}_{\square}\gamma$, labeled with ${}_{\square}\gamma_i$, must match the corresponding operable site in ${}_{\square}\gamma$, that is,

$$|\gamma @ \pi_i| = |\gamma_i @ \epsilon| \quad .$$

(The notation $\gamma @ \pi_i$ can be thought of as the node in $\gamma$ labeled by diacritic $\boxed{i}$.)

A derivation is COMPLETE if it is rooted in an initial tree that is itself rooted in the initial symbol:

**Initial symbol at root:** The tree ${}_{\square}\alpha_r$ at the root of the derivation tree must be an initial tree labeled at its root by the initial symbol; that is, $|\alpha_r @ \epsilon| = |S|$.[5]

For example, the tree in Figure 5(a) is a well-formed complete derivation tree for the grammar in Figure 4. Note, for instance, that $|\alpha_1 @ \pi_2| = S = |\beta_1 @ \epsilon|$ as required by the label-matching constraint, and the root is an initial tree $\alpha_1$ whose root is consistent with the initial symbol $S_{\downarrow}$.

A simple tree automaton can check these conditions, and therefore define the set of well-formed complete derivation trees. This automaton is constructed as follows. The states of the automaton are the set $\{q_N \mid N \in |\mathcal{F}|\}$, one for each unranked vocabulary symbol in the derived tree language. The start state is $q_{|S|}$. For each tree ${}_{\square}\gamma = \langle \gamma, \pi \rangle \in P$, of arity $n$ and rooted with the symbol $N$, there is a transition of the form

$$q_{|N|}({}_{\square}\gamma(x_1, \ldots, x_n)) \doteq {}_{\square}\gamma(q_{|\gamma @ \pi_1|}(x_1), \ldots, q_{|\gamma @ \pi_n|}(x_n)) \quad . \quad (2)$$

The set of well-formed derivation trees is thus a regular tree set.

[5] The stripping of ranks and diacritics is necessary to allow, for instance, the initial symbol to match root nodes of differing arities.

For the grammar of Figure 4, the automaton defining well-formed derivation trees is given by

$$\begin{aligned} q_S(\alpha_1(x,y)) &\doteq \alpha_1(q_T(x), q_S(y)) \\ q_T(\alpha_2) &\doteq \alpha_2 \\ q_S(\beta_1(x)) &\doteq \beta_1(q_S(x)) \\ q_S(\beta_2(x)) &\doteq \beta_2(q_S(x)) \\ q_S(\mathrm{NA}_S) &\doteq \mathrm{NA}_S \end{aligned}$$

which recognizes the tree of Figure 5(a):

$$\begin{aligned} q_S(\alpha_1(\alpha_2, \beta_1(\beta_2(\mathrm{NA}_S)))) &\doteq \alpha_1(q_T(\alpha_2), q_S(\beta_1(\beta_2(\mathrm{NA}_S)))) \\ &\doteq \alpha_1(\alpha_2, \beta_1(q_S(\beta_2(\mathrm{NA}_S)))) \\ &\doteq \alpha_1(\alpha_2, \beta_1(\beta_2(q_S(\mathrm{NA}_S)))) \\ &\doteq \alpha_1(\alpha_2, \beta_1(\beta_2(\mathrm{NA}_S))) \end{aligned}$$

The DERIVATION RELATION $\mathcal{D}$, that is, the relation between derivation trees and the derived trees that they specify, can be simply defined via the hierarchical iterative operation of trees at operable sites. In particular, for a derivation tree with root labeled with the elementary tree ${}_{\square}\gamma = \langle \gamma, \pi \rangle$ of arity $n$, we define

$$\mathcal{D}({}_{\square}\gamma(t_1, \ldots, t_n)) \equiv \gamma[\mathrm{OP}_{\pi_1}\, \mathcal{D}(t_1), \mathrm{OP}_{\pi_2}\, \mathcal{D}(t_2), \ldots, \mathrm{OP}_{\pi_n}\, \mathcal{D}(t_n)]$$

where, following Schabes and Shieber (1994), the right-hand side specifies the *simultaneous* application of the specified operations. We define this in terms of the *sequential* application of operations as follows:

$$\begin{aligned} &\gamma[\mathrm{OP}_{p_1}\, \gamma_1, \mathrm{OP}_{p_2}\, \gamma_2, \ldots, \mathrm{OP}_{p_n}\, \gamma_n] \\ &\qquad \equiv \gamma[\mathrm{OP}_{p_1}\, \gamma_1][\mathrm{OP}_{update(p_2,\gamma_1,p_1)}\, \gamma_2, \ldots, \mathrm{OP}_{update(p_n,\gamma_1,p_1)}\, \gamma_n] \end{aligned} \quad (3)$$

The *update* function adjusts the paths at which later operations take place to compensate for an earlier adjunction. (Recall the notations $q \prec p$ for $q$ a proper prefix of $p$ and $p - q$ for the sequence obtained by removing the prefix $q$ from $p$.)

$$update(p, \gamma, q) \equiv \begin{cases} p & \text{if } \gamma \text{ is an initial-form tree} \\ p & \text{if } \gamma \text{ is an auxiliary-form tree and } q \not\prec p \\ q \cdot \mathrm{foot}(\gamma) \cdot (p - q) & \\ & \text{if } \gamma \text{ is an auxiliary-form tree and } q \prec p \end{cases}$$

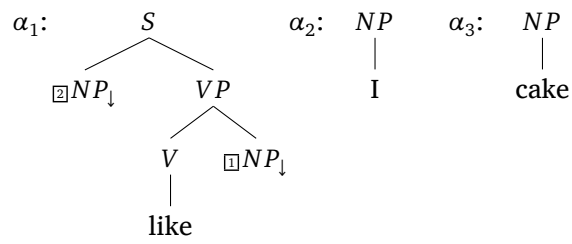



Figure 6: Grammar for a tiny English fragment.

Schabes and Shieber (1994) prove that adjunctions at distinct sites commute: if $p \neq q$ then

$$\gamma[\ldots, \mathrm{ADJ}_p\ \gamma_1, \mathrm{ADJ}_q\ \gamma_2, \ldots] = \gamma[\ldots, \mathrm{ADJ}_q\ \gamma_2, \mathrm{ADJ}_p\ \gamma_1, \ldots] \qquad (4)$$

that is, that the order of adjunctions is immaterial according to this definition. The proof applies equally well to substitution and mixtures of operations. This proves that the order of the permutation over operable sites is truly arbitrary; any order will yield the same result. (In Section 8, the introduction of multiple adjunction presents the potential for noncommutativity. We address the issue in that section.)

As the base case, this definition gives, as expected,

$$\mathcal{D}(_{\square}\gamma) \equiv \gamma$$

for elementary trees of arity 0, that is, trees with no operable sites.

### 4.4 *Tree-substitution grammars*

Tree-substitution grammars are simply tree-adjoining grammars with no auxiliary trees, so that the elementary trees are only combined by substitution.

As a simple natural-language example, we consider the grammar with three elementary trees of Figure 6 and initial symbol $S$. The arities of the symbols should be clear from their usage and the associated permutations from the link diacritics.

As in Section 4.3, the derived tree for a derivation tree $D$ is generated by performing all of the requisite substitutions. In this section, we provide a new definition of the derivation relation between a derivation tree and the derived tree it specifies as a simple homomorphism $h_{\mathcal{D}}$, and prove that this definition is equivalent to that of Section 4.3.

We define $h_{\mathcal{D}}$ in equational form. For each elementary tree ${}_{\square}\alpha \in P$, there is an equation of the form

$$h_{\mathcal{D}}({}_{\square}\alpha(x_1, \ldots, x_n)) \doteq \lfloor {}_{\square}\alpha \rfloor$$

where the right-hand-side transformation $\lfloor \cdot \rfloor$ is defined by

$$\begin{aligned} \lfloor A(t_1, \ldots, t_n) \rfloor &= A(\lfloor t_1 \rfloor, \ldots, \lfloor t_n \rfloor) \\ \lfloor {}_{\boxed{k}}A_{\downarrow} \rfloor &= h_{\mathcal{D}}(x_k) \qquad . \end{aligned} \tag{5}$$

Essentially, this transformation replaces each operable site $\pi_i$ by the homomorphic image of the corresponding variable $x_i$, that is,

$$\lfloor \alpha \rfloor = \alpha[\pi_1 \mapsto h_{\mathcal{D}}(x_1)] \ldots [\pi_n \mapsto h_{\mathcal{D}}(x_n)]$$

for a tree $\alpha$ with $n$ substitution sites in its permutation $\pi$.

### 4.5 *An example derivation*

Returning to the example, the equations corresponding to the elementary trees of Figure 6 are

$$\begin{aligned} h_{\mathcal{D}}(\alpha_1(x_1, x_2)) &\doteq S(h_{\mathcal{D}}(x_2), VP(V(like), h_{\mathcal{D}}(x_1))) \\ h_{\mathcal{D}}(\alpha_2) &\doteq NP(I) \\ h_{\mathcal{D}}(\alpha_3) &\doteq NP(cake) \qquad . \end{aligned}$$

We define the derived tree corresponding to a derivation tree $D$ as the application of this homomorphism to $D$, that is $h_{\mathcal{D}}(D)$. For the example above, the derived tree is that shown in Figure 2(a):

$$\begin{aligned} h_{\mathcal{D}}(\alpha_1(\alpha_3, \alpha_2)) &\doteq S(h_{\mathcal{D}}(\alpha_2), VP(V(like), h_{\mathcal{D}}(\alpha_3)) \\ &\doteq S(NP(I), VP(V(like), NP(cake))) \end{aligned}$$

By composing the automaton recognizing well-formed derivation trees with the homomorphism above, we can construct a single transducer doing the work of both. We do this explicitly for TAG in Section 7.1.

Note that, by construction, each variable occurs exactly once on the right-hand side of a given equation. Thus, this homomorphism $h_{\mathcal{D}}$ is linear and complete.

### 4.6 *Equivalence of $\mathcal{D}$ and $h_{\mathcal{D}}$*

We can show that this definition in terms of the linear complete homomorphism $h_{\mathcal{D}}$ is equivalent to the traditional definition $\mathcal{D}$:

$$\mathcal{D}(D) = h_{\mathcal{D}}(D) \tag{6}$$

The proof is by induction on the height of $D$. Since $h_{\mathcal{D}}$ is the identity function everywhere except at operable sites,

$$\mathcal{D}({}_{\square}\alpha) = \alpha = h_{\mathcal{D}}({}_{\square}\alpha) \quad .$$

This serves as the base case for the induction.

Now suppose, that Equation (6) holds for trees of height $k$, and consider tree ${}_{\square}\alpha(D_1, \ldots, D_n)$ of height $k+1$. Then

$$\begin{aligned}
\mathcal{D}({}_{\square}\alpha(D_1, \ldots, D_n)) &= \alpha[\textsc{subst}_{\pi_1}\ \mathcal{D}(D_1), \ldots, \textsc{subst}_{\pi_n}\ \mathcal{D}(D_n)] \\
&= \alpha[\textsc{subst}_{\pi_1}\ \mathcal{D}(D_1)] \ldots [\textsc{subst}_{\pi_n}\ \mathcal{D}(D_n)] \\
&= \alpha[\pi_1 \mapsto \mathcal{D}(D_1)] \ldots [\pi_n \mapsto \mathcal{D}(D_n)] \\
&= \alpha[\pi_1 \mapsto h_{\mathcal{D}}(D_1)] \ldots [\pi_n \mapsto h_{\mathcal{D}}(D_n)] \qquad \Leftarrow \\
&= \alpha[\pi_1 \mapsto h_{\mathcal{D}}(x_1)] \ldots [\pi_n \mapsto h_{\mathcal{D}}(x_n)][D_1, \ldots, D_n] \\
&= \lfloor \alpha \rfloor [D_1, \ldots, D_n] \\
&= h_{\mathcal{D}}({}_{\square}\alpha(D_1, \ldots, D_n)) \quad .
\end{aligned}$$

The marked step applies the induction hypothesis.

Later, in Section 7 we will provide a similar reformulation of the derivation relation for tree-adjoining grammars. To do so, however, requires additional power beyond simple tree homomorphisms, which is the subject of that section.

## 5 SYNCHRONOUS GRAMMARS

We perform synchronization of tree-adjoining and tree-substitution grammars as per the approach taken in earlier work (Shieber 1994). Synchronous grammars consist of pairs of elementary trees with a linking relation between operable sites in each tree. Simultaneous operations occur at linked nodes. In the case of synchronous tree-substitution grammars, the composition operation is substitution, so the linked nodes are substitution nodes.

We define a synchronous tree-adjoining grammar, then, as a quintuple $G = \langle \mathcal{F}_{in}, \mathcal{F}_{out}, P, S_{in}, S_{out} \rangle$, where

- $\mathcal{F}_{in}$ and $\mathcal{F}_{out}$ are the input and output ranked alphabets, respectively,
- $S_{in} \in \mathcal{F}_{in\downarrow}$ and $S_{out} \in \mathcal{F}_{out\downarrow}$ are the input and output initial symbols, and
- $P$ is a set of elementary linked tree pairs, each of the form $\langle \gamma_{in}, \gamma_{out}, \frown \rangle$, where $\gamma_{in} \in \mathcal{T}(\mathcal{F}_{in} \cup \mathcal{F}_{in\downarrow} \cup \mathcal{F}_{in*})$ and $\gamma_{out} \in \mathcal{T}(\mathcal{F}_{in} \cup \mathcal{F}_{in\downarrow} \cup \mathcal{F}_{in*})$ are input and output trees and $\frown \subseteq \overline{\gamma_{in}} \times \overline{\gamma_{out}}$ is a bijection between operable sites from the two trees.

We define $G_{in} = \langle \mathcal{F}_{in}, P_{in}, S_{in} \rangle$ where $P_{in} = \{\langle \gamma, \pi_{in} \rangle \mid \langle \gamma, \gamma', \frown \rangle \in P\}$; this is the left projection of the synchronous grammar onto a simple TAG. The right projection $G_{out}$ is defined similarly. Recall that the elementary trees in this grammar need a permutation on their operable sites. In order to guarantee that derivations for the synchronized grammars are isomorphic, the permutations for the operable sites for paired trees should be consistent. We therefore choose an arbitrary permutation $\langle p_{in,1} \frown p_{out,1}, \ldots, p_{in,n} \frown p_{out,n} \rangle$ over the linked pairs, and take the permutations $\pi_{in}$ for $\gamma_{in}$ and $\pi_{out}$ for $\gamma_{out}$ to be defined as $\pi_{in} = \langle p_{in,1}, \ldots, p_{in,n} \rangle$ and $\pi_{out} = \langle p_{out,1}, \ldots, p_{out,n} \rangle$. Since $\frown$ is a bijection, these projections are permutations as required.

A synchronous derivation was originally defined (Shieber 1994) as a pair $\langle D_{in}, D_{out} \rangle$ where[6]

1. $D_{in}$ is a well-formed derivation tree for $G_{in}$, and $D_{out}$ is a well-formed derivation tree for $G_{out}$, and
2. $D_{in}$ and $D_{out}$ are isomorphic.[7]

The derived tree pair for derivation $\langle D_{in}, D_{out} \rangle$ is then $\langle \mathcal{D}(D_{in}), \mathcal{D}(D_{out}) \rangle$.

[6] In our earlier definition (Shieber 1994), a third condition required that the isomorphic operations be sanctioned by links in tree pairs. This condition can be dropped here, as it follows from the previous definitions. In particular, since the permutations for paired trees are chosen to be consistent, it follows that the isomorphic children of isomorphic nodes are substituted at linked paths.

[7] By "isomorphism" here, we mean the normal sense of isomorphism of rooted trees where the elementary-tree-pairing relation in $P$ serves as the bijection witnessing the isomorphism.

Presentations of synchronous tree-adjoining grammars typically weaken the requirement that the linking relation be a bijection; multiple links are allowed to impinge on a single node. One of two interpretations is possible in this case. We might require that if multiple links impinge upon a node, only one of the links be used. Under this interpretation, the multiple links at a node can be thought of as abbreviatory for a set of trees, each of which contains only one of the links. (The abbreviated form allows for exponentially fewer trees, however.) Thus, the formalism is equivalent to the one described in this section in terms of bijective link relations. Alternatively, we might allow true multiple adjunction of nontrivial trees, which requires an extended notion of derivation tree and derivation relation. This interpretation, proposed by Schabes and Shieber (1994), is arguably better motivated. We defer discussion of multiple adjunction to Section 8, where we address the issue in detail.

## 6 THE BIMORPHISM CHARACTERIZATION OF STSG

The central result we provide relating STSG to tree transducers is this: STSG is weakly equivalent to $B(LC, LC)$, that is, equivalent in the characterized string relations. To show this, we must demonstrate that any STSG is reducible to a bimorphism, and vice versa.

### 6.1 *Reducing STSG to* $B(LC, LC)$

Given an STSG $G = \langle \mathcal{F}_{in}, \mathcal{F}_{out}, P, S_{in}, S_{out} \rangle$, we need to construct a bimorphism characterizing the same tree relation. All the parts are in place to do this. We start by defining a language $\mathbb{D}$ of synchronous derivation trees, which recasts synchronous derivations as single derivation trees from which the left and right derivation trees can be projected via homomorphisms. Rather than taking a synchronous derivation to be a pair of isomorphic trees $D_{in}$ and $D_{out}$, we take it to be the single tree $D$ isomorphic to both, whose element at address $p$ is the elementary tree pair in $P$ that includes $D_{in}@p$ and $D_{out}@p$. The two synchronized derivations $D_{in}$ and $D_{out}$ can be separately recovered by projecting this new derivation tree on its first and second elements via homomorphisms: $h_{in}$ that projects on the first component and $h_{out}$ that projects on the second, respectively. These homomorphisms are trivially linear and complete (indeed, they are mere delabelings).

We define the set $\mathbb{D}$ of well-formed synchronous derivation trees to be the set of trees $D \in \mathcal{T}(P)$ such that $h_{in}(D)$ and $h_{out}(D)$ are both well-formed derivation trees as per Section 4.3. Since tree automata are closed under inverse homomorphism and intersection, the set $\mathbb{D}$ is a regular tree language.

The fact that for any tree $D \in \mathbb{D}$, $h_{in}(D)$ and $h_{out}(D)$ are well-formed derivation trees for their respective TSGs is trivial by construction. It is also trivial to show that any paired derivation has a corresponding synchronous derivation tree in $\mathbb{D}$.

For a given derivation tree $D \in \mathbb{D}$, the paired derived trees can be constructed as $h_{\mathcal{D}}(h_{in}(D))$ and $h_{\mathcal{D}}(h_{out}(D))$, respectively. Thus the mappings from the derivation tree to the derived trees are the compositions of two linear complete homomorphisms, hence linear complete homomorphisms themselves. We take the bimorphism characterizing the STSG tree relation to be $\langle \mathbb{D}, h_{\mathcal{D}} \circ h_{in}, h_{\mathcal{D}} \circ h_{out} \rangle$. Thus, the tree relation defined by the STSG is in $B(LC, LC)$.

## 6.2 *Reducing $B(LC, LC)$ to STSG*

The other direction is somewhat trickier to prove. Given a bimorphism $\langle L, h_{in}, h_{out} \rangle$ over input and output alphabets $\mathcal{F}_{in}$ and $\mathcal{F}_{out}$, respectively, we construct a corresponding STSG $G = \langle \mathcal{F}'_{in}, \mathcal{F}'_{out}, P, S_{in}, S_{out} \rangle$. By "corresponding", we mean that the tree relation defined by the bimorphism is obtainable from the tree relation defined by the STSG via simple homomorphisms of the input and output that eliminate the nodes labeled in $Q$ (as described below). The tree yields are unchanged by these homomorphisms; thus, the string relations defined by the bimorphism and the synchronous grammar are identical.

As the language $L$ is a regular tree language, it is generable by a nondeterministic tree automaton $h_D = \langle Q, \mathcal{F}_d, \Delta, q_0 \rangle$. We use the states of this automaton in the input and output alphabets of the STSG. The input alphabet of the STSG is $\mathcal{F}'_{in} = \mathcal{F}_{in} \cup Q$, composed of the input symbols of the bimorphism, along with the states of the automaton (taken to be symbols of arity 1), and similarly for the output alphabet. The state symbols mark the places in the tree where substitutions occur, allowing control for appropriate substitutions. It is these state symbols that can be eliminated by a simple homomorphism.[8]

[8] In previous work (Shieber 2004), we used a construction that did not in-

The basic idea of the STSG construction is to construct an elementary tree pair corresponding to each compatible pair of transitions in the transducer $h_D \circ h_{in} = \langle Q_{in}, \mathcal{F}_d, \mathcal{F}_{in}, \Delta_{in}, q_{in,0} \rangle$ and $h_D \circ h_{out} = \langle Q_{out}, \mathcal{F}_d, \mathcal{F}_{out}, \Delta_{out}, q_{out,0} \rangle$. For each pair of transitions of the form

$$q_{in,i}(f(x_1, \ldots, x_n)) \doteq \tau_{in} \in \Delta_{in}$$

and

$$q_{out,j}(f(x_1, \ldots, x_n)) \doteq \tau_{out} \in \Delta_{out}$$

we construct a tree pair

$$\langle q_{in,i}(\lceil \tau_{in} \rceil), q_{out,j}(\lceil \tau_{out} \rceil) \rangle$$

where the following transformation is applied to the right-hand sides of the transitions to form the body of the synchronized trees:

$$\lceil f(t_1, \ldots, t_m) \rceil = f(\lceil t_1 \rceil, \ldots, \lceil t_m \rceil)$$
$$\lceil q_j(x_k) \rceil = \boxed{k} q_{j\downarrow}$$

Note that this transformation generates the tree along with a permutation of the operable sites (all substitution nodes) in the tree, and that there will be exactly $n$ such sites in each element of the tree pair, since the transitions are linear and complete by hypothesis. Thus, the two permutations define an appropriate linking relation, which we take to be the synchronous grammar linking relation for the tree pair.

An example may clarify the construction. Take the language of the bimorphism to be defined by the following two-state automaton:

$$q(f(x, y)) \doteq f(q'(x), q'(y))$$
$$q(a) \doteq a$$
$$q'(g(x)) \doteq g(q(x))$$

troduce any extra tree structure in the STSG, so that the trees generated by the bimorphism relation could be recovered by a delabeling rather than a homomorphism deleting extra nodes. However, the proof of equivalence was considerably more subtle, and did not generalize as readily to the case of STAG. Nonetheless, it is useful to note that even more faithful STSG reconstructions of bimorphisms are possible.

Alternately, the definition of STSG (and similarly, STAG) can be modified to incorporate finite-state information explicitly at operable sites. By adding in this information, the bookkeeping done here can be folded into the states, allowing for a stricter strong-generative capacity equivalence. This elegant approach is pursued by Büchse *et al.* (2014).

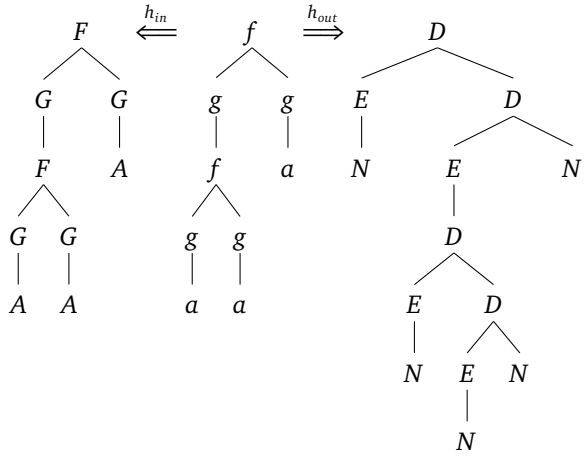



Figure 7: Example of bimorphism construction

This automaton uses the states to alternate $g$'s with $f$'s and $a$'s level by level. For instance, it admits the middle tree in Figure 7. With input and output homomorphisms defined by

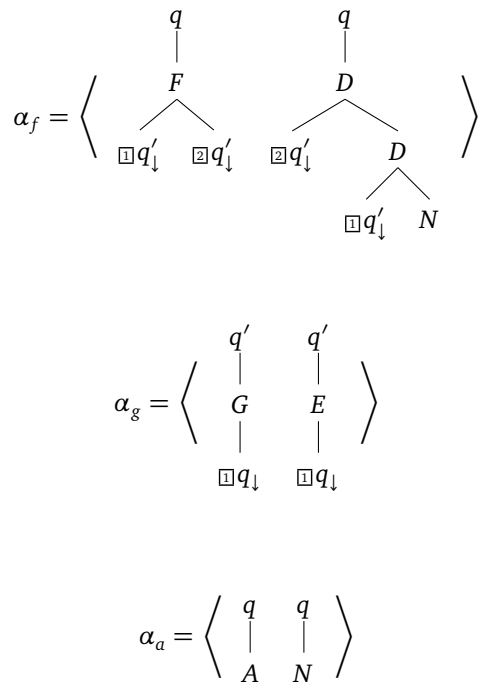
$$\hat{h}_{in}(f) \doteq F(x,y) \qquad \hat{h}_{out}(f) \doteq D(y, D(x,N))$$
$$\hat{h}_{in}(g) \doteq G(x) \qquad \hat{h}_{out}(g) \doteq E(x)$$
$$\hat{h}_{in}(a) \doteq A \qquad \hat{h}_{out}(a) \doteq N$$

the bimorphism so defined generates the tree relation instance exemplified in the figure.

The construction given above generates the elementary tree pairs in Figure 8 for this bimorphism. The reader can verify that the grammar generates a tree pair which corresponds to that shown in Figure 7 generated by the bimorphism after deletion of the state symbols.

By placing STSG in the class of bimorphisms, which have already been used to characterize tree transducers, we synthesize these two independently developed approaches to specifying tree relations. But the relation between a TAG derivation tree and its derived tree is not a mere homomorphism. The appropriate morphism generalizing linear complete homomorphisms to allow adjunction can be used to provide

Figure 8: Generated STSG for previous example of bimorphism construction (in Figure 7)




a bimorphism characterization of STAG as well, further unifying these strands of research. It is to this possibility that we now turn.

## 7 EMBEDDED TREE TRANSDUCERS

We have shown that the string relations defined by synchronous tree-substitution grammars were exactly the relations $B(LC, LC)$. Intuitively speaking, the tree language in such a bimorphism represents the set of derivation trees for the synchronous grammar, and each homomorphism represents the relation between the derivation tree and the derived tree for one of the projected tree-substitution grammars. The homomorphisms are linear and complete because the tree relation between a tree-substitution grammar derivation tree and its associated derived tree is exactly a linear complete tree homomorphism. To characterize the relations defined by synchronous tree-adjoining grammars, it similarly suffices to *find a simple homomorphism-like char-*

*acterization of the tree relation between TAG derivation trees and derived trees*. In Section 7.3 below, we show that linear complete embedded tree homomorphisms (which we introduce next) serve this purpose.

Embedded tree transducers are a generalization of tree transducers in which states are allowed to take a single additional argument in a restricted manner. They correspond to a restrictive subcase of macro tree transducers with one recursion variable. We use the term "embedded tree transducer" rather than the more cumbersome "monadic macro tree transducer" for brevity and by analogy with embedded pushdown automata (Schabes and Vijay-Shanker 1990), another automata-theoretic characterization of the tree-adjoining languages.

We modify the grammar of transducer equations to add an extra optional argument to each occurrence of a state $q$. To highlight the special nature of the extra argument, it is written in angle brackets before the input tree argument. We uniformly use the otherwise unused variable $x_0$ for this argument in the left-hand side, and add $x_0$ as a possible right-hand side itself. Finally, right-hand-side occurrences of states may be passed an arbitrary further right-hand-side tree in this argument. (The use of square brackets in the metanotation indicates optionality.)

$$\begin{array}{rcl} q \in Q & & \\ f^{(n)} \in \mathcal{F}^{(n)} & & \\ x \in \mathcal{X} & ::= & x_0 \mid x_1 \mid x_2 \mid \cdots \\ Eqn & ::= & q\langle [x_0] \rangle (f^{(n)}(x_1, \ldots, x_n)) \doteq \tau^{(n)} \\ \tau^{(n)} \in \mathcal{R}^{(n)} & ::= & f^{(m)}({\tau^{(n)}}_1, \ldots, {\tau^{(n)}}_m) \\ & \mid & x_0 \\ & \mid & q_j\langle [{\tau^{(n)}}_j] \rangle (x_i) \quad \text{where } 1 \leq i \leq n \end{array} \tag{7}$$

Embedded transducers are strictly more expressive than traditional transducers, because the extra argument allows unbounded communication between positions unboundedly distant in depth in the output tree. For example, a simple embedded transducer can compute the reversal of a string, transducing $1(2(2(nil)))$ to $2(2(1(nil)))$, for instance. (This is not computable by a traditional tree transducer.) It is given by the following equations:

$$\begin{aligned} r\langle\rangle(nil) &\doteq nil \\ r\langle\rangle(1(x)) &\doteq s\langle 1(nil)\rangle(x) \\ r\langle\rangle(2(x)) &\doteq s\langle 2(nil)\rangle(x) \\ s\langle x_0\rangle(nil) &\doteq x_0 \\ s\langle x_0\rangle(1(x)) &\doteq s\langle 1(x_0)\rangle(x) \\ s\langle x_0\rangle(2(x)) &\doteq s\langle 2(x_0)\rangle(x) \end{aligned} \tag{8}$$

This is, of course, just the normal accumulating reverse functional program, expressed as an embedded transducer.[9] The additional power of embedded transducers is exactly what is needed to characterize the additional power that TAGs represent over CFGs in describing tree languages, as we will demonstrate in this section. In particular, we show that the relation between a TAG derivation tree and derived tree is characterized by a deterministic linear complete embedded tree transducer (DLCETT).

The first direct presentation of the connection between the tree-adjoining languages and macro tree transducers – the basis for the presentation here – was given in an earlier paper (Shieber 2006). However, the connection may be implicit in a series of previous results in the formal-language theory literature.[10] For instance, Fujiyoshi and Kasai (2000) show that linear, complete monadic context-free tree grammars generate exactly the tree-adjoining languages via a normal form for spine grammars. Separately, the relation between context-free tree grammars and macro tree transducers has been described, where the relationship between the monadic variants of each is implicit. Thus, taken together, an equivalence between the tree-adjoining

[9] A simpler set of equations achieves the same end.

$$\begin{aligned} r\langle\rangle(x) &\doteq s\langle nil\rangle(x) \\ s\langle x_0\rangle(nil) &\doteq x_0 \\ s\langle x_0\rangle(1(x)) &\doteq s\langle 1(x_0)\rangle(x) \\ s\langle x_0\rangle(2(x)) &\doteq s\langle 2(x_0)\rangle(x) \end{aligned} \tag{9}$$

Unfortunately, this set of equations doesn't satisfy the structure of an embedded tree transducer given in Equation (7). Surprisingly, however, the compilation from equations to TAG presented in Section 7.2 applies to this set of equations as well, generating a TAG whose derived trees also reverse its derivation trees.

[10] We are indebted to Uwe Mönnich for this observation.

languages and the image languages of monadic macro tree transducers might be pieced together.

In the present work, we define the relation between tree-adjoining languages and linear complete embedded tree transducers directly, simply, and transparently, by giving explicit constructions in both directions. First, we show that for any TAG we can construct a DLCETT that specifies the tree relation between the derivation trees for the TAG and the derived trees. Then, we show that for any DLCETT we can construct a TAG such that the tree relation between the derivation trees and derived trees is related through a simple homomorphism to the DLCETT tree relation. Finally, we use these results to show that STAG and the bimorphism class $B(ELC, ELC)$ are weakly equivalent, where $ELC$ stands for the class of linear complete embedded homomorphisms.

## 7.1 *From TAG to transducer*

As the first part of the task of characterizing TAG in terms of DLCETT, we show that for any TAG grammar $G = \langle \mathcal{F}, P, S \rangle$, there is a DLCETT $\langle \{h_{\mathcal{D}}\}, P, \mathcal{F}, \Delta, h_{\mathcal{D}} \rangle$ (in fact, an embedded homomorphism), that transduces the derivation trees for the grammar to the corresponding derived trees. This transducer plays the same role for TAG as the definition of $h_{\mathcal{D}}$ in Section 4.3 did for TSG. We define the components of the transducer as follows: The single state, evocatively named $h_{\mathcal{D}}$, is the initial state. The input alphabet is the set of elementary trees $P$ in the grammar, since the input trees are to be the derivation trees of the grammar. The arity of a tree (qua symbol in the input alphabet) is as described in Section 4.3. The output alphabet is that used to define the trees in the TAG grammar, $\mathcal{F}$.

We now turn to the construction of the equations, one for each elementary tree ${}_{\square}\gamma \in P$. Suppose ${}_{\square}\gamma$ has a permutation $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ on its operable sites. (We use this ordering by means of the diacritic representation below.) If $\gamma$ is an auxiliary tree, construct the equation

$$h_{\mathcal{D}}\langle x_0 \rangle({}_{\square}\gamma(x_1, \ldots, x_n)) \doteq \lfloor \gamma \rfloor$$

and if $\gamma$ is an initial tree, construct the equation

$$h_{\mathcal{D}}\langle\rangle({}_{\square}\gamma(x_1, \ldots, x_n)) \doteq \lfloor \gamma \rfloor$$

where the right-hand-side transformation $\lfloor \cdot \rfloor$ is defined by[11]

$$\begin{aligned} \lfloor f(t_1, \ldots, t_n) \rfloor &= f(\lfloor t_1 \rfloor, \ldots, \lfloor t_n \rfloor) \\ \lfloor \boxed{k} f(t_1, \ldots, t_n) \rfloor &= h_{\mathcal{D}} \langle \lfloor f(t_1, \ldots, t_n) \rfloor \rangle (x_k) \\ \lfloor f_* \rfloor &= x_0 \\ \lfloor \boxed{k} f_{\downarrow} \rfloor &= h_{\mathcal{D}} \langle \rangle (x_k) \qquad . \end{aligned} \tag{10}$$

Note that the equations so generated are linear and complete, because each variable $x_i$ is generated once as the tree $\alpha$ is traversed, namely at position $\pi_i$ in the traversal (marked with $\boxed{i}$), and the variable $x_0$ is generated at the foot node only. Thus, the generated embedded tree transducer is linear and complete. Because only one equation is generated per tree, the transducer is trivially deterministic. Because there is only one state, it is a kind of embedded homomorphism.

As noted for TSG in Section 4.3, by composing the automaton recognizing well-formed derivation trees from Section 4.3 with the embedded homomorphism above generating the derived tree, we can construct a single DLCETT doing the work of both. Where the construction of Section 4.3 would generate a transition of the form in Equation 2, repeated here as

$$q_{|N|}({}_{\square}\gamma(x_1, \ldots, x_n)) \doteq {}_{\square}\gamma(q_{|\gamma@\pi_1|}(x_1), \ldots, q_{|\gamma@\pi_n|}(x_n))$$

we compose this transition with the corresponding transition from the previous section

$$h_{\mathcal{D}} \langle x_0 \rangle ({}_{\square}\gamma(x_1, \ldots, x_n)) \doteq \lfloor \gamma \rfloor$$

[11] It may seem like trickery to use the diacritics in this way, as they are not really components of the tree being traversed, but merely manifestations of an extrinsic ordering. But their use is benign. The same transformation can be defined, a bit more cumbersomely, keeping the permutation $\pi$ separate, by tracking the permutation and the current address $p$ in a revised transformation $\lfloor \cdot \rfloor_{\pi,p}$ defined as follows:

$$\begin{aligned} \lfloor f(t_1, \ldots, t_n) \rfloor_{\pi,p} &= f(\lfloor t_1 \rfloor_{\pi,p \cdot 1}, \ldots, \lfloor t_n \rfloor_{\pi,p \cdot n}) \\ \lfloor f(t_1, \ldots, t_n) \rfloor_{\pi,p} &= h_{\mathcal{D}} \langle \lfloor f(t_1, \ldots, t_n) \rfloor_{\pi,p} \rangle (x_{\pi^{-1}(p)}) \\ \lfloor f_* \rfloor_{\pi,p} &= x_0 \\ \lfloor f_{\downarrow} \rfloor_{\pi,p} &= h_{\mathcal{D}} \langle \rangle (x_{\pi^{-1}(p)}) \end{aligned}$$

We then use $\lfloor \alpha \rfloor_{\pi,\epsilon}$ for the transformation of the tree $\alpha$.

or

$$h_{\mathcal{D}}\langle\rangle(_{\square}\gamma(x_1,\ldots,x_n)) \doteq \lfloor\gamma\rfloor$$

for auxiliary and initial trees respectively. The composition construction generates a transducer with states in the cross-product of the states of the input transducers. In this case, since the latter transducer has a single state, we simply reuse the state set of the former, generating

$$q_{|N|}\langle x_0\rangle(_{\square}\gamma(x_1,\ldots,x_n)) \doteq \lfloor\gamma\rfloor$$

or

$$q_{|N|}\langle\rangle(_{\square}\gamma(x_1,\ldots,x_n)) \doteq \lfloor\gamma\rfloor$$

where

$$\begin{aligned}
\lfloor f(t_1,\ldots,t_n)\rfloor &= f(\lfloor t_1\rfloor,\ldots,\lfloor t_n\rfloor)\\
\lfloor \boxed{k} f(t_1,\ldots,t_n)\rfloor &= q_{|f|}\langle\lfloor f(t_1,\ldots,t_n)\rfloor\rangle(x_k)\\
\lfloor f_*\rfloor &= x_0\\
\lfloor \boxed{k} f_{\downarrow}\rfloor &= q_{|f_{\downarrow}|}\langle\rangle(x_k) \qquad .
\end{aligned} \tag{11}$$

### 7.1.1 An example derivation

By way of example, we consider the tree-adjoining grammar given by the following trees:

$$\begin{aligned}
\alpha: &\quad \boxed{1}A(e)\\
\beta_A: &\quad A(\boxed{1}B(a),\boxed{2}C(\boxed{3}D(A_*)))\\
\beta_B: &\quad \boxed{1}B(b,B_*)\\
\mathrm{NA}_B: &\quad B_*\\
\mathrm{NA}_C: &\quad C_*\\
\mathrm{NA}_D: &\quad D_*
\end{aligned}$$

Starting with the auxiliary tree $\beta_A = A(\boxed{1}B(a),\boxed{2}C(\boxed{3}D(A_*)))$, the adjunction sites, corresponding to the nodes labeled $B$, $C$, and $D$ at addresses 1, 2, and 21, have been arbitrarily given a preorder permu-

tation. We therefore construct the equation as follows:

$$\begin{aligned}
h_{\mathcal{D}}\langle x_0\rangle(\beta_A(x_1,x_2,x_3)) &\doteq \lfloor A(\boxed{1}B(a),\boxed{2}C(\boxed{3}D(A_*)))\rfloor \\
&= A(\lfloor\boxed{1}B(a)\rfloor,\lfloor\boxed{2}C(\boxed{3}D(A_*))\rfloor) \\
&= A(h_{\mathcal{D}}\langle\lfloor B(a)\rfloor\rangle(x_1),\lfloor\boxed{2}C(\boxed{3}D(A_*))\rfloor) \\
&= A(h_{\mathcal{D}}\langle B(\lfloor a\rfloor)\rangle(x_1),\lfloor\boxed{2}C(\boxed{3}D(A_*))\rfloor) \\
&= \cdots \\
&= A(h_{\mathcal{D}}\langle B(a)\rangle(x_1),h_{\mathcal{D}}\langle C(h_{\mathcal{D}}\langle D(x_0)\rangle(x_3))\rangle(x_2))
\end{aligned}$$

Similar derivations for the remaining trees yield the (deterministic linear complete) embedded tree transducer defined by the following set of equations:

$$\begin{aligned}
h_{\mathcal{D}}\langle\rangle(\alpha(x_1)) &\doteq h_{\mathcal{D}}\langle A(e)\rangle(x_1) \\
h_{\mathcal{D}}\langle x_0\rangle(\beta_A(x_1,x_2,x_3)) &\doteq A(h_{\mathcal{D}}\langle B(a)\rangle(x_1),h_{\mathcal{D}}\langle C(h_{\mathcal{D}}\langle D(x_0)\rangle(x_3))\rangle(x_2)) \\
h_{\mathcal{D}}\langle x_0\rangle(\beta_B(x_1)) &\doteq h_{\mathcal{D}}\langle B(b,x_0)\rangle(x_1) \\
h_{\mathcal{D}}\langle x_0\rangle(\mathrm{NA}_B()) &\doteq x_0 \\
h_{\mathcal{D}}\langle x_0\rangle(\mathrm{NA}_C()) &\doteq x_0 \\
h_{\mathcal{D}}\langle x_0\rangle(\mathrm{NA}_D()) &\doteq x_0
\end{aligned}$$

We can use this transducer to compute the derived tree for the derivation tree

$$\alpha(\beta_A(\beta_B(\mathrm{NA}_B),\mathrm{NA}_C,\mathrm{NA}_D))$$

as follows:

$$\begin{aligned}
&h_{\mathcal{D}}\langle\rangle(\alpha(\beta_A(\beta_B(\mathrm{NA}_B),\mathrm{NA}_C,\mathrm{NA}_D))) \\
&\quad\doteq h_{\mathcal{D}}\langle A(e)\rangle(\beta_A(\beta_B(\mathrm{NA}_B),\mathrm{NA}_C,\mathrm{NA}_D)) \\
&\quad\doteq A(h_{\mathcal{D}}\langle B(a)\rangle(\beta_B(\mathrm{NA}_B)),h_{\mathcal{D}}\langle C(h_{\mathcal{D}}\langle D(A(e))\rangle(\mathrm{NA}_D))\rangle(\mathrm{NA}_C)) \\
&\quad\doteq A(h_{\mathcal{D}}\langle B(b,B(a))\rangle(\mathrm{NA}_B),C(h_{\mathcal{D}}\langle D(A(e))\rangle(\mathrm{NA}_D))) \\
&\quad\doteq A(B(b,B(a)),C(D(A(e))))
\end{aligned}$$

### 7.1.2 Equivalence of $\mathcal{D}$ and $h_{\mathcal{D}}$

We can now show for TAG derivations, as we did for TSG derivations in Section 4.3, that the embedded homomorphism $h_{\mathcal{D}}$ constructed in this way computes the derivation relation $\mathcal{D}$.

In order to simplify the argument, we take advantage of the commutativity of operations (Equation 4), and assume without loss of generality that each permutation associated with the operable sites of an elementary tree is consistent with a postorder traversal of the nodes in the tree. We can then simplify Equation 3 to

$$\gamma[\mathrm{OP}_{p_1}\ \gamma_1, \mathrm{OP}_{p_2}\ \gamma_2, \ldots, \mathrm{OP}_{p_n}\ \gamma_n] \equiv \gamma[\mathrm{OP}_{p_1}\ \gamma_1][\mathrm{OP}_{p_2}\ \gamma_2, \ldots, \mathrm{OP}_{p_n}\ \gamma_n]$$

since in a postorder traversal, $p_i \not\prec p_{i+k}$.

It will also prove to be useful to have a single notation for the effect of both substitution and adjunction. Recall the definitions of substitution and adjunction:

$$\begin{aligned}\gamma[\mathrm{SUBST}_p\ \boldsymbol{\alpha}] &\equiv \gamma[p \mapsto \boldsymbol{\alpha}]\\ \gamma[\mathrm{ADJ}_p\ \boldsymbol{\beta}] &\equiv \gamma[p \mapsto \boldsymbol{\beta}[\mathrm{foot}(\boldsymbol{\beta}) \mapsto \gamma/p]]\end{aligned}$$

Under the convention that mapping a (nonexistent) "foot" of an initial tree leaves the tree unchanged, that is,

$$\boldsymbol{\alpha}[\mathrm{foot}(\boldsymbol{\alpha}) \mapsto \gamma] \equiv \boldsymbol{\alpha}$$

the two operations collapse notationally, so that we can write

$$\gamma[\mathrm{OP}_p\ \gamma'] \equiv \gamma[p \mapsto \gamma'[\mathrm{foot}(\gamma') \mapsto \gamma/p]]$$

for both substitution and adjunction.

We prove that $\mathcal{D}(D) = h_{\mathcal{D}}\langle\rangle(D)$ for derivations $D$ rooted in an initial tree, and $\mathcal{D}(D)[\mathrm{foot}(\mathcal{D}(D)) \mapsto x] = h_{\mathcal{D}}\langle x\rangle(D)$ for derivations rooted in an auxiliary tree. The proof is again by induction on the height of the derivation $D$.

For the base case, the derivation consists of a single tree with no operable sites. If it is an initial tree $\alpha$, then $\mathcal{D}(\alpha) = \alpha = h_{\mathcal{D}}\langle\rangle(\alpha)$ straightforwardly from the definition of $h_{\mathcal{D}}$, using only the first equation in Equation (10). Similarly, the base case for auxiliary trees,

$$\mathcal{D}(\beta)[\mathrm{foot}(\beta) \mapsto x] = \beta[\mathrm{foot}(\beta) \mapsto x] = h_{\mathcal{D}}\langle x\rangle(\beta)$$

requires only the first and third equations in (10).

For the recursive case,

$$
\begin{aligned}
&h_{\mathcal{D}}\langle\rangle(\alpha(D_1,\ldots,D_n))\\
&\quad = \lfloor\alpha\rfloor[x_1 \mapsto D_1,\ldots,x_n \mapsto D_n]\\
&\quad = \alpha[\pi_1 \mapsto h_{\mathcal{D}}\langle\alpha/\pi_1\rangle(D_1)]\cdots[\pi_n \mapsto h_{\mathcal{D}}\langle\alpha/\pi_n\rangle(D_n)]\\
&\quad = \alpha[\pi_1 \mapsto \mathcal{D}(D_1)[\text{foot}(\mathcal{D}(D_1)) \mapsto \alpha/\pi_1]]\\
&\qquad\quad \cdots[\pi_1 \mapsto \mathcal{D}(D_n)[\text{foot}(\mathcal{D}(D_n)) \mapsto \alpha/\pi_n]] \qquad\qquad \Leftarrow\\
&\quad = \alpha[\text{OP}_{\pi_1}\ \mathcal{D}(D_1)]\cdots[\text{OP}_{\pi_n}\ \mathcal{D}(D_n)]\\
&\quad = \alpha[\text{OP}_{\pi_1}\ \mathcal{D}(D_1),\cdots,\text{OP}_{\pi_n}\ \mathcal{D}(D_n)]\\
&\quad = \mathcal{D}(\alpha(D_1,\ldots,D_n))
\end{aligned}
$$

with the marked step appealing to the induction hypothesis.

Similarly, for derivations rooted in an auxiliary tree,

$$
\begin{aligned}
&h_{\mathcal{D}}\langle x\rangle(\beta(D_1,\ldots,D_n))\\
&\quad = \lfloor\beta\rfloor[x_0 \mapsto x, x_1 \mapsto D_1,\ldots,x_n \mapsto D_n]\\
&\quad = \beta[\text{foot}(\beta) \mapsto x][\pi_1 \mapsto h_{\mathcal{D}}\langle\beta/\pi_1\rangle(D_1)]\\
&\qquad\quad \cdots[\pi_n \mapsto h_{\mathcal{D}}\langle\beta/\pi_n\rangle(D_n)]\\
&\quad = \beta[\text{foot}(\beta) \mapsto x][\pi_1 \mapsto \mathcal{D}(D_1)[\text{foot}(\mathcal{D}(D_1)) \mapsto \beta/\pi_1]]\\
&\qquad\quad \cdots[\pi_1 \mapsto \mathcal{D}(D_n)[\text{foot}(\mathcal{D}(D_n)) \mapsto \beta/\pi_n]]\\
&\quad = \beta[\text{foot}(\beta) \mapsto x][\text{OP}_{\pi_1}\ \mathcal{D}(D_1)]\cdots[\text{OP}_{\pi_n}\ \mathcal{D}(D_n)]\\
&\quad = \beta[\text{foot}(\beta) \mapsto x][\text{OP}_{\pi_1}\ \mathcal{D}(D_1),\ldots,\text{OP}_{\pi_n}\ \mathcal{D}(D_n)]\\
&\quad = \beta[\text{OP}_{\pi_1}\ \mathcal{D}(D_1),\cdots,\text{OP}_{\pi_n}\ \mathcal{D}(D_n)][\text{foot}(\mathcal{D}(\beta(D_1,\ldots,D_n))) \mapsto x]\\
&\quad = \mathcal{D}(\beta(D_1,\ldots,D_n))[\text{foot}(\mathcal{D}(\beta(D_1,\ldots,D_n))) \mapsto x] \qquad .
\end{aligned}
$$

### 7.2 *From transducer to TAG*

Having shown how to construct a DLCETT that captures the relation between derivation trees and derived trees of a TAG, we turn now to showing how to construct a TAG that mimics in its derivation/derived tree relation a DLCETT. Given a linear complete embedded tree transducer $\langle Q, \mathcal{F}, \mathcal{G}, \Delta, q_0\rangle$, we construct a corresponding TAG $\langle \mathcal{G} \cup \dot{Q}, P, \dot{q}_0\rangle$ where the alphabet consists of the output alphabet $\mathcal{G}$ of the transducer together with the disjoint set of unary symbols $\dot{Q} = \{\dot{q}_1,\ldots,\dot{q}_{|Q|}\}$ corresponding to the states of the input transducer. The initial symbol of

the grammar is the symbol $\dot{q}_0$ corresponding to the initial state $q_0$ of the transducer.

The elementary trees of the grammar are constructed as follows. For each rule of the form

$$q\langle[x_0]\rangle(f^{(m)}(x_1,\ldots,x_m)) \doteq \tau$$

we build a tree named $\langle q, f, \tau\rangle$. Where this tree appears is determined solely by the state $q$, so we take the root node of the tree to be the corresponding symbol $\dot{q}$. Any foot node in the tree will also need to be marked with the same label, so we pass this information down as the tree is built inductively. The tree is therefore of the form $\dot{q}(\lceil\tau\rceil_q)$ where the right-hand-side transformation $\lceil\cdot\rceil_q$ constructs the remainder of the tree by the inductive walk of $\tau$, with the subscript noting that the root is labeled by state $q$.

$$\begin{aligned}
\lceil f^{(m)}(t_1,\ldots,t_m)\rceil_q &= f(\lceil t_1\rceil_q,\ldots,\lceil t_m\rceil_q)\\
\lceil q_j\langle\tau\rangle(x_k)\rceil_q &= \boxed{k}\dot{q}_j(\lceil\tau\rceil_q)\\
\lceil q_j\langle\rangle(x_k)\rceil_q &= \boxed{k}\dot{q}_{j\downarrow}\\
\lceil x_0\rceil_q &= \dot{q}_*
\end{aligned}$$

Note that at $x_0$, a foot node is generated of the proper label. (Because the equation is linear, only one foot node is generated, and it is labeled appropriately by construction.) Where recursive processing of the input tree occurs ($q_j\langle\tau\rangle(x_k)$), we generate a tree that admits adjunctions at $\dot{q}_j$. The role of the diacritic $\boxed{k}$ is merely to specify the permutation of operable sites for interpreting derivation trees; it says that the $k$-th child in a derivation tree rooted in the current elementary tree is taken to specify adjunctions at this node.

The trees generated by this TAG correspond to the outputs of the corresponding tree transducer. Because of the more severe constraints on TAG, in particular that all combinatorial limitations on putting subtrees together must be manifest in the labels in the trees themselves, the outputs actually contain more structure than the corresponding transducer output. In particular, the state-labeled nodes are merely for bookkeeping. A simple homomorphism removing these nodes gives

the desired transducer output:[12]

$$\begin{aligned} rem(\dot{q}(x)) &\doteq rem(x) && \text{for } \dot{q} \in \dot{Q} \\ rem(f^{(n)}(x_1, \ldots, x_n)) &\doteq f^{(n)}(rem(x_1), \ldots, rem(x_n)) && \text{for } f^{(n)} \in \mathcal{G}^{(n)} \end{aligned}$$

An example may clarify the construction. Recall the reversal embedded transducer in (8) above. The construction above generates a TAG containing the following trees. We have given them indicative names rather than the cumbersome ones of the form $\langle q_i, f, \tau \rangle$.

$$\begin{aligned} \alpha_{nil}: &\quad \dot{r}(nil) \\ \alpha_1: &\quad \dot{r}(\boxdot \dot{s}(1(nil))) \\ \alpha_2: &\quad \dot{r}(\boxdot \dot{s}(2(nil))) \\ \beta_{nil}: &\quad \dot{s}(\dot{s}_*) \\ \beta_1: &\quad \dot{s}(\boxdot \dot{s}(1(\dot{s}_*))) \\ \beta_2: &\quad \dot{s}(\boxdot \dot{s}(2(\dot{s}_*))) \end{aligned}$$

It is simple to verify that the derivation tree

$$\alpha_1(\beta_2(\beta_2(\beta_{nil})))$$

derives the tree

$$\dot{r}(\dot{s}^4(2(\dot{s}(2(\dot{s}(1(nil))))))) \quad .$$

Simple homomorphisms that extract the input function symbols on the input and drop the bookkeeping states on the output (that is, the homomorphism *rem* provided above) reduce these trees to $1(2(2(nil)))$ and $2(2(1(nil)))$ respectively, just as for the corresponding tree transducer.

### 7.2.1 Equivalence of DLCETT and TAG

We demonstrate that the compilation from DLCETT to TAG generates a grammar with the same language as that of the DLCETT by appeal to the previous result of Section 7.1.2. Consider a DLCETT $T = \langle Q, \mathcal{F}, \mathcal{G}, \Delta, q_0 \rangle$ converted by the compilation above to a grammar $G = \langle \mathcal{G} \cup \dot{Q}, P, \dot{q}_0 \rangle$. That grammar may itself be compiled to a

[12] As noted in Footnote 8, a formalization of a modified form of TAG that directly incorporates state information at operable sites (Büchse *et al.* 2014) eliminates this need for bookkeeping through extra nodes in the tree structure, making the equivalence even stronger.

DLCETT using the compilation of Section 7.1.2, previously shown to be language-preserving. We show that this round-trip conversion preserves the language that is the range of the DLCETT by showing that each equation in the original grammar "round-trip" compiles to an equation that differs only in the tree structure. In particular, a rule of the form $q\langle x_0\rangle(f(x_1,\ldots,x_m)) = \tau$ compiles to the equation $q\langle x_0\rangle(f(x_1,\ldots,x_m)) = \tau'$ where $\tau = rem(\tau')$. We will write $\tau' \approx \tau$ when $\tau = rem(\tau')$.

For each rule in $T$ of the form $q\langle x_0\rangle(f(x_1,\ldots,x_m)) = \tau$, we generate a tree $\langle q, f, \tau\rangle$ in the grammar $G$ of the form $\dot{q}(\lceil\tau\rceil_q)$. This tree, in turn, is compiled as in Section 7.1 to an equation in the output transducer $T'$:

$$\begin{aligned} q\langle x_0\rangle(\langle q,f,\tau\rangle(x_1,\ldots,x_m)) &= \lfloor\dot{q}(\lceil\tau\rceil_q)\rfloor \\ &= \dot{q}(\lfloor\lceil\tau\rceil_q\rfloor) \\ &\approx \lfloor\lceil\tau\rceil_q\rfloor \end{aligned}$$

(Here and in the following, we write $q$ for $q_{|\dot{q}|}$ in the $\lfloor\cdot\rfloor$ construction, taking advantage of the bijection between the $\dot{Q}$ symbols and the corresponding states of the generated transducer.) Note that this is exactly of the required form, so long as $\lfloor\lceil\tau\rceil_q\rfloor \approx \tau$, which we now prove by induction on the structure of $\tau$.

- If $\tau = x_0$, $\lfloor\lceil x_0\rceil_q\rfloor = \lfloor\dot{q}_*\rfloor = x_0$.
- If $\tau = q_j\langle\rangle(x_k)$, $\lfloor\lceil q_j\langle\rangle(x_k)\rceil_q\rfloor = \lfloor\boxed{k}\dot{q}_{j\downarrow}\rfloor = q_j\langle\rangle(x_k)$.
- If $\tau = q_j\langle\tau_0\rangle(x_k)$,

  $$\begin{aligned} \lfloor\lceil q_j\langle\tau_0\rangle(x_k)\rceil_q\rfloor &= \lfloor\boxed{k}\dot{q}_j(\lceil\tau_0\rceil_q)\rfloor \\ &= q_j\langle\lfloor\dot{q}_j(\lceil\tau_0\rceil_q)\rfloor\rangle(x_k) \\ &= q_j\langle\dot{q}_j(\lfloor\lceil\tau_0\rceil_q\rfloor)\rangle(x_k) \\ &\approx q_j\langle\tau_0\rangle(x_k). \end{aligned}$$

  The last step follows from the induction hypothesis and the fact that $rem$ removes the symbol $\dot{q}_j$.
- If $\tau = f^{(m)}(t_1,\ldots,t_m)$,

  $$\begin{aligned} \lfloor\lceil f^{(m)}(t_1,\ldots,t_m)\rceil_q\rfloor &= \lfloor f^{(m)}(\lceil t_1\rceil_q,\ldots,\lceil t_m\rceil_q)\rfloor \\ &= f^{(m)}(\lfloor\lceil t_1\rceil_q\rfloor,\ldots,\lfloor\lceil t_m\rceil_q\rfloor) \\ &\approx f^{(m)}(t_1,\ldots,t_m). \end{aligned}$$

Again, the last step applies the induction hypothesis.

Writing $L(T)$ for the range string language of the transducer $T$, we have that $L(G) = L(T')$ and $L(T) = L(T')$. We conclude that $L(T) = L(G)$. In fact, by the above, the tree languages are identical up to the homomorphism *rem*. Most importantly, then, the weak generative capacity of TAGs and the range of DLCETTs are identical.

### 7.3 *The bimorphism characterization of STAG*

The major advantage of characterizing TAG derivation in terms of tree transducers (via the compilation (10)) is the integration of synchronous TAGs into the bimorphism framework, which follows directly.

In order to model a synchronous grammar formalism as a bimorphism, the well-formed derivations of the synchronous formalism must be characterizable as a regular tree language and the relation between such derivation trees and each of the paired derived trees as a homomorphism of some sort. As shown in Section 6, for synchronous tree-substitution grammars, derivation trees are regular tree languages, and the map from derivation to each of the paired derived trees is a linear complete tree homomorphism. Thus, synchronous tree-substitution grammars fall in the class of bimorphisms $B(LC, LC)$. The other direction holds as well; all bimorphisms in $B(LC, LC)$ define string relations expressible by an STSG.

A similar result follows for STAG. Crucially relying on the result above that the derivation relation is a DLCETT, we can use the same method directly to characterize the synchronous TAG string relations as just $B(ELC, ELC)$. We have thus integrated synchronous TAG with the other transducer and synchronous grammar formalisms falling under the bimorphism umbrella.

## 8 MULTIPLE ADJUNCTION

The discussion so far has assumed that derivations allow at most one operation to occur at any given node in an elementary tree (in fact, exactly one). This constraint inhered in the original formulations of TAG derivation (Vijay-Shanker 1987), and had the effect of removing systematic spurious ambiguities without reducing the range of defin-

able languages. Schabes and Shieber (1994) point out the desirability of allowing multiple adjunctions at a single node, and provide various arguments for this generalization, most notably as needed for many applications of synchronous TAG, which is precisely the case that we are concerned with in this paper. It therefore behooves us to examine the effect of multiple adjunction on the analysis.

There are various ways in which multiple adjunction can be inserted. Most simply, one could specify that the set of operable nodes of a tree allows for a given node in the set a fixed number of times. (This could be graphically depicted by allowing more than one diacritic at a given node, with each diacritic to be used exactly once.) In theory, this would allow multiple nontrivial adjunctions to occur at a single node, inducing ambiguity as to the resulting derived tree, but we can eliminate this possibility by requiring that nontrivial (that is, non-NA) trees be adjoined at at most one site at a given node. We start by handling this kind of simple generalization of TAG derivation in Sections 8.1–8.2.

More generally, Schabes and Shieber (1994) call for allowing an arbitrary number of adjunctions at a given node. In particular, they call for distinguishing predicative and modifier auxiliary trees, and allowing any number of modifier trees and at most one predicative tree to adjoin at a given node. The derived tree is ambiguous as to the relative orderings of the modifier trees, but the predicative tree is required to fall above the modifier trees. We address this major generalization of TAG derivation in Section 8.3.

## 8.1 *Simple multiple adjunction*

We start with a simple generalization of TAG derivation in which operable nodes may be used a fixed number of times. Since the set of operable nodes may now include duplicates, adjunction nodes may occur more than once in the permutation $\pi$. To guarantee that at most one of these can be nontrivially adjoined, we need to revise the definition of derivation tree, that is, fix the tree automaton from Section 4.3 defining well-formed derivation trees, and to prove that the derivation relation $\mathcal{D}$ is still well-defined.

We present an alternative automaton defining the regular tree language of well-formed derivation trees now allowing the limited form of multiple adjunction. We double the number of states from

the previous construction. The states of the automaton are the set $\{q_{N\triangle} \mid N \in \mathcal{F}\} \cup \{q_{N\bullet} \mid N \in \mathcal{F}\}$, two for each unranked vocabulary symbol in the derived tree language. The $\triangle$ diacritic indicates a nontrivial tree rooted in the given symbol; the $\bullet$ diacritic requires a nonadjunction NA tree rooted in that symbol. The start state is $q_{|S|\triangle}$.

For each nontrivial tree (that is, not an NA tree) ${}_{\square}\gamma = \langle \gamma, \pi \rangle$, of arity $n$ and rooted with the symbol $N$, we construct all possible transitions of the form

$$q_{|N|\triangle}(\gamma(x_1, \ldots, x_n)) \doteq \gamma(q_1(x_1), \ldots, q_n(x_n))$$

where each $q_i$ is either $q_{|\gamma@\pi_i|\bullet}$ or $q_{|\gamma@\pi_i|\triangle}$, subject to the constraint that for each node $\eta$ in $\alpha$, the sequence $\langle q_i \mid \pi_i = \eta \rangle$ contains at most one $\triangle$. Because there are many such ways of setting the $q_i$ to satisfy this constraint, there are many (though still a finite number of) transitions for each $\gamma$.

In addition, for NA trees, there is a transition

$$q_{|N|\bullet}(\gamma) \doteq \gamma \qquad .$$

The set of well-formed derivation trees is thus still a regular tree set.

The only remaining issue is to verify that the limited form of multiple adjunction that we allow still yields a well-defined derived tree. In general, multiple adjunctions at the same site do not commute. However, the only cases of multiple adjunctions that we allow involve all but one of the auxiliary trees being vestigial nonadjunction trees. Such cases do commute. It suffices to show that $\gamma[\text{ADJ}_p\ \beta, \text{ADJ}_p\ \text{NA}] = \gamma[\text{ADJ}_p\ \text{NA}, \text{ADJ}_p\ \beta]$; we derive this as follows:

$$\begin{aligned}
\gamma[\text{ADJ}_p\ \beta, \text{ADJ}_p\ \text{NA}] &= \gamma[\text{ADJ}_p\ \beta][\text{ADJ}_{update(p,\beta,p)}\ \text{NA}] \\
&= \gamma[\text{ADJ}_p\ \beta][\text{ADJ}_p\ \text{NA}] \\
&= \gamma[\text{ADJ}_p\ \beta] \\
&= \gamma[\text{ADJ}_p\ \text{NA}][\text{ADJ}_p\ \beta] \\
&= \gamma[\text{ADJ}_p\ \text{NA}][\text{ADJ}_{update(p,\beta,p)}\ \beta] \\
&= \gamma[\text{ADJ}_p\ \text{NA}, \text{ADJ}_p\ \beta]
\end{aligned}$$

### 8.2 *Fixed multiple adjunction*

What if we allow more than one of the multiple (fixed) occurrences of a node to be operated on by a nontrivial auxiliary tree? At that point,

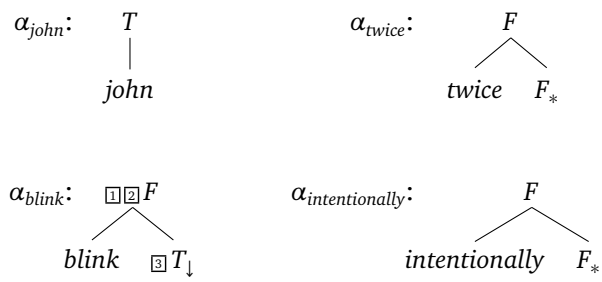



Figure 9:
A fragment with multiple adjunction.

the definition of simultaneous operations no longer commutes, and which auxiliary tree is used at which position becomes important.

The definition of the derivation tree language given in Section 4.3 allows such derivations to be specified merely by relaxing the constraint that a node appears only once in the set of operable sites. If we move to a multiset of operable sites, with $\pi$ a permutation over that multiset, the remaining definitions generalize properly.

We present (Figure 9) a fragment based on the semantic half of a synchronous TAG presented previously (Shieber 1994, Figure 1) to exemplify simultaneous adjunction. This grammar uses simultaneous adjunction at the root of the $\alpha_{blink}$ tree. That tree has three operable sites, two of which are the root node. We will take the permutation of operable sites for the tree to be $\langle \boxed{1}, \boxed{2}, \boxed{3} \rangle$.

We can examine what the compilation of Section 7.1 provides as the interpretation for this grammar. Applying it to the output trees in the grammar generates a DLCETT. We start with the problematic multiple adjunction tree $\alpha_{blink}$.

$$\begin{aligned} q_F\langle\rangle(\alpha_{blink}(x_1,x_2,x_3)) &\doteq \lfloor \alpha_{blink} \rfloor \\ &= \lfloor \boxed{1}\boxed{2}F(blink, \boxed{3}T) \rfloor \\ &= q_F\langle \lfloor \boxed{2}F(blink, \boxed{3}T) \rfloor \rangle(x_1) \\ &= q_F\langle q_F\langle \lfloor F(blink, \boxed{3}T) \rfloor \rangle(x_1) \rangle(x_2) \\ &= q_F\langle q_F\langle F(blink, q_T\langle\rangle(x_3)) \rangle(x_1) \rangle(x_2) \end{aligned}$$

(Here, the second line uses the obvious generalization of the second equation of (10) to sets of diacritics, that is,

$$\lfloor \boxed{k} \cdots f(t_1,\ldots,t_n) \rfloor = q_{|f|}\langle \lfloor \cdots f(t_1,\ldots,t_n) \rfloor \rangle(x_k) \quad ,$$

the ellipses standing in for arbitrary further diacritics.)

The second and third steps are notable here, in that the choice of which of the two operable sites to use first was arbitrary. That is, one could just as well have chosen to process diacritic [2] before [1], in which case the generated rule would have been

$$q_F\langle\rangle(\alpha_{blink}(x_1, x_2, x_3)) \doteq q_F\langle q_F\langle F(blink, q_T\langle\rangle(x_3))\rangle(x_2)\rangle(x_1) \quad .$$

This is, of course, just the consequence of the fact that multiple adjunctions at the same node do not commute. To manifest the ambiguity, we can just generate both transitions (and in general, all such transitions) in the transducer defining the derivation relation. The transducer naturally becomes nondeterministic. Alternatively, a particular order might be stipulated, regaining determinism, but giving up analyses that take advantage of the ambiguity.

Completing the compilation, we generate transitions for the other trees:

$$\begin{aligned} q_T\langle\rangle(\alpha_{john}) &\doteq T(john) \\ q_F\langle x_0\rangle(\beta_{twice}) &\doteq F(twice, x_0) \\ q_F\langle x_0\rangle(\beta_{intentionally}) &\doteq F(intentionally, x_0) \end{aligned}$$

The derivation tree $\alpha_{blink}(\beta_{intentionally}, \beta_{twice}, \alpha_{john})$ then derives trees as follows:

$$\begin{aligned} &q_F\langle\rangle(\alpha_{blink}(\beta_{intentionally}, \beta_{twice}, \alpha_{john})) \\ &\quad\doteq q_F\langle q_F\langle F(blink, q_T\langle\rangle(\alpha_{john}))\rangle(\beta_{intentionally})\rangle(\beta_{twice}) \\ &\quad\doteq q_F\langle q_F\langle F(blink, T(john))\rangle(\beta_{intentionally})\rangle(\beta_{twice}) \\ &\quad\doteq q_F\langle F(intentionally, F(blink, T(john)))\rangle(\beta_{twice}) \\ &\quad\doteq F(twice, F(intentionally, F(blink, T(john)))) \end{aligned}$$

corresponding to the meaning *twice*(*intentionally*(*blink*(*john*))). Alternatively, use of the other nondeterministic alternative transition yields

$$\begin{aligned} &q_F\langle\rangle(\alpha_{blink}(\beta_{intentionally}, \beta_{twice}, \alpha_{john})) \\ &\quad\doteq q_F\langle q_F\langle F(blink, q_T\langle\rangle(\alpha_{john}))\rangle(\beta_{twice})\rangle(\beta_{intentionally}) \\ &\quad\doteq q_F\langle q_F\langle F(blink, T(john))\rangle(\beta_{twice})\rangle(\beta_{intentionally}) \\ &\quad\doteq q_F\langle F(twice, F(blink, T(john)))\rangle(\beta_{intentionally}) \\ &\quad\doteq F(intentionally, F(twice, F(blink, T(john)))) \end{aligned}$$

giving the alternative reading for the sentence.

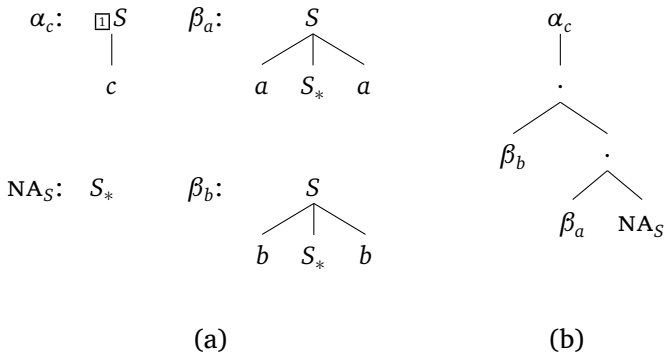



Figure 10:
A grammar (a) for $\{wcw^R \mid w \in \{a,b\}^*\}$ using general multiple adjunction, and (b) a derivation of the string $abba$.

## 8.3 *General multiple adjunction*

Finally, fully general multiple adjunction as described by Schabes and Shieber (1994) allows for one and the same operable site to be used an arbitrary number of times. To enable this interpretation of TAG derivations, major changes need to be made to the definitions of derivation tree and derivation relation.

Consider the sample grammar of Figure 10 where the two auxiliary trees $\beta_a$ and $\beta_b$ are modifier trees (in the terminology of Schabes and Shieber (1994)) and thus allowed to multiply adjoin at the two operable nodes in the initial tree. This grammar should generate the language $\{wcw^R \mid w \in \{a,b\}^*\}$.

Derivation trees must allow an arbitrary number of operations to occur at a given site. To represent this in a ranked tree, we can encode the sequence of trees adjoined at a given location with a recursive structure. In particular, we use a binary symbol · (which we write infix) to build a list of trees to be adjoined at the site, using a nonadjunction tree to mark the end of the list. Essentially, derivation trees now contain lists of auxiliary trees to operate at a site rather than a single tree, with the nonadjoining trees serving as the *nil* elements of the list and the binary · serving as the binary *constructor*. For example, a derivation for the grammar of Figure 10(a) can be represented by the tree in Figure 10(b).

The derivation tree language with lists instead of individual trees is still regular. In fact, the full specification of multiple adjunction given by Schabes and Shieber (1994) specifies that at a given operable site an arbitrary number of modifier trees but at most one predicative

tree may be adjoined. Further, the predicative tree is to appear highest in the derived tree above the adjoined modifiers. This constraint can be specified by defining the derivation tree language appropriately, allowing at most one predicative tree, and placing it at the end of the list of nontrivial trees adjoining at a site. It is a simple exercise to show that the derivation tree language so restricted still falls within the regular tree languages.

Finally, we must provide a definition of the derivation relation for this generalized form of multiple adjunction. In particular, we need transitions for the new form of constructor node, which specifies the combination of two adjunctions at a single site. We handle this by stacking the rest of the adjunctions above the first. We add to the definition of the derivation transducer of Section 7.1 transitions of the following form for each symbol $N$ that is the root of some auxiliary tree:

$$q_N\langle x_0\rangle(x_1 \cdot x_2) \doteq q_N\langle q_N\langle x_0\rangle(x_1)\rangle(x_2)$$

Note that the new transition is still linear and complete.

For the grammar of Figure 10(a) we would thus have the following transitions defining the derivation relation:

$$\begin{aligned}
q_S\langle\rangle(\alpha(x)) &\doteq q_S\langle S(c)\rangle(x)\\
q_S\langle x_0\rangle(\mathrm{NA}_S) &\doteq x_0\\
q_S\langle x_0\rangle(\beta_a) &\doteq S(a, x_0, a)\\
q_S\langle x_0\rangle(\beta_b) &\doteq S(b, x_0, b)\\
q_S\langle x_0\rangle(x_1 \cdot x_2) &\doteq q_S\langle q_S\langle x_0\rangle(x_1)\rangle(x_2)
\end{aligned}$$

Using this derivation relation, the derived tree for the derivation tree of Figure 10(b) can be calculated as

$$\begin{aligned}
q_S\langle\rangle(\alpha_c(\beta_b \cdot \beta_a \cdot \mathrm{NA}_S)) &\doteq q_S\langle S(c)\rangle(\beta_b \cdot \beta_a \cdot \mathrm{NA}_S)\\
&\doteq q_S\langle q_S\langle S(c)\rangle(\beta_b)\rangle(\beta_a \cdot \mathrm{NA}_S)\\
&\doteq q_S\langle S(b, S(c), b)\rangle(\beta_a \cdot \mathrm{NA}_S)\\
&\doteq q_S\langle q_S\langle S(b, S(c), b)\rangle(\beta_a)\rangle(\mathrm{NA}_S)\\
&\doteq q_S\langle S(a, S(b, S(c), b), a)\rangle(\mathrm{NA}_S)\\
&\doteq S(a, S(b, S(c), b), a)
\end{aligned}$$

corresponding to the string $abcba$ as expected.

## 9 CONCLUSION

Synchronous grammars and tree transducers – two approaches to the specification of language relations useful for a variety of formal and computational linguistics modeling of natural languages – are unified by means of the elegant construct of the bimorphism. This convergence synthesizes the approaches and allows a direct comparison among these and other potential systems for describing language relations through other bimorphisms. The examination of additional bimorphism classes may open up further possibilities for useful modeling tools for natural language.

## ACKNOWLEDGEMENTS

This paper has been gestating for a long time. I thank the participants in my course on Transducers at the 2003 European Summer School on Logic, Language, and Information in Vienna, Austria, where some of these ideas were presented, and Mark Dras, Mark Johnson, Uwe Mönnich, Rani Nelken, Rebecca Nesson, James Rogers, and Ken Shan for helpful discussions on the topic of this paper and related topics. The extensive comments of the JLM reviewers were invaluable in improving the paper. This work was supported in part by grant IIS-0329089 from the National Science Foundation.

## REFERENCES

Alfred V. AHO and Jeffrey D. ULLMAN (1969), Syntax Directed Translations and the Pushdown Assembler, *Journal of Computer and System Sciences*, 3(1):37–56, doi:10.1016/S0022-0000(69)80006-1.

Hiyan ALSHAWI, Srinivas BANGALORE, and Shona DOUGLAS (2000), Learning Dependency Translation Models as Collections of Finite State Head Transducers, *Computational Linguistics*, 26(1):45–60, doi:10.1162/089120100561629.

André ARNOLD and Max DAUCHET (1982), Morphismes et bimorphismes d'arbres [Morphisms and bimorphisms of trees], *Theoretical Computer Science*, 20(1):33–93, doi:10.1016/0304-3975(82)90098-6.

Matthias BÜCHSE, Andreas MALETTI, and Heiko VOGLER (2012), Unidirectional Derivation Semantics for Synchronous Tree-Adjoining Grammars, in *Developments in Language Theory*, volume 7410 of *Lecture Notes in Computer Science*, pp. 368–379, Springer, doi:10.1007/978-3-642-31653-1_33.

Matthias BÜCHSE, Heiko VOGLER, and Mark-Jan NEDERHOF (2014), Tree Parsing for Tree-Adjoining Machine Translation, *Journal of Logic and Computation*, 24(2):351–373, doi:10.1093/logcom/exs050.

Hubert COMON, Max DAUCHET, Remi GILLERON, Florent JACQUEMARD, Denis LUGIEZ, Sophie TISON, and Marc TOMMASI (2008), Tree Automata Techniques and Applications, `http://tata.gforge.inria.fr/`, release of November 18, 2008.

Steve DENEEFE and Kevin KNIGHT (2009), Synchronous Tree Adjoining Machine Translation, in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 727–736, Association for Computational Linguistics, Singapore, `http://aclweb.org/anthology/D09-1076`.

Akio FUJIYOSHI and Takumi KASAI (2000), Spinal-Formed Context-Free Tree Grammars, *Theory of Computing Systems*, 33:59–83, doi:10.1007/s002249910004.

Michel GALLEY, Mark HOPKINS, Kevin KNIGHT, and Daniel MARCU (2004), What's In a Translation Rule, in *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pp. 273–280, Association for Computational Linguistics, Boston, Massachusetts, `http://aclweb.org/anthology/N04-1035`.

Jonathan GRAEHL and Kevin KNIGHT (2004), Training Tree Transducers, in *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pp. 105–112, Association for Computational Linguistics, Boston, Massachusetts, `http://aclweb.org/anthology/N04-1014`.

Chung-Hye HAN and Nancy HEDBERG (2008), Syntax and Semantics of It-Clefts: A Tree Adjoining Grammar Analysis, *Journal of Semantics*, 25:345–380, doi:10.1093/jos/ffn007.

Aravind JOSHI and Yves SCHABES (1997), Tree-Adjoining Grammars, in G. ROZENBERG and A. SALOMAA, editors, *Handbook of Formal Languages*, volume 3, pp. 69–124, Springer, Berlin.

Alexander KOLLER and Marco KUHLMANN (2011), A Generalized View on Parsing and Translation, in *Proceedings of the 12th International Conference on Parsing Technologies*, IWPT '11, pp. 2–13, Association for Computational Linguistics, Stroudsburg, PA, USA, ISBN 978-1-932432-04-6, `http://dl.acm.org/citation.cfm?id=2206329.2206331`.

Philip M. LEWIS II and Richard E. STEARNS (1968), Syntax-Directed Transduction, *Journal of the Association for Computing Machinery*, 15(3):465–488, ISSN 0004-5411, doi:10.1145/321466.321477.

Andreas MALETTI (2008), Compositions of Extended Top-down Tree Transducers, *Information and Computation*, 206(9-10):1187–1196, doi:10.1016/j.ic.2008.03.019.

Andreas MALETTI, Jonathan GRAEHL, Mark HOPKINS, and Kevin KNIGHT (2009), The power of extended top-down tree transducers, *SIAM Journal on Computing*, 39:410–430, doi:10.1137/070699160.

I. Dan MELAMED (2003), Multitext Grammars and Synchronous Parsers, in *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 79–86, Association for Computational Linguistics, Edmonton, Canada, doi:10.3115/1073445.1073466.

I. Dan MELAMED (2004), Statistical Machine Translation by Parsing, in *Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics*, pp. 653–660, Association for Computational Linguistics, Barcelona, Spain, doi:10.3115/1218955.1219038.

Mark-Jan NEDERHOF and Heiko VOGLER (2012), Synchronous Context-Free Tree Grammars, in *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pp. 55–63, Paris, France.

Rebecca NESSON and Stuart M. SHIEBER (2006), Simpler TAG Semantics Through Synchronization, in *Proceedings of the 11th Conference on Formal Grammar*, pp. 129–142, Center for the Study of Language and Information, Malaga, Spain, `http://nrs.harvard.edu/urn-3:HUL.InstRepos:2252595`.

Rebecca NESSON, Stuart M. SHIEBER, and Alexander RUSH (2006), Induction of Probabilistic Synchronous Tree-Insertion Grammars for Machine Translation, in *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA 2006)*, pp. 128–137, Association for Machine Translation in the Americas, Cambridge, Massachusetts, `http://nrs.harvard.edu/urn-3:HUL.InstRepos:2261232`.

William C. ROUNDS (1970), Mappings and Grammars on Trees, *Mathematical Systems Theory*, 4(3):257–287, doi:10.1007/BF01695769.

Yves SCHABES and Stuart M. SHIEBER (1994), An Alternative Conception of Tree-Adjoining Derivation, *Computational Linguistics*, 20(1):91–124, `http://aclweb.org/anthology/J94-1004`.

Yves SCHABES and K. VIJAY-SHANKER (1990), Deterministic Left to Right Parsing of Tree Adjoining Languages, in *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pp. 276–283, Association for Computational Linguistics, Pittsburgh, Pennsylvania, doi:10.3115/981823.981858.

Stuart M. SHIEBER (1994), Restricting the Weak-Generative Capacity of Synchronous Tree-Adjoining Grammars, *Computational Intelligence*, 10(4):371–385, doi:10.1111/j.1467-8640.1994.tb00003.x.

Stuart M. SHIEBER (2004), Synchronous Grammars as Tree Transducers, in *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*, pp. 88–95, Vancouver, Canada, http://nrs.harvard.edu/urn-3:HUL.InstRepos:2019322.

Stuart M. SHIEBER (2006), Unifying Synchronous Tree-Adjoining Grammars and Tree Transducers via Bimorphisms, in *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pp. 377–384, European Chapter of the Association for Computational Linguistics, Trento, Italy, http://nrs.harvard.edu/urn-3:HUL.InstRepos:2252609.

Stuart M. SHIEBER and Yves SCHABES (1990), Synchronous Tree-Adjoining Grammars, in *Proceedings of the 13th International Conference on Computational Linguistics*, volume 3, pp. 253–258, International Committee on Computational Linguistics, Helsinki, Finland, doi:10.3115/991146.991191.

K. VIJAY-SHANKER (1987), *A Study of Tree Adjoining Grammars*, Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, Pennsylvania, http://repository.upenn.edu/dissertations/AAI8804974/.

Dekai WU (1996), A Polynomial-Time Algorithm for Statistical Machine Translation, in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pp. 152–158, Association for Computational Linguistics, Santa Cruz, California, doi:10.3115/981863.981884.

Dekai WU (1997), Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora, *Computational Linguistics*, 23(3):377–404, http://aclweb.org/anthology/J97-3002.

Elif YAMANGIL and Stuart M. SHIEBER (2010), Bayesian Synchronous Tree-Substitution Grammar Induction and Its Application to Sentence Compression, in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 937–947, Association for Computational Linguistics, Uppsala, Sweden, http://nrs.harvard.edu/urn-3:HUL.InstRepos:4733833.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*
http://creativecommons.org/licenses/by/3.0/


# LFG parse disambiguation for Wolof

*Cheikh M. Bamba Dione*
University of Bergen

## ABSTRACT

This paper presents several techniques for managing ambiguity in LFG parsing of Wolof, a less-resourced Niger-Congo language. Ambiguity is pervasive in Wolof and This raises a number of theoretical and practical issues for managing ambiguity associated with different objectives. From a theoretical perspective, the main aim is to design a large-scale grammar for Wolof that is able to make linguistically motivated disambiguation decisions, and to find appropriate ways of controlling ambiguity at important interface representations. The practical aim is to develop disambiguation strategies to improve the performance of the grammar in terms of efficiency, robustness and coverage.

To achieve these goals, different avenues are explored to manage ambiguity in the Wolof grammar, including the formal encoding of noun class indeterminacy, lexical specifications, the use of Constraint Grammar models (Karlsson 1990) for morphological disambiguation, the application of the c-structure pruning mechanism (Cahill *et al.* 2007, 2008; Crouch *et al.* 2013), and the use of optimality marks for preferences (Frank *et al.* 1998, 2001). The parsing system is further controlled by packing ambiguities. In addition, discriminant-based techniques for parse disambiguation (Rosén *et al.* 2007) are applied for treebanking purposes.

*Keywords: LFG, computational grammar, Constraint Grammar, c-structure pruning, discriminant-based disambiguation, Wolof, underspecification, optimality marks*

## 1 INTRODUCTION

This paper deals with the ambiguity problem in the process of analyzing texts in Wolof, a less-resourced language.[1] Specifically, it reports on several techniques used to manage ambiguity in a broad-coverage computational grammar and parser for Wolof. The grammar is implemented in the linguistic framework of Lexical Functional Grammar (LFG) (Kaplan and Bresnan 1982) using the Xerox Linguistic Environment (XLE) (Crouch *et al.* 2013).[2] In LFG, traditional analyses focus on two levels of syntactic representation (Kaplan and Bresnan 1982): Constituent structure (c-structure) models the surface exponence of syntactic information (e.g., word order, dominance and phrasal groupings), and functional structure (f-structure) represents grammatical functions like subject and object.

Wolof, like most natural languages, has pervasive ambiguity, that is, a word or sentence can be analyzed in more than one way. The language is rich in ambiguities of many kinds, including morphological, lexical, syntactic and semantic ambiguities. The ambiguity phenomenon is perhaps the most serious problem faced by natural language processing (NLP) systems, and this is true for many reasons. First, ambiguity typically pertains to all levels of sentence analysis. As MacDonald *et al.* (1994) noted, theoretically, linguistic information can be ambiguous at any given point in a sentence. Furthermore, many sentences that do not seem ambiguous to humans, due to their extensive world knowledge, may present ambiguities to automatic parsers (and to other NLP systems as well in general). Accordingly, large-scale, linguistically motivated grammars tend to be massively ambiguous. Ambiguities can arise, for example, via alternative definitions of morphological and lexical entries, from syntactic or semantic ambiguities, and the interaction of the different ambiguities.

Second, ambiguity typically increases the range of possible interpretations of natural language, and a parser has to find a way to

[1] Wolof is a member of the Senegambian branch of the Niger-Congo language family mainly spoken in Senegal, Gambia and Mauritania. Some Wolof speakers can also be found in Guinea, Guinea-Bissau, Mali and France (see `http://www.ethnologue.com/language/WOL`).

[2] See Section 4 for a brief description of the implementation of the Wolof LFG grammar.

deal with this. It also increases the search space, therefore leading to a combinatorial explosion, which results from multiplying up each individual ambiguity. For instance, for a ten word sentence in which each word could have three interpretations, there are 59,049 possible interpretations for the whole sentence. The situation is exacerbated by the interaction of independent ambiguities. Due to syntactic, semantic and pragmatic ambiguities, the actual number of possible interpretations will be huge.[3] To attempt to resolve all these interpretations becomes hardly possible in a reasonable time.

In this work, the concern for ambiguity management stems both from theoretical and practical requirements and goals. From a theoretical point of view, an important purpose of this work is to develop a parsing system for Wolof which is able to disambiguate (when necessary) the input text in order to ensure correct analysis of the language. In other words, given a string and a context, the aim is to have a system that is able to distinguish the intended reading from the implausible one, but also to preserve linguistically appropriate ambiguities. For an NLP system, this kind of disambiguation decision is particularly relevant, as has been emphasised by Manning and Schütze (1999, pp. 17-18):

> An NLP system needs to determine something of the structure of text – normally at least enough that it can answer "Who did what to whom?" Conventional parsing systems try to answer this question only in terms of possible structures that could be deemed grammatical for some choice of words of a certain category. [...] Therefore, a practical NLP system must be good at making disambiguation decisions of word sense, word category, syntactic structure, and semantic scope.

A secondary, but no less important objective is to apply methods for ambiguity management with the aim to gain efficiency, while maintaining parsing accuracy. Thus, following previous work done in creating language resources and tools for Wolof (Dione 2012b, 2013a), the present research discusses the avenues explored to improve the efficiency and performance of the parser (i.e., to speed up the grammar

[3] For instance, according to Manning and Schütze (1999), Martin *et al.* (1987) report their system giving 455 parses for the sentence *List the sales of the products produced in 1973 with the products produced in 1972*.

development activity and to increase parsing robustness and coverage) by managing ambiguity. The applied methods aim to avoid or minimise the combinatorial explosion that results from ambiguity, as well as to facilitate maintainability of a large code base.

This work addresses several research questions with respect to dealing with ambiguity in linguistically motivated grammar development projects. The first question is how to choose an approach to ambiguity. Managing ambiguity often takes the form of a binary decision: either eliminate or preserve the ambiguity. While the former is the most obvious approach, it is not always feasible or desirable. For instance, handling ambiguity caused by prepositional phrase (PP) attachment may require context and linguistic intuition. For example, in the sentence *she opens the door with the key*, the key is more likely perceived as an instrument used to open the door, rather than it being a feature of the door. Nevertheless, as Chantree (2004, pp. 2) pointed out, "the decision of whether to disambiguate this sentence or not might depend upon the users' proficiency with English and the context provided by the surrounding text."

Conversely, an obvious approach to ambiguity-preserving parsing is to provide the grammatical descriptions or constraints to generate all possible readings. This approach may be problematic in that the number of potential readings might grow exponentially with the length of the sentence. Thus, even though it is clear that "some ambiguities can safely be left in the text", the question still remains over "which ones can be left and which ones must be removed" (Chantree 2004, pp. 1). In linguistically motivated grammar development projects, there is this split between providing the grammatical descriptions to generate all possible readings on the one hand, and the selection of the appropriate one in a given context on the other hand. This paper is adding the latter view to the Wolof project, building on prior work from the LFG framework and other approaches.

A second important question is when and how to attack ambiguity. Ambiguity management can occur before, during or after parsing. If an ambiguity is dealt with early, there is the possibility of losing the right analysis. If it is dealt with late, there is the computational cost of processing many analyses. As Copperman and Segond (1996, pp. 8) pointed out, "the proper balance point in this tradeoff varies for different types of ambiguities, and there is no universal metric".

Concerning the methodology, there exist in the literature a wide range of advanced techniques that can be applied to tackle ambiguity issues, including statistical and non-statistical ones. However, when dealing with ambiguity in languages like Wolof, there is a restriction on the use of certain disambiguation methods. Due to the lack of resources, there is a very limited possibility to apply statistical approaches that often require a large data set to ensure reliable results.

To address these different research questions and to decide among the alternative ways of managing ambiguity, this work is based on three main premises. First, ambiguities are divided into different categories. This is essential to better distinguish ambiguities that are so liable to misinterpretation or to computational complexity that they should be removed from those which can be allowed to remain. A second important point is to manage ambiguity at various levels of description. As Attia (2008, pp. 4) pointed out, it seems to be a good idea "to deal with ambiguity not as one big problem, but rather as a number of divisible problems spreading over different levels of the sentence analysis: pre-parsing, parsing and post-parsing stages." Finally, the third consideration is to manage ambiguity in a systematic way using approaches that can be applied to languages having a lack-of-data problem.

In this work, disambiguation is divided into three stages: pre-parsing, parsing and post-parsing. Disambiguation at the pre-parsing stage focuses on discarding some morphological analyses that are implausible with respect to a given context. The parsing phase covers the topics of the formal encoding of noun class indeterminacy via underspecification, the application of syntactic constraints, lexical specifications and the use of a probabilistic context-free grammar. The post-parsing stage includes the application of preference marks for ranking analyses, and the use of grammar engineering tools for packing ambiguities. In addition, the post-parsing stage involves manually selecting parse solutions using discriminants (Rosén *et al.* 2007). Discriminant-based disambiguation is employed as a timesaving method for constructing a treebank for the language by automatically parsing a corpus with the Wolof LFG grammar. One of the motivations for building the treebank is to create a gold standard test set for Wolof that can be used to evaluate the parser as well as the effect of the other disambiguation methods, e.g., the use of optimality marks for prefer-

ences (Frank *et al.* 1998, 2001) and statistical disambiguation. Because quality controlled treebanks that can serve as gold standards cannot be constructed without considerable manual effort towards ambiguity resolution (Rosén *et al.* 2007), discriminant-based disambiguation is used as an intelligent way of minimizing these efforts. The aim is to optimize the efficiency of manual disambiguation, as inspecting full analyses proved to be a tedious and time-consuming task.

I will attempt to show how the application of the different disambiguation techniques discussed in this paper helps to manage ambiguity and to reduce parse time in the process of analyzing texts in Wolof. However, note that the various disambiguation methods are applied on different parsing levels and parser versions, and thus have interactions that are very difficult to control systematically. Also, note that the purpose is not to give an exhaustive account of all the disambiguation methods used within this research work or to provide an exhaustive overview of their systematic interaction but to illustrate ambiguity management in LFG parsing of Wolof focusing on some example constructions which present particular challenges for grammar development and treebanking work for the language.

This paper is structured as follows. Section 2 provides a general description of ambiguity in natural languages and a common categorization of the different ambiguity types. Section 3 presents evidence that Wolof is massively ambiguous, particularly with respect to morphological, lexical and syntactic ambiguities. Section 4 briefly presents background information on the Wolof grammar relevant for the discussion of ambiguity management in subsequent sections. Section 5 discusses techniques for handling morpho-lexical and syntactic ambiguities, including the formal encoding of noun class indeterminacy, lexical specifications, morphological and lexical disambiguation based on Constraint Grammar (CG) (Karlsson 1990). Section 6 presents some approaches to syntactic ambiguity used for Wolof, including c-structure pruning (Cahill *et al.* 2007, 2008; Crouch *et al.* 2013) and optimality marks (Frank *et al.* 2001). Section 7 presents grammar engineering tools for packing ambiguity in XLE and discusses disambiguation strategies used to increase parsing efficiency by removing spurious ambiguities. Section 8 describes discriminant-based disambiguation techniques to LFG grammars (Rosén *et al.* 2007). A conclusion is given in Section 9.

## 2 AMBIGUITY CATEGORIZATION

Research on ambiguity typically distinguishes between the scope (global vs. local) (Gazdar and Mellish 1989) and types of ambiguity (Gómez 1996).

### 2.1 *Scope: global vs. local*

Global ambiguity means that an entire word string has more than one structure associated with it, as in (1).

(1) Flying planes made her duck. (Gómez 1996, pp. 16)

The sentence in (1) has various readings, including the two following ones: (i) the airplanes made her change her position; (ii) the act of piloting made her change her position. In terms of LFG/XLE, global ambiguities give rise to different whole-sentence f-structures. In general, global ambiguities are linguistically appropriate, and therefore may need to be preserved: Their resolution typically requires semantic and/or pragmatic knowledge.

In contrast, the sentence in (2) from Gómez (1996, pp. 16) involves a local ambiguity, because some subparts of the whole string have different readings. Readers who process this sentence and focus on the last three words, might settle on the existence of a sentential subconstituent made up of *SGEL sold Xerox*.

(2) The company that bought SGEL sold Xerox.

In contrast to global ambiguity resolution, local ambiguity can sometimes be resolved by syntactic analysis. From that perspective, local ambiguity includes the following ambiguities discussed in Section (2.2): lexical, morphological, and syntactic ambiguities that are resolved when a larger sentential context is taken into account.

### 2.2 *Types of ambiguity*

The causes for obtaining different analyses for an input string (a word or a sentence) might be diverse, including lexical, morphological, syntactic and referential, and the interaction of all these levels. This work will concentrate on these aforementioned ambiguity types.

In lexical ambiguity, a given word may be assigned to more than one grammatical category or part of speech (POS) according to the context. For instance, the English word *bank* could be a noun or

verb. Morphological ambiguity typically refers to ambiguity within the same syntactic category, that is, ambiguity of different forms of one lexeme within the same POS. For instance, in the sentence *I saw her run to the bank*, the word *bank* is unambiguously a noun, however, it is still unclear whether it refers to a financial institution or to a river side. This phenomenon is also known as word sense ambiguity. Morpholexical ambiguity is not a uniform phenomenon, but a phenomenon that distinguishes between homonymy and polysemy (Klepousniotou 2002). In theoretical linguistics, the etymological derivation of words and the 'relatedness/unrelatedness' of meaning – a matter of degree that relies on native speaker's feeling – have been proposed for the distinction between homonymy and polysemy. In homonymy, a lexical item accidentally carries two (or more) distinct and unrelated meanings, while in polysemy, a single lexical item has several different but related senses.

Syntactic ambiguity can be divided into structural and functional ambiguity. A sentence is viewed as structurally ambiguous if it can be interpreted or represented by more than one syntactic structure. Attachment of adjuncts (e.g., PP attachment and adjective attachment) represents a canonical case of structural ambiguity. An instance of PP attachment in Wolof is given in (3).[4]

(3) *Góor g-i séen xale b-i ci saxaar g-i.*
man cl-DFP see child cl-DFP in train cl-DFP
"The man saw the child in the train."

The ambiguity here arises from the fact that the grammar provides several sources for the PP. The attachment of the PP *in the train* is syntactically permissible both to the noun phrase (NP) *the child* and the verb *saw*. In general, attachment of adjuncts results semantically in scope ambiguity. The outcome of the attachment depends mainly on two factors: (i) which subcategorization frame the verb prefers and (ii) which attachment is semantically more plausible. In LFG, this ambi-

[4] Abbreviations in the glosses: ADV: adverb; cl: noun class marker; COMP: complementizer; CONJ: conjunction; COP: copula; DET: determiner; DFP: definite proximal; DFD: definite distal; +F: finite; GEN: genitive; INF: infinitive; NDF: indefinite article; NEG: negation; NSFOC: non-subject focus; PREP: preposition; PST: past tense; pl: plural; Rel: relative; S: subject; SFOC: subject focus; sg: singular; VFOC: verb focus; 1, 2, 3: first, second, third person.

guity is reflected both in the c-structure and in the f-structure (adjunct attachment). Adjunct attachment is notoriously difficult: The syntax has no way to determine the attachment, even if humans can.

In contrast, functional ambiguity is semantic without necessarily involving phrase structure distinctions. In LFG, this refers to ambiguity within the f-structure. A typical example is when a constituent can bear both an oblique argument and an adjunct function within the functional structure (see Section 3.1.4).

Referential ambiguity arises, when more than one object is being referred to by a noun phrase or a deictic expression. This is typically the case when readers or listeners are unable to select a unique referent for a linguistic expression out of multiple candidates. For instance, in the sentence *After **they** finished the exam, the students and lecturers left.*, the pronoun *they* is ambiguous: It can refer to *students* only, to *lecturers* only, or to both. One aspect I will point out in the discussion of referential ambiguity is unclear reference of pronominal subjects in some constructions in Wolof (see Section 3.3).

Syntactically legitimate ambiguities contrast with so called spurious ambiguities, which constitute a purely engineering problem. Spurious ambiguities can refer to duplicated solutions – the same full-sentence f-structure associated with different c-structures or processing sequences – or incorrect f-structures, that is, a reading of the sentence that a native speaker would not attest to. This work will focus on the former definition. Spurious ambiguities mainly refer here to multiple parse solutions that are completely identical (Komagata 2004), for example, when many different derivations or trees generate the same structure. As such, this ambiguity type poses serious grammar engineering issues in terms of efficiency, and therefore needs to be removed.

Having discussed the main ambiguity types the present work will deal with, I will now turn to some ambiguity issues in Wolof that present a particular challenge in the context of grammar implementation.

## 3 AMBIGUITY IS PERVASIVE IN WOLOF

### 3.1 *Morphological and lexical ambiguity*

As noted above, ambiguities can arise from linguistically justified lexical and morphological ambiguities. Morpholexical ambiguity in the Wolof grammar arises mainly from polysemy and homonymy caused by Wolof noun classes (NC). Conversely, lexical ambiguity stems from different sources, including ideophones acting as verb collocations, words with several parts of speech and verbs with various subcategorization frames. These issues are discussed in the next sections.

#### 3.1.1 Ambiguity due to Wolof noun classes

As is typical for Atlantic languages (Sapir 1971), Wolof is a noun class language with noun class agreement (McLaughlin 2010; Torrence 2005). The language has 13 noun classes identified by their index:[5] 8 singular (*b, g, j, k, l, m, s, w*), 2 plural (*y, ñ*), 2 locative (*f, c*), and 1 manner (*n*). Of the singular noun classes, the *s* class also functions as a diminutive class. As for plural noun classes, *y* is the class of most nouns, while *ñ* is the class of a restricted small set of human nouns. Accordingly, a noun may belong to as many as three classes (McLaughlin 2010): a singular, a plural and a diminutive singular class.

Unlike the noun class system found in Bantu languages, nouns in Wolof lack a class marker on the noun itself. Instead, class membership is marked on noun specifiers such as determiners (definite and indefinite articles, demonstratives) or quantifiers, on relative pronouns, etc. For instance, Wolof possesses two definite and two indefinite articles, all agreeing in class with the noun. Indefinite and definite determiner phrases (DPs) have a different word order, as shown in (4-6). While the definite article obligatorily follows the NP, the indefinite article obligatorily precedes the NP. The vowel suffixes *i* and *a* on the definite articles respectively encode proximity and distance in space, time, or conversation. In contrast, the vowel prefix *a* marks indefiniteness.

[5] The noun class index functions as a stem to which a determiner/pronoun/etc. affix is added. In this paper, the stem is glossed *cl*.

(4) *a-b xale*
NDF-cl child
"A child"

(5) *xale b-i*
child cl-DFP
"The child (here)"

(6) *xale b-a*
child cl-DFD
"The child (there)"

Although eight singular classes and two plural classes can clearly be distinguished, the morphological paradigms of the noun class system are characterised by noun class syncretism, that is, a single morphological form corresponds to two or more morphosyntactic descriptions (Baerman *et al.* 2005). For example, due to homonymy/polysemy, the word form *ndaw* in (7) corresponds to different noun classes, as marked on the definite articles. The noun surfaces in the same form both in the singular and plural noun class.

(7)

| Number | Noun Class | Example |
|---|---|---|
| Singular | *g* class | *ndaw* ***g-i*** |
| | *s* class | *ndaw* ***s-i*** |
| | *l* class | *ndaw* ***l-i*** |
| Plural | *ñ* class | *ndaw* ***ñ-i*** |
| | *y* class | *ndaw* ***y-i*** |

The paradigm in (7) shows that some Wolof nouns like *ndaw* have many readings at the word level, thereby increasing ambiguity in the grammar. The examples in (8) illustrate sentences in which the same form *ndaw* occurs with the noun class *g* in (8a), *s* in (8b), *l* in (8c), *ñ* in (8d) and *y* in (8e).

(8) a. *A-**g** ndaw gàddaay na sama jëmm j-ii.*
NDF-cl youth leave +F.3sg POSS1SG face cl-DEM
"I do not look young anymore."

b. *Ta amaana kon, di-na jël ndaw **s**-i.*
CONJ perhaps then, IPF-+F.3sg take woman cl-DFP
"And he would then possibly marry the woman."

c. *Ndaw **l**-i ñëw na.*
messenger cl-DFP arrive +F.3sg
"The messenger has arrived."

d. *Ma xool ndaw **ñ**-i.*
1sg look.at young cl-DFP
"So, I look at the young people."

e. *Nu doon ndaw **y**-u gëm l-a nu-y wut.*
1pl COP.PST young cl-Rel believe cl-Rel 1pl-IPF look.for
"We were young people who believed in what we were doing."

Likewise, the word form *mag* can occur with at least four noun classes (*j*, *m*, *ñ*, and *y*): for example, *mag* ***j****-i* 'the brother', *mag* ***m****-i* 'the old man' *mag* ***ñ****-i* 'the old people', and *a-****y*** *mag* 'some old people'. Accordingly, in the Wolof grammar, the nominal coordination in (9) has at least 20 readings that result from the ambiguous forms of the two conjuncts.

(9) Mag ak ndaw
old CONJ young
"Old and young people"

### 3.1.2 Co-verbs using *ne/ni*

In the Wolof grammar, lexical ambiguity arises from ideophonic expressions. *Ideophone* is a common term for expressive vocabulary found in languages in Africa, Eurasia, and Australia. Doke (1935, pp. 118) defines an ideophone as "a vivid representation of an idea in sound" or "a word, often onomatopoeic, which describes a predicate, qualificative or adverb in respect to manner, colour, sound, smell, action, state or intensity".

In morphophonological and syntactic terms, ideophones represent onomatopoeic or synesthetic expressions which tend to have an emotive function and exhibit specific syntactic, morphological, and/or phonological properties that make them a distinct group (Voeltz and Kilian-Hatz 2001). In addition, ideophones are associated with spoken and dramatic registers of speech. Accordingly, a common distributional feature of ideophones is that they tend to occur in collocations with a restricted set of generic verbs such as 'do', 'say', or 'go' (Creissels 2001). Ideophones seem to be well documented, but little work has been done on their implementation in computational grammars.

In Wolof, ideophones "can either accompany a verb as an intensifier and are thus known as coverbal ideophones, or they can be used in quotative constructions with the verb *ne* 'say'" (McLaughlin 2004, pp. 256), as in the examples in (10) and (11).

(10) *Sa mbubb dafa set wecc.*
2sg:POSS gown 3sg:VFOC ADJ:clean IDEO
"Your gown is perfectly clean." (McLaughlin 2004, pp. 256)

(11) *Mu ne tekk.*
3sg say IDEO:of saying
"S/He was quiet."

The use of coverbal ideophones increases the ambiguity of collocational verbs like *ne*, which belongs to the items with the most notorious hotspots of ambiguity. It can additionally be a comparative preposition (12), a complementizer (13), a regular verb without coverbal ideophones (14) and a copular verb (15). Accordingly, a special treatment of ideophones was necessary to limit this ambiguity.

(12) *Mu mel ne xale.*
3sg look like child
"S/He looks like a child."

(13) *Mu xam ne dem na.*
3sg know COMP go +F.3sg
"S/He knows that s/he has left."

(14) *Mu ne leen ñu dem.*
3sg tell 3sg/O 3pl go
"S/He told them to go."

(15) *Mu ne ci kër gi.*
3sg COP prep house cl.DFP
"S/He was in the house."

### 3.1.3 Lexical ambiguity: POS

As the collocational verb *ne* discussed in the previous section illustrates, in Wolof (like in many languages), most words can have several parts of speech. This includes lexical items that can belong to different word classes such as determiners, bound and free relative pronouns, complementizers, etc. In particular, short tokens like *la* are multiply ambiguous, making it evident that lexical ambiguity is extremely widespread in Wolof. This item can have both a verbal and a non-verbal reading, as shown in (16) from Dione (2014). In this example, *la* can be a non-subject focus morpheme (INFL) (16a), a copular verb (16b), a clitic object (16c), a determiner or a bound pronoun (16d), a free relative pronoun (16e) or a complementizer (16f).

(16) a. *Fas **la** gis.* *INFL*
horse.w-cl 3sg.FOC see
'It is the horse that he saw.'

b. *Fas **la**.* *Non-subject copula*
horse.w-cl COP.3sg
'It is a horse.'

c. *Gis-u-ma **la**.* *Clitic object*
see-NEG-1sg 2sgO
'I haven't seen you.'

d. *Ngelaw* ***la*** *agsi* *Determiner/Rel. Pron.*
wind.l-cl l-cl.det/REL arrive
'The wind came around / which came around.'

e. ***la*** *mu gis-oon ...* *Free relative*
free.REL he see-PST
'What he saw ...'

f. ***la*** *mu doon ngelaw lépp ...* *Complementizer*
COMP 3sg ipf.PST be.windy quant
'Despite the fact that it was windy ...'

In the grammar, assigning so many parts of speech to the same word form (e.g., to the lexical entry *la*) poses both ambiguity and efficiency problems.

### 3.1.4 Lexical ambiguity: subcategorization frames

As with grammatical categories, words have often more than one subcategorization frame. In English, the verb *break* may have a transitive and an intransitive reading (e.g., *I broke it* vs. *It broke*). Likewise, the verb *want* may have bare transitive reading (I want something) or a transitive with infinitive reading (I want it to leave). Similarly, the Wolof verbs can have several subcategorization frames; for example, the verb *dugg* 'enter' in (17) has at least two subcategorization frames: It may have a bare intransitive and an oblique reading.

(17) *Mu dugg ci kër gi.*
3sg enter in house cl.DFP
"S/He entered the house."

In (17), a lexical and functional ambiguity problem arises caused by the semantics associated with the PP *ci kër gi* "in the house". This ambiguity does not involve structural distinctions, since the constituent is clearly a PP that attaches to the verb *dugg*. The question is: Which grammatical function does this PP bear within the verbal phrase (VP)? Is it an argument or an adjunct of the verb?

On the one hand, one might assume that the PP is subcategorized for by the verb. In particular, that amounts to considering the PP as an instance of oblique arguments, that is, "nonsubject arguments which are not of the appropriate morphosyntactic form to be objects and which do not undergo syntactic processes which affect objects" (Butt *et al.* 1999, pp. 50). On the other hand, the PP may be analyzed as an

adjunct, that is, as an optional constituent of the verb that, when removed, will not affect the remainder of the sentence except to discard from it some auxiliary information. As such, the PP is seen as a modifying phrase that depends on the VP, bearing an adverbial function within the latter phrase.[6]

### 3.2 *Syntactic ambiguity*

In Wolof, ambiguous lexical forms are also a source of syntactic ambiguity; but, even without lexical ambiguity, there are legitimate syntactic ambiguities such as PP attachment and coordination ambiguity. One might want to constrain these to legitimate cases and make sure they are processed efficiently. Some syntactic ambiguity issues in Wolof are discussed in the next sections.

#### 3.2.1 Structural ambiguity

Structural ambiguity occurs when the arrangement of words in a grammatical structure permits two or more meanings to emerge, as is the case with PP attachment discussed above. Structural ambiguity can also be caused by an interaction of lexically ambiguous forms and syntactic ambiguity, as illustrated in (18). For example, *bi* can be a determiner, a bound or a free relative pronoun or a complementizer; *moom* can either be a verb, a strong pronoun or a topic adverb; *doon* can be a copula or a past progressive auxiliary, etc. Three possible interpretations of this sentence are shown in the translations in (18).

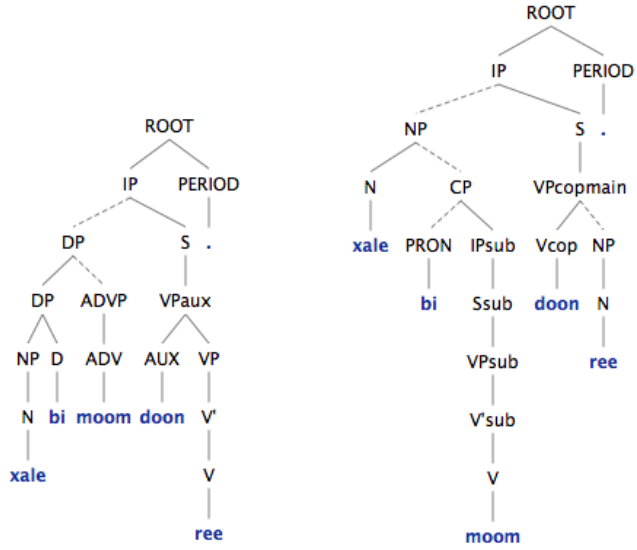
(18) Xale b-i moom doon ree.
child cl-DFP adv.TOP IPF.PST laugh
child cl-Rel own COP laugh
child cl-DFP own IPF.PST laugh

"As for the child, (s)he was laughing."
"The child who owns (something) becomes a laugh."
"The child owns (something) and was laughing."

Before disambiguation, the sentence in (18) has more than 100 c-structure trees that are valid with respect to the grammar. The c-structures for the two first interpretations given in (18) are represented in Figure 1.

[6] For the distinction between arguments and modifiers (in particular between oblique and adjunct functions) and the several tests conducted to illuminate this

Figure 1: Two possible c-structures for *Xale bi moom doon ree*




The third reading arises from coordination without an explicit conjunction. Conjuncts in a coordinate structure can be joined by an overt conjunction (syndetic coordination) or not (asyndetic coordination) (McShane 2005). Like many languages, Wolof permits coordinate structure without an overt conjunction (see Section 6.4).

### 3.3 *Referential ambiguity: pro-drop and impersonal passive*

In Wolof, an example of referential ambiguity with a global scope arises from pro-drop (Chomsky 1981; Baptista 1995)and Wolof impersonal passive constructions (19).

(19) a. *Góor ñi-i gor na-ñu garab g-i.*
man cl-DFP cut.down +F-3pl tree cl-DFP
"The men cut down the tree."

b. *Gor na-ñu garab g-i.*
cut.down +F-3pl tree cl-DFP
"They cut down the tree."
"The tree was cut down."

distinction, see for example, (Dalrymple 2001).

Example (19) illustrates the pro-drop nature of the language. The sentence in (19a) is similar to the one in (19b), except that in the latter example the overt subject is missing; nevertheless both sentences are grammatical. In (19b), there is no overt subject, because Wolof freely allows the omission of such an argument.

Sentences with a third plural subject like (19b) are ambiguous because they can express both a pro-drop or an impersonal passive reading. Because Wolof lacks a true passive derivation (Voisin-Nouguier 2002), it often uses an active sentence with an impersonal third plural subject to express the passive idea (Torrence 2005). The two different readings of this sentence are reflected in the translations. The ambiguity here is due to the interpretations of the third plural element (also called subject marker) *nañu*. On the one hand, this element can be a referential subject, in which case it is understood to refer to a specific group of individuals who cut down the tree.[7] On the other hand, it can be a third person plural denoting a generalized human subject frequently cited as a source of passives (Givón 1979), meaning that there was cutting down of the tree and that this action has no determinate subject. Impersonal here means simply that the third plural element is not understood to refer to any specific group of individuals.

In short, the prevalence of independent morphological, lexical, syntactic, and referential ambiguities can lead to a combinatorial explosion, making many Wolof sentences massively ambiguous.

## 4 IMPLEMENTATION OF THE WOLOF GRAMMAR

In the previous section, we have briefly looked at different ambiguity issues in Wolof, paying attention to those relevant to the discussion of developing a grammar for the language. In what follows, I will suggest and discuss some methods for handling these issues, keeping in mind that the application of some of these techniques always has the potential to eliminate a valid analysis. Before suggesting these techniques, I would like to briefly describe the Wolof grammar and the data used for grammar development and evaluation.

[7] Note that the English pronoun *they* in the translation of Example (19b) is to be considered here as a referential and non-arbitrary pronoun.

Developed as part of the Parallel Grammar (ParGram) project (Butt *et al.* 2002), the Wolof grammar provides a formal description of the syntactic analysis of core constructions of the language within LFG, as well as linguistically well motivated analyses of challenging constructions in Wolof, including clitics (Dione 2013a), clefts (Dione 2012a), valency change and complex predicates (Dione 2013b). The grammar parses sentences on the basis of XLE rules and templates, two lexicons, and a cascade of finite-state transducers (FST) (Kaplan *et al.* 2004). In its current state, the grammar has 250 rules (with right-hand sides based on regular expression). The lexicons contain ca. 2000 verb stems and 2836 subcategorization frame–verb stem entries. The preprocessing components of the grammar include a Wolof finite-state morphological analyzer (WoMA) (Dione 2012b), as well as other finite-state modules for tokenization and normalization. The grammar is not part of an application-oriented set-up, meaning that it is not embedded in a larger application pipeline. Consequently, some sources of information that could be applied to eliminate inappropriate readings that may come out of the parser/grammar, such as domain restrictions and selectional restrictions, are mostly not available.

The development of the grammar is based on a corpus of natural Wolof texts. The basic development and test (i.e., unseen) data consist of two disjoint sets of randomly selected sentences from short stories (Cissé 1994; Garros 1997) and a semi-autobiographical novel (Ba 2007). The development and test sets consist respectively of a total of 626 and 2364 sentences used to evaluate the grammar in terms of accuracy and efficiency, but also to assess the effects of design decisions in the grammar and the impact of the disambiguation methods discussed within this work. As the grammar constitutes a starting point for the construction of further NLP resources for Wolof, the test set was run through it to establish a treebank for the language.

## 5 MORPHOLOGICAL AND LEXICAL DISAMBIGUATION

This section presents systematic approaches used to manage the morphological and lexical ambiguities discussed in Sections 3.1.1-3.1.3. It focuses on the formal encoding of noun class indeterminacy, lexical specifications and CG-based disambiguation.

## 5.1 *Ambiguity resolution for Wolof noun classes*

In the initial LFG approach to Wolof noun classes, nominal class attributes were represented as atomic feature values. So, the noun and its specifier were elements of the f-structure and had to agree either in the singular (e.g., *NOUN-CLASS-SG*) or plural (e.g., *NOUN-CLASS-PL*) noun classifier. Accordingly, the f-structure for the nominal phrase in (20) was represented as shown in (21). This f-structure representation says that the noun *xale* specifies *b* and *y* as its respective singular and plural noun class, while the specifier *bi* belongs to the *b* class.

(20) *Xale b-i*
child cl-DFP
"The child"

(21)

| | | | |
|---|---|---|---|
| PRED | 'xale' | | |
| NOUN-CLASS-SG | b | | |
| NOUN-CLASS-PL | y | | |
| NUM | sg | | |
| PERS | 3 | | |
| SPEC | DET | PRED | 'bi' |
| | | NOUN-CLASS-SG | b |
| | | DEIXIS | proximal |
| | | DET-TYPE | def |

One potential problem with this analysis is that Wolof noun classes typically have forms that can be attributed 'indeterminately' to different values. As Dalrymple *et al.* (2009, pp. 31) noted, "forms that are indeterminately specified for the value of a feature can simultaneously satisfy conflicting requirements on that feature and thus are a challenge to constraint-based formalisms which model the compatibility of information carried by linguistic items by combining or integrating that information."

Similarly, Wolof nouns typically show no noun class distinction. As Example (22) illustrates, a noun like *ndaw* in (8) can satisfy different class requirements. A similar case has been observed for the German noun *Papageien* 'parrots' in (23), which shows no case distinction and can satisfy different CASE requirements (Dalrymple *et al.* 2009).

(22) *Ndaw*
young
G/L/S/Ñ/Y
'young/youth/messenger (*g*, *l*, *s*, *ñ* or *y* noun class)'

(23) *Papageien*
parrots
NOM/ACC/DAT/GEN
'parrots' (nominative, accusative, dative or genitive)

Because the approach given in (21) relied on specification of simple atomic values for indeterminate features, the integration (typically by unification) of information from head and dependent was problematic. Assuming that a noun like *mag* 'old person' (see Section 3.1.1), for instance, specifies *ñ* for its specifier's nominal class value, and that the determiner *a-y* 'some' and the relative pronoun *ñ-u* 'who/which' specify *ñ*, we obtain a clash of nominal classifiers (e.g., [NOUN-CLASS-PL = y] and [NOUN-CLASS-PL = ñ]) in sentences like (24), leading to the incorrect prediction that the example is unacceptable.

(24) *Ma gis a-y ndaw ñ-u am xam-xam.*
1sg see NDEF-cl young.people cl-Rel have knowledge
"I saw some wise young people."

This problem implies shifting away from the initial approach described in (21). This shift in approach has two different, but interrelated, objectives: to avoid coverage problems for cases like (24), which show that indeterminate forms can stand in for two values simultaneously (like syncretic forms in many languages); and to reduce the number of readings for normal cases by assuming a suitable underspecified representation rather than a disjunctive listing of all options.

Accordingly, the analysis in (21) above is replaced by an approach similar to the representation of CASE proposed in Dalrymple *et al.* (2009).[8] Following this representation, nouns such as *ndaw* and *mag* in (9) will have the feature structure for the noun class attribute, as respectively shown in (25) and (26).

(25) Noun class feature for *ndaw*

$$\begin{bmatrix} \text{NOUN-CLASS} & \begin{bmatrix} \text{G} & + \\ \text{L} & + \\ \text{S} & + \\ \text{Ñ} & + \\ \text{Y} & + \end{bmatrix} \end{bmatrix}$$

(26) Noun class feature for *mag*

$$\begin{bmatrix} \text{NOUN-CLASS} & \begin{bmatrix} \text{J} & + \\ \text{M} & + \\ \text{Ñ} & + \\ \text{Y} & + \end{bmatrix} \end{bmatrix}$$

The value of this attribute allows specification of each noun class by means of a separate Boolean-valued attribute: G, L, S, Ñ, Y, etc. A

[8] Alternatively, the set-based approach to feature resolution (Dalrymple and Kaplan 1997) could be used to handle feature indeterminacy. It allows for an account of complex agreement phenomena like those found in German free relatives, case in Polish coordination and noun class in Xhosa coordination.

negative value indicates the inability of a form to satisfy the corresponding noun class requirement. Nouns and their modifiers specify negative values or do not specify any value for the noun classes they do not express, and specify or are compatible with positive values for the classes they do express.

The noun class specification for the form *ndaw*, which is class-indeterminate, is given in (27); this can be read as requiring that, within the NOUN-CLASS structure, the value for G, L, S, Ñ or Y must be +. Thus, *ndaw* must express some noun class or other, but there are no restrictions on which noun class it expresses. This permits the form to occur in contexts compatible with a positive specification of one or more of the noun classes, and does not impose any negative class specification that would rule out class possibilities for the form.

(27) *Ndaw*; NOUN-CLASS{G|L|S|Ñ|Y} = +

The output for the word form *ndaw* produced by the Wolof morphological analyzer (Dione 2012b) is shown in (28). The FST translates the form into a string that represents its morphological makeup: a noun that agrees with its modifier in the classes *g*, *l*, *s*, *ñ* or *y*. All class indexes compatible with this form should be contained in the output.

(28) ndaw+Noun+Common+g+l+s+ñ+y

We may note in passing that, in the Wolof lexicon, polysemous and homonymous nouns are treated in a similar way. This means that words like *ndaw* that have different related and unrelated meanings are associated with only one lexical entry. This follows the goal to reduce ambiguity for lexical items that have many readings which, however, do not affect the syntax. A similar approach has been taken by the ParGram LFG English grammar (Riezler *et al.* 2002). The different readings of a polysemous item like *bank* (“river bank” or “financial bank”) are not distinguished in the grammar, but rather in a semantic post-processor, that is, the English transfer rules.

In the Wolof grammar, this underspecification approach led to substantial reductions in morpholexical ambiguity and parse time. To assess the impact of underspecification, the ambiguity rate and the time the grammar needs to parse the test data have been measured before and after the application of this approach. The results show

that the ambiguity rate decreased by approximately 8%, leading to a reduction of parse time by 4%. In the grammar, this change affected ca. 10 rules, 20 morphological tags, and 39 templates, which are called at many places in the different rules and lexical entries.

In the context of grammar implementation, the advantage of underspecification over the disjunctive approach in terms of processing efficiency has also been attested in previous work (Flickinger 2000; Crysmann 2005). According to Flickinger (2000), the compactness of linguistic description achieved by the elimination of disjunctive features provides a great benefit in terms of processing efficiency. The performance comparison of the disjunctive and the underspecification approach shows that the latter outperforms the former by a factor of 3–4, with an otherwise unchanged grammar[9] running on the same processing platform (PAGE, Uszkoreit *et al.*, 1994).

### 5.2 *Disambiguating co-verbs using ne/ni*

Like Wolof noun class ambiguity, ambiguity caused by ideophones are dealt with using a systematic approach. Wolof ideophones behave like particles that are selected by the verb. In this respect, they show some similarity to Norwegian particles like *ut* in the sentence in (29).

(29) *Han vil slippe ut hund-en.*
3sg will release out dog-DEF
"He will let the dog out."

Given this similarity, the coverbal ideophones are treated as particles. As in the Norwegian grammar (Dyvik 2000), these particles are introduced by a special c-structure category PART of adverbial type (i.e., PART[adv]). The verbs like *ne* which subcategorize for ideophones are constrained to specify the lexical form (30) of the particle.

(30) V-SUBJ-PRT (P PART) = @(CONCAT P '* PART %FN)
@(VOICE (↑ PRED) = '%FN < (↑ SUBJ) > ')
(↑ PRT-FORM) =$_c$ PART.

The rule in (30) makes use of the XLE built-in template *CONCAT* (Crouch *et al.* 2013) to concatenate all arguments except the last argument and to produce the final argument. If applied to the structure *ne*

[9] The LinGO English Resource Grammar (Copestake and Flickinger 2000).

*tekk* in (11), %FN will be set to *ne*tekk* once the *tekk* particle is found and (↑ PART) is set to *tekk*. The rule shows a subcategorization frame for an intransitive verb like *ne* functioning as the verb in the structure. The frame uses a constraining equation (↑ PRT-FORM) = c PART to require a special particle selected by the verb, constraining the value of the 'PRT-FORM', introduced by *PART*. This rule also specifies that such a structure – the verb and co-verbs taken together – requires a special treatment. The *CONCAT* device allows for concatenation of two independent lexical entries that coreference each other in the lexicon. The lexical entries of base verbs introduce the semantic form of the particle verb with its argument structure. The lemma of the base verb and the form of the particle are concatenated via the device so that the combination of the two, rather than just the lemma of the base verb, is the PRED of the f-structure.

One of the reasons for treating the verb and the particle in this way is that syntactic constituents can intervene between the verb and the particle, as illustrated in (31). Another reason is that, for instance, the verb *ne* can appear with an OBJ, but not if there is a PRT-FORM *tekk*, which is provided by the particle. In such a case, this verb can only be intransitive.

(31) *Mu ne ma jàkk.*
3sg say 1sg IDEO:of staring at someone
"S/He was staring at me."

Figure 2 shows an example analysis of Wolof coverbal ideophones.

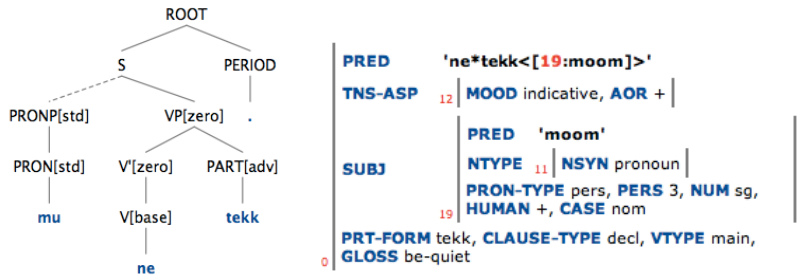



Figure 2: Analysis of Sentence (11) as an illustration of the treatment of coverbal ideophones in Wolof

In addition, some further steps were required. First, the ideophonic particle forms had to be explicitly listed in the lexical entry of the collocational verb. Second, several subcategorization frames were

defined to allow for the different verb argument structures. For a verb like *ne*, the frames given in Table 1 were defined.[10] Finally, optimality marking (see Section 6.2) was used to state a preference reading for ideophones as such, when they occur with a collocational verb. For instance, some rare ideophones like *tekk* in (11) may belong to another grammatical category. In fact, *tekk* can also be a noun. However, in this configuration, the noun reading is very unlikely, not to say impossible. Hence, for such rare cases, the use of the preference mark for the ideophone reading helps to discard the implausible readings arising from nouns from the output.[11]

Table 1: Subcategorization frames for *ne* as a collocational verb

| Subcategorization frame | Examples |
|---|---|
| V-SUBJ-PRT | *ne cell* 'to be silent' |
| V-SUBJ-OBJ-PRT | *ne jàkk* 'to stare at someone / something' |
| V-SUBJ-OBL-TH-PRT | *ne mërr ak* 'to disappear with' |
| V-SUBJ-OBJ-OBJ-TH-PRT | *ne keww kenn dara* 'to stare at somebody with something' |
| V-SUBJ-XCOMP-PRT | *ne mes ànd ak njaxlaf* 'to disappear quickly and in a dynamic way' |
| V-SUBJ-OBL-COMPAR-PRT | *ne ràyy ni melax* 'to flash like a lightning' |

Applied on the test set, this approach substantially reduces the ambiguity rate related to the coverbal ideophones by ca. 4%. Also, the parse time for coverbal ideophones could be reduced by 16%, while maintaining the parsing accuracy. This change affected 7 templates and ca. 139 verb subcategorization frames.

### 5.3 *Coping with POS ambiguity*

One of the major causes of non-determinism in a computational grammar is POS ambiguity. When a word can belong to two different grammatical categories, a non-deterministic parser may have to explore both possibilities.

As noted in Section 3.1.3, *la* in Wolof is very ambiguous between different grammatical categories, and because of this, the sentence in (16a), repeated in (32), has ca. 42 readings. The multi-tagged text of this sentence before disambiguation is displayed in (33). The analysis

[10] *PRT* is the abbreviation for *particles*.

[11] See Section 6.2 for a more detailed discussion of using optimality marks in the Wolof grammar.

line <"la"> has received seven different readings in the morphology analysis.

(32) *Fas* ***la*** *gis.*
horse.w-cl 3sg.FOC see
'It is the horse that he saw.'

(33) <"fas"> fas + V + Base + Main + Active
fas + N + Common + w + y + Count
fas + N + Common + g + y + Count

<"la"> la + Comp + Free
la + Det + Def + l + Sg + Dist
la + Pron + Rel + l + Sg + Dist
la + Pron + Free + l + Sg + Dist
la + INFL + NonSubjCopula + 3SgSubj
la + INFL + CompFoc + 3SgSubj + Indic
la + Clt + Obj + Pers + 2 + Sg + Weak + Acc

<"gis">
gis + V + Base + Main + Active
gis + N + Common + b + y + Count

<".">
. + PERIOD

A possible method to tackle the non-trivial issue of POS ambiguity is to use a methodological paradigm that is based on local morphological disambiguation performed by context-sensitive disambiguation constraints. Local disambiguation refers to "constraints or strategies that make it possible to discard some readings just by local inspection of the current cohort" (i.e., the set of readings from a word form) "without invoking any contextual information" (Karlsson 1990, pp. 2).

Constraint Grammar (CG) (Karlsson 1990) is an example of such a mechanism that allows this kind of local disambiguation. CG is a language-independent formalism for surface-oriented, morphology-based parsing of running text (Karlsson 1990). In this formalism, context dependent rules are compiled into a grammar that assigns readings to words or other tokens in a given text. Tags can be of different types, including lexeme, base form, syntactic or semantic tags, valency, etc. Constraints are used to discard as many alternatives as possible. Constraint rules typically consist of two parts: (i) an operation on a pattern and (ii) a context. Each rule either adds, removes,

selects or replaces a tag or a set of grammatical tags in a given sentence context. A context can be defined as any combination of words or tags in a given sentence. Context conditions can be linked to any tag or tag set of any word anywhere in the sentence, either locally (defined distances) or globally (undefined distances). Context conditions in the same rule may be linked (i.e., conditioned upon each other) negated, or blocked by interfering words or tags.

The idea that lexical ambiguity can be reduced for a given sentence by using the CG model is particularly attractive for at least two reasons. First, the CG-based model does not require a large data set for training. Second, the model allows a grammar writer to select meanings or remove them from words or other tokens, depending on local information. The context sensitive constraints of this model provide a disambiguation possibility that is generally unavailable in context-free grammar approaches. This constitutes one of the main motivations of using CG in this work.

Note that the development of the Wolof grammar follows in many respects Maxwell and Kaplan's (1993) model, according to which parsing time can be speeded up if conditions on certain finite-valued syntactic features are translated from f-structure constraints to variant c-structure categories. This means that the constraints can be enforced by the polynomial context-free c-structure system and not by the possibly exponential f-structure satisfiability algorithm. This works particularly well for features that can be evaluated fairly locally in the tree. For instance, the Wolof grammar uses parameterized c-structure categories (also known as complex categories) (Crouch *et al.* 2013; Butt *et al.* 1999) provided by XLE as a way of systematically propagating and enforcing features that provide subclasses of context-free categories. However, while this approach is beneficial as it provides the means to prune inconsistent analyses early (i.e., in the chart building phase instead of the unification phase), it does not provide the same gain in efficiency as the separate CG component does. One of the reasons is that the use of parameterised c-structure categories also increases the number of categories which are built in the XLE chart. A further, and perhaps more important, reason is that, with the CG-based approach, XLE will not even try to build c-structure for the undesired analyses, as these readings are removed at earlier stages.

Accordingly, morphological disambiguation based on CG has been incorporated into the Wolof grammar. The implementation of the CG model used for Wolof is developed by Didriksen (2003) within the VISL NLP framework,[12] and is based on the third-generation compiler *vislcg3* (Bick 2000). As the Wolof CG disambiguator is discussed in details in Dione (2014), I will here only briefly outline the use of the CG model to handle lexical ambiguity.

To illustrate how CG-based disambiguation works for Wolof, let us consider Example (33). In order to remove undesired readings for this input sentence, a number of detailed constraints have been developed. Some of these are exemplified in the rules in (34)–(36), which are written in accordance with the CG-3 compiler documentation.[13]

In (33), a large number of ambiguities can be resolved by looking at the Wolof noun class agreement. For instance, specifiers such as determiners or demonstratives and modifiers such as relative pronouns agree with the head noun. Accordingly, those analysis lines in (33) which contain a noun class tag that does not occur in the analysis line of the adjacent noun can be removed. This means, for example, that the determiner reading of *la* can be safely removed: It refers to the *l* class which differs from the possible classes for the noun *fas*, which can take either the *g* or the *w* index. This is accomplished by the constraint rule in (34).

(34)
```
REMOVE (Det Def) + $$NC
IF (NEGATE -1 NOM + $$NC)
    (NEGATE *-1 Pron + $$NC BARRIER CLB);
```

- `REMOVE (Det Def) + $$NC`: remove a definite determiner with a noun class index, `IF`
  - `(NEGATE -1 NOM + $$NC)`: there is no nominal (`NOM`), with the same class, occurring immediately to the left (`-1`).
  - `(NEGATE *-1 Pron + $$NC BARRIER CLB)`: there is no pronoun with the same noun class anywhere (`*`) to the left of the first neighboring position, and there is no clause boundary (`CLB`) in between (`BARRIER`).

[12] See `http://beta.visl.sdu.dk`.

[13] See Bick (2009) and `http://beta.visl.sdu.dk/cg3.html`.

Likewise, the relative pronoun reading can be removed using a rule similar to (34). In addition, relative pronouns can be directly removed in a more general context, for example, if the right adjacent constituent is a prepositional phrase, a conjunction or a punctuation symbol ('.', parenthesis, etc.).

The rule in (35) removes the non-subject copular reading, depending on the part of speech of the left adjacent and right adjacent word.

```
(35)  REMOVE (Icop)       IF  (-1 Verb LINK 2 Verb);
```

- `REMOVE (Icop)`: remove a copular reading, `IF`
  - `(-1 Verb LINK 2 Verb)`: left adjacent and right adjacent words are verbs.

The rule in (36) removes *la* as a complementizer if an unambiguous (C) transitive verb occurs anywhere to the right from the first neighbouring position, and if there is no clause boundary in between.

```
(36)  REMOVE ("la" Comp)
      IF (*1C (Verb Trans) BARRIER CLB);
```

Having applied the rules in (34-36), only three analysis lines of *la* in (33) will be retained. While many local ambiguities can be resolved using the given rules, in some cases it is difficult to fully disambiguate. For example, the disambiguation of free relative pronouns and object clitics requires a careful rule design. With respect to the example discussed so far, in the current Wolof CG disambiguator, the surviving analysis lines may remain undisambiguated.

The Wolof CG disambiguator consists of a modest size of rules (ca. 250 rules), but is relatively effective. Applied on the Wolof test data (Cissé 1994; Garros 1997; Ba 2007), it helped to reduce the average numbers of readings per token from 2.69 to 1.55. The Wolof CG disambiguator is evaluated along with the c-structure pruning mechanism which has been used to tackle some issues of syntactic ambiguity, as discussed in Section 6.1.

## 6 SYNTACTIC DISAMBIGUATION

The simplest method of reducing syntactic ambiguity would be to write more restrictive rules. In some cases, it could be possible to find a restriction that rules out exactly the undesired analyses, for example,

by disallowing attachment of some PPs to the sentence level in ambiguity involving PP attachment. This strategy is obviously not always possible, as it may lead to incorrect analyses (e.g., of some PPs) or eliminate analyses containing particular ambiguities (e.g., global ambiguities) that need to be preserved. Accordingly, structural or scoping ambiguities have often been dealt with by ranking the different analyses, using either statistical models or linguistic intuition.

To deal with syntactic ambiguity in Wolof, I have explored various disambiguation models, including probabilistic as well as non-statistical ones. The former build upon the c-structure pruning mechanism of XLE (Cahill *et al.* 2007, 2008; Crouch *et al.* 2013), while the latter are based on optimality marks (Frank *et al.* 2001). In addition, I adopt ambiguity preserving approaches for constructions involving global ambiguity. These different approaches are discussed in the following sections.

### 6.1 *Coping with structural ambiguity by using c-structure pruning*

In Section 3.2.1, structural ambiguities in Wolof have been discussed. In Example (18), the word form *bi* can have different grammatical categories. For instance, it can be a determiner or a relative pronoun, leading to different c-structures, for example, for the constituent *xale bi moom*, which can be analyzed as a DP or as an NP with an embedded relative clause. The probability that this constituent occurs as a relative NP in some given texts is lower than the probability that the same constituent occurs as a DP. Similar facts can be noted about the constituent *doon ree*, which, in principle, is much more likely to be an auxiliary VP (VPaux) than a copular VP (VPcopmain). A probabilistic grammar takes these probabilities into account in a way that a non-probabilistic grammar does not. Accordingly, it is possible to assign a probability to a sentence, and base a given analysis of the sentence (from the set of possible analyses) on the probability associated with it.

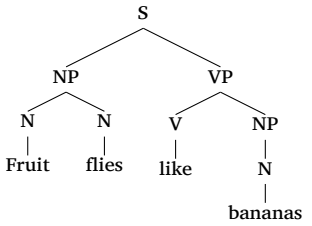
Thus, to deal with structural ambiguities such as those discussed in Section 3.2.1, I have conducted various experiments based on the c-structure pruning mechanism of XLE (Cahill *et al.* 2007, 2008; Crouch *et al.* 2013), in combination with CG during the development of the Wolof grammar. The experiments are extensively discussed in Dione (2014). In the following, I will outline the main aspects and results achieved by using this approach.

The c-structure pruning mechanism of XLE provides a possible method to control structural ambiguities and to make parsing faster by discarding low-probability c-structures before functional annotations (f-annotations)[14] are solved. Typically, XLE parses a sentence in a series of passes (Crouch *et al.* 2013). First, the morphology analyzes the sentence, looks up each morpheme in the lexicon and initializes a chart with the morphemes and their constraints. Then, the chart builds all possible constituents out of the morphemes using the c-structure rules given in the grammar. Constraints are processed after all of the constituents have been built. Next, the unifier processes the constraints bottom up, only visits those constituents that are part of a tree with the correct root category that covers the sentence, and builds a constraint graph for each subtree. Subsequent passes are concerned with finding locally incomplete analyses and solving the Boolean satisfaction problem for edges. One main reason for using c-structure pruning is that unification is typically the most computation-intensive part of LFG parsing. This is particularly true for Wolof. The typical proportions of overall runtime of some XLE components with the Wolof grammar are: Morphology (0.1%), Chart (3.1%) and Unifier (85.5%).

The basic procedure of testing the c-structure pruning mechanism consists in training a probabilistic context-free grammar (PCFG) on a corpus annotated with syntactic bracketing, and, subsequently, discarding all c-structures that are *n* times less probable than the most probable c-structure. Context-free rewrite rules typically consist of one non-terminal symbol on the left-hand side and a combination of terminal and/or non-terminal symbols on the right-hand side. XLE grammar rules are context-free rules augmented with f-annotations. Examples of PCFGs are given in Figures 3–4, which represent two different analyses of the sentence “Fruit flies like bananas”.

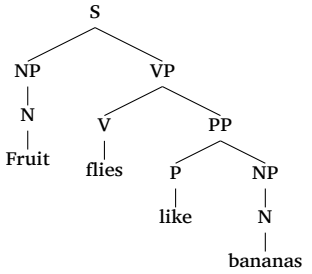
As can be seen in the Figures 3–4, each c-structure has hypothetical probabilities attached to it: 8.4375E-14 and 4.21875E-12 for Analysis 1 and Analysis 2, respectively. Accordingly, Analysis 1 is 50 times less probable than Analysis 2. Thus, depending on how the c-

[14] Functional annotations refer to the set of f-structure constraints associated with the analysis of a sentence. For example, the constraint ($f$ TENSE) = PAST specifies that the feature TENSE in the f-structure $f$ has the value PAST.




| | | | |
|---|---|---|---|
| S | → | NP VP | 0.5000 |
| NP | → | N N | 0.1500 |
| N | → | Fruit | 0.0010 |
| N | → | flies | 0.0015 |
| VP | → | V NP | 0.2000 |
| V | → | like | 0.0050 |
| NP | → | N | 0.5000 |
| N | → | bananas | 0.0015 |
| | | | 8.4375E-14 |

Figure 3:
Analysis (1) for the string *Fruit flies like bananas* with hypothetical probabilities




| | | | |
|---|---|---|---|
| S | → | NP VP | 0.5000 |
| NP | → | N | 0.5000 |
| N | → | Fruit | 0.0010 |
| VP | → | V PP | 0.1000 |
| V | → | flies | 0.0025 |
| P | → | like | 0.0500 |
| PP | → | P NP | 0.9000 |
| NP | → | bananas | 0.0015 |
| | | | 4.21875E-12 |

Figure 4:
Analysis (2) for the string *Fruit flies like bananas* with hypothetical probabilities

structure pruning mechanism is set, Analysis 1 may be discarded even before corresponding f-annotations are solved.

The probabilities for the rule can be estimated as relative frequencies found in a parsed (and disambiguated) corpus.[15] With these estimations, XLE makes use of a chart-based mechanism to prune subtrees at the level of individual constituents in the chart. A subtree is

[15] See Crouch *et al.* (2013) on how XLE computes the rule probabilities.

pruned if its probability is lower than the best probability by a given factor. For that purpose, the grammar writer can specify a so-called **cutoff** value (typically between 4 and 10), which corresponds to the natural logarithm of that factor. For instance, a value of 5 means that a subtree will be pruned if its probability is about a factor of 150 less than the best probability.

To test the c-structure pruning mechanism for Wolof,[16] a PCFG was built, trained and tested in two different ways: (i) only using the regular Wolof grammar without CG-based disambiguation, and (ii) using the CG parser (see Section 3.1.3) for morphological disambiguation. This had the purpose of evaluating the parsing system in terms of parsing time, accuracy and ambiguity reduction. The LFG metric used to measure the parsing quality is based on the comparison of full f-structures, represented as *relation(predicate,argument)* triples. Accordingly, the triples of the system are compared to a triple-based gold standard manually built for this purpose. For each comparison, the best match, that is, the reading that comes closest to the intended analysis (out of all source analyses) is chosen. The metric, referred to as the **oracle f-score** is defined as the geometrical mean of precision and recall (i.e., $F = (2 * P * R)/(P + R)$) which is calculated from the set of the triples in best match solution.

The results of applying the c-structure pruning mechanism on the development set, as reported by Dione (2014), show that a cutoff of 10 seems to provide the best trade-off between time and accuracy, if the LFG parsing is not combined with CG.[17] Otherwise, if CG-based disambiguation is used in addition to c-structure pruning, a cutoff of 9 seems to perform best on the development data. Having established the best cutoff values for the two training forms, the c-structure pruning mechanism is applied to the Wolof test set.

The results on the test set are given in Table 2. These show that c-structure pruning and CG-based disambiguation, independently, yield a great reduction in parsing time. Using only the c-structure pruning (with a cutoff of 10) leads to a speed-up over 36%. If the test set is disambiguated using CG, a cutoff value of 9 allows for a speed-up of

[16] See Dione (2014) for details about the experiments, the training data, and on how the gold standard data have been built.

[17] For a discussion of how the pruning algorithm is trained on the Wolof data and the process used to establish the best cutoff values, see Dione (2014).

30%. Using only CG-based disambiguation, parsing efficiency can be improved by ca. 40%. In total, combining c-structure pruning with CG-based disambiguation leads to a speed-up of 58%.

| | Without CG | | With CG | |
|---|---|---|---|---|
| Pruning Level | None | 10 | None | 9 |
| Total CPU Time (sec) | 7374 | 4779 | 4473 | 3164 |
| Oracle f-score | 93.02 | 92.05 | 90.52 | 89.40 |
| # Full Parses | 1712 | 1613 | 1551 | 1434 |
| # Fragment Parses | 627 | 737 | 775 | 917 |
| # Time Outs | 10 | 5 | 8 | 6 |
| # Skimmed Sentences | 348 | 240 | 191 | 125 |

Table 2:
Results of the c-structure pruning experiments on Wolof test data

However, as can be seen in Table 2, this increase in speed leads to a relatively significant drop in f-score. The c-structure pruning and CG-based disambiguation, independently, have a negative impact on the quality of the f-structure: The number of fragment parses increases.[18] Without CG-based disambiguation, a cutoff of 10 leads to a drop in f-score of 0.97 points. CG pre-filtering without c-structure pruning causes a drop in f-score of 2.5 points. Using CG-based disambiguation and a cutoff of 9, the f-score decreases by 1.12 points. In total, combining c-structure pruning (with a threshold of 9) with CG-based disambiguation results in a drop in f-score of 3.62 points.

| | | Pruning Cutoff | Ambiguity Rate | Ambiguity Reduction |
|---|---|---|---|---|
| 1 | w/o CG | None | 209.77 | 72.92% |
| | | 10 | 56.81 | |
| 2 | w/o CG | None | 174.38 | 77.64% |
| | with CG | None | 56.45 | |
| 3 | w/o CG | None | 154.66 | 80.66% |
| | with CG | 9 | 29.92 | |

Table 3:
Ambiguity reduction when using c-structure pruning and CG-based disambiguation

Table 3 shows the ambiguity reduction achieved by using the c-structure pruning algorithm and CG. Because the ambiguity rate was

[18] Fragments are produced when the grammar is unable to provide a full parse for the input sentence. This partial parsing technique allows the sentence to be analyzed as a sequence of well-formed chunks with both c-structure and f-structure associated with them. Similarly, skimmed parses are produced, when the amount of time or memory spent on a sentence exceeds a threshold. This technique is used to avoid time-out and memory problems.

measured relative to the common full parse solutions produced by the specific test run, the values for ambiguity rate are not absolute, but rather relative values. Combining c-structure pruning with CG-based disambiguation (Row 3) provides the best results with over 80% ambiguity reduction.

While statistical disambiguation is convenient if a corpus annotated with syntactic bracketing exists, it is also a source of errors, which are often caused by a lack of data. Also, the application of this disambiguation technique may be inappropriate in some cases. On the one hand, c-structure pruning will not often be able to disambiguate between two constructions if they are both very frequent in the corpus data. For example, in Wolof, constructions involving asyndetic coordination might be undesired in many cases. They do, however, have a relatively high frequency in the data, so that a statistical disambiguator will not readily prune them, and even if it did, this would often result in incorrect analyses. On the other hand, the c-structure pruning mechanism cannot be used to manage some syntactic ambiguities like those discussed in Section 3.1.4, which involve the f-structure rather than the c-structure. Thus, managing such ambiguity might require the use of non-statistical mechanisms such as optimality marking (Frank *et al.* 1998, 2001).

### 6.2 *Using optimality marks*

When dealing with syntactic ambiguities, humans can make use of extra-linguistic knowledge and context. Parsers, however, have only the grammar as a knowledge base and they deliver all possible solutions, including potentially many implausible ones. This might adversely affect parsing efficiency, often making a manual correction of the output necessary. In this respect, a possible method to constrain these ambiguities to legitimate cases and to indicate a preference for one syntactic analysis over another is the use of the formal mechanism based on optimality marks (Frank *et al.* 1998, 2001).

Optimality Theory (OT) was first developed by Prince and Smolensky (1993) for phonology, and later extended to other areas such as syntax and semantics. Theoretical OT models grammars as systems that provide mappings from inputs to outputs; the inputs are viewed as underlying representations and the competing output candidates (or analyses) as their surface realizations. Accordingly, grammars are

seen as having a set of violable constraints. The constraints are universal and ranked by each language, giving rise to cross-linguistic variation. Constraint ranking determines the winning candidate, that is, the candidate that incurs fewer violations than all other candidates.

For example, given constraints C1, C2, and C3, where C1 dominates C2, which dominates C3, A is optimal if it outperforms B on the highest ranking constraint which assigns them a different number of violations. If A and B tie on C1, but A does better than B on C2, A is optimal, even if A has 100 more violations of C3 than B. Table 4 shows an example of the standard table notation for OT analyses.

| | /**Input**/ | CONSTRAINT 1 | CONSTRAINT 2 | CONSTRAINT 3 |
|---|---|---|---|---|
| ☞ | Candidate A | * | * | *** |
| | Candidate B | * | **! | |

Table 4: Example of the standard table notation in OT

The optimal candidate is highlighted with a pointing finger in the tableau, and each cell displays an asterisk for each violation for a given candidate and constraint. Once a candidate does worse than another candidate on the highest ranking constraint distinguishing them, it incurs a fatal violation, resulting in the elimination of the candidate (marked in the tableau by an exclamation mark '!'). Once a candidate incurs a crucial violation, there is no way for it to be optimal, even if it outperforms the other candidates on the rest of the universal constraint set.

The OT model has been adopted and extended within the LFG framework for ranking preferences and constraints. Two fundamental aspects in the extension of the OT model in LFG can be described as follows:

- The model used in LFG is a violable constraint system used as a preference filter on analyses. The possible analyses are ranked using this preference filter, which does not necessarily rule out sub-optimal structures entirely;
- The violation of a constraint is not always negative. There are positive constraints, whose fulfillment is desired in some context.

In LFG, OT is an additional projection (*o*-projection or *o*-structure) formally defined as a multiset of constants (constraints or 'optimal-

ity marks'). The constraints are projected from c-structure (Frank *et al.* 1998) and are introduced by *o*-descriptions within the grammar. The *o*-structure serves as a record of constraints used for each candidate analysis.[19] The XLE grammar development environment provides an implementation of OT in LFG, incorporating the idea of ranking and (dis)preference. This utility allows for filtering syntactic and lexical ambiguities in a way that aims to reconcile robustness and accuracy.

Unlike theoretical OT which only includes dispreference marks, XLE OT defines both preference and dispreference marks. Preference marks come to use when one specific reading out of a set of analyses is preferred. In general, they allow one to mark more frequent structures, which are preferred to the less frequent ones. Example (37) (from Frank *et al.* 1998, pp. 5) illustrates the use of such marks to state a preference for multiword terms in technical documentation.

(37) a. I want [print quality] images.
b. *I want [print] [quality] images.

With a respective preference mark, the analysis with the multiword expression (37a) will be preferred over all other readings. However, if there is no valid analysis for the multiword expression (as in 37b), an analysis using the individual lexicon entries is still possible.

Dispreference marks are used for rare grammatical constructions which need to be covered, but interact in unexpected ways with frequent constructions, making them 'dispreferred'. For example, these marks may be used to exclude NPs being headed by adjectives from the candidate set. Dispreferred constructions are selected only when no other, more plausible, analysis is possible. Yet, it can be difficult, in general, to decide whether to use a preference or dispreference mark. The difficulty stems mainly from two main issues: (i) whether there is any interaction between the marks, and (ii) which analysis is easier to mark. For instance, it is easier to mark a multiword expression with a preference than to mark all of its components with a dispreference.

In addition, XLE provides other special marks such as STOPPOINTs. STOPPOINT marks slowly increase the search space of the

[19] Note that the *o*-structure is just a set of marks, not of f-structure features. The f-structure of the grammar remains unaltered. Optimality marks are in their own projection, the extra representation level referred to as the *o*-structure.

grammar if no good solution can be found. They constitute a way of increasing the robustness of the grammar without sacrificing performance. For instance, in the OT field of the configuration, STOPPOINT marks can be inserted into the hierarchy of preference and dispreference marks (e.g., to the right or left of STOPPOINT in optimality order). As such, dispreferred constructions like rare or computationally expensive constructions will only be considered if the core grammar fails to find a valid analysis for frequent ones.

In the context of Wolof, I have conducted experiments with the OT mechanism following two main objectives: (i) to select preferred analyses for ideophones (see Section 5.2) and ambiguous verb subcategorization frames (see Section 6.3), and (ii) to increase robustness by managing ambiguity caused by computationally expensive constructions like coordination without an overt conjunction (see Section 6.4).

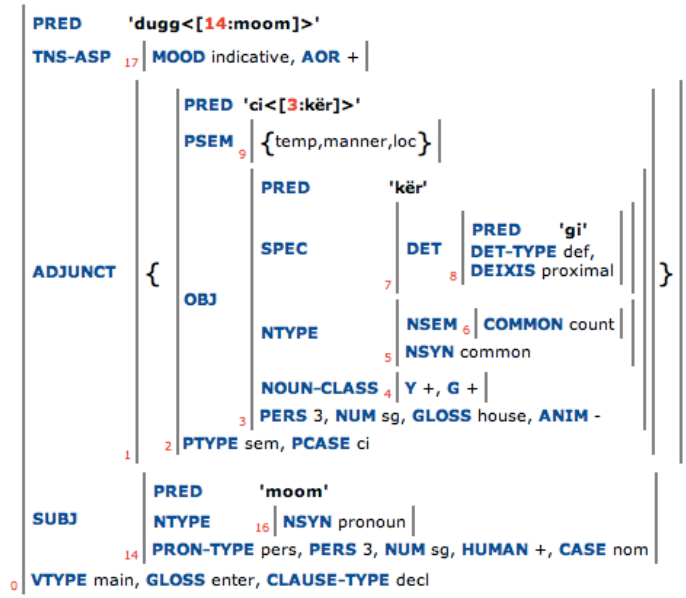
### 6.3 *Managing ambiguity caused by subcategorization frames*

Section 3.1.4 discussed verbs in Wolof like *dugg* "enter" in (17), repeated in (38), which have several subcategorization frames. These include a bare intransitive and an oblique reading, as illustrated by Figures 5 and 6, respectively.

(38) *Mu dugg ci kër gi.*
3sg enter in house cl.DFP
"S/He entered the house."

For Wolof, I have attempted to suppress ambiguity caused by subcategorization frames through the use of optimality marks. Accordingly, I have introduced preference marks in the grammar to help indicate the preferred reading in (38), for example, to select the oblique reading over the adjunct one. As Example (39) shows, the OT constraints are used within a disjunction at the level of functional annotation. This example specifies that a Wolof verbal phrase (VP) may expand into a verb V followed by an optional determiner phrase (DP) and several prepositional phrases (PP*). A PP may be realized either as an oblique argument or an adjunct, and each choice is marked with an *o*-projection mark for preference ranking. The disjunction under PP in rule (39) is a typical source of optimality marks for sentence (38).

Figure 5: Analysis of *dugg* as an intransitive verb




(39) VP → V ( DP, (↑ OBJ)=↓ ) { PP*, (↑ OBL-TH) = ↓, MARK1 ∈ *o**, ↓ ∈ (↑ ADJUNCT), MARK2 ∈ *o** }

However, in the context of Wolof, the use of the OT mechanism encounters some essential problems. For instance, the use of preference constraints was frequently faced with exceptions and counterexamples. There are still cases where OT chooses the wrong structure. By way of example, let us consider the sentences in (40).

(40) a. *Mu tontu ca laaj ba.*
3sg reply PREP question cl.DIST
"So, (s)he replies to the question."

b. *Mu tontu ca saa sa.*
3sg reply PREP instant cl.DIST
"So, (s)he replies immediately."

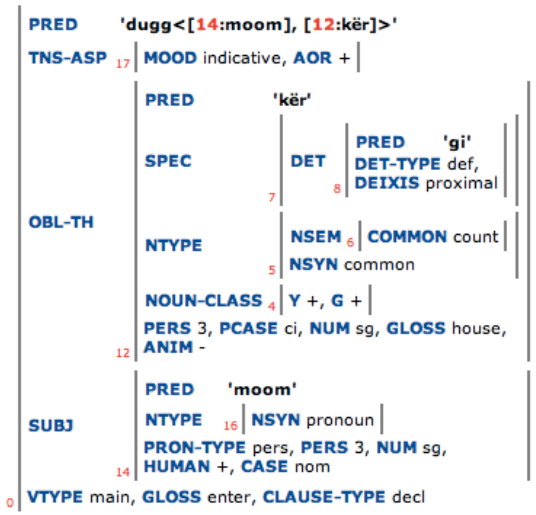



Figure 6:
Analysis of *dugg* as a verb with an oblique argument

These sentences contain the verb *tontu*, which typically selects for the prepositions *ci* and *ca*, meaning, inter alia, ‘on’, ‘to’, and ‘about’ according to the context. Accordingly, the PP *ca laaj ba* in (40a) can be assumed to be subcategorized for by the verb, and therefore bears an argument function (specifically an oblique function) within the clause. As this kind of construction is quite common for this verb, a preference mark for PP obliques was introduced in the Wolof grammar in the early stage of grammar development. This aimed to automatically suppress certain ambiguities due to the adjunct reading. However, as exemplified by (40b), this approach is not successful in all contexts. In (40b), the prepositional phrase *ca saa sa* appears as a modifier rather than an argument of the verb. It modifies the verb *tontu*, but it is not governed by this predicate. Hence, a preference mark for oblique PPs over adjuncts will falsely choose the oblique reading for the PP *ca saa sa* in the sentence (40b).

A similar situation can be observed in English: The preference of oblique PPs over adjuncts may lead to an incorrect analysis of the constituent *for two hours* in the sentence *John waited for two hours*. As Copperman and Segond (1996, pp. 6) pointed out, it is difficult

to find in the literature "a discussion of preferring arguments to adjuncts (via subcategorization frames), which strikes us a valid general preference. Of course, in many cases the question of whether to consider something an argument or an adjunct is no more solved than PP attachment, so this will actually help only in clear cases."

Another disadvantage of this mechanism is that the use of optimality marks requires careful adjusting and experimenting to get certain effects, both in terms of preferences and performance. This reflects the fact that the OT specifications have global interactions and are thus difficult to describe. For instance, for a grammar writer, it may be very difficult to introduce new marks or reorder old ones to get the relatively straightforward outcomes that (s)he is looking for. The indirect consequences of minor adjustments can be hard to understand and predict. Thus, with regard to the Wolof grammar, the idea of using preference marks for PP obliques was abandoned due to the large number of counterexamples.

### 6.4 *Handling coordination ambiguity*

As discussed in Section 3.2.1, Example (18), repeated in (41),[20] also illustrates asyndetic coordination (i.e., coordinated structures without an explicit conjunction). Such structures are frequently encountered in the Wolof data. A typical syntactic feature of these coordinate structures is that they may exhibit *forward conjunction reduction* (Kempen 1991) involving a subject gap: The subject of the left conjunct is omitted from the second clause and understood to be identical to the first clause's subject. In LFG terms, the fact that the second conjunct seems to be missing a subject raises a particular issue with regard to Completeness (Kaplan and Bresnan 1982): All the governable grammatical functions required by the PRED of the f-structure should have a value in the f-structure.

(41) Xale b-i moom doon ree.
child cl-DFP own IPF.PST laugh
"The child owns (something) and was laughing."

To handle this kind of coordination in Wolof, rules like (42) are used. These allow phrases of same constituents to be coordinated. In

[20] The gloss and translation only retain the reading as asyndetic coordination, which is the relevant one for the discussion.

the associated f-structure, the coordinate phrase is represented as a set-valued f-structure. Each of the conjuncts is represented as an element within the set by the functional annotations ↓ ∈ ↑. To solve the Completeness problem, the symmetric analysis with *asymmetric grammaticalised discourse function (GDF) projection* proposed in Frank (2002) for German subject-gap constructions is adopted for Wolof. For example, (42) defines symmetric S coordination in c-structure, with symmetric projection of the conjunct's f-structures in terms of the classical ↓ ∈ ↑ annotations. The annotation (↑ SUBJ) = (↓ SUBJ) defines the first conjunction's subject as the subject of the coordination as a whole.[21] The predicate *e* matches against the empty coordinating conjunction string.[22] The feature (↑ COORD-FORM) specifies the form of the conjunction (e.g., *and* or *or*). In this rule, the form is assumed to be null, since the conjunction is not overtly realized. The annotation *(↑ COORD)* = + indicates that the whole structure is a coordinate phrase.

(42) SCoord → { S: ↓ ∈ ↑ (↑ SUBJ) = (↓ SUBJ);
e: (↑ COORD-FORM) = null (↑ COORD) = +
CWCONJ ∈ o*;
S: ↓ ∈ ↑
}.

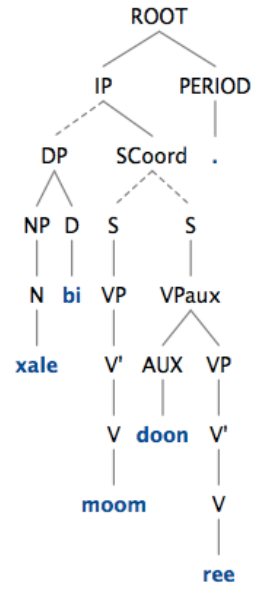
Figure 7 shows the c-structure related to the reading of the sentence (41) as an instance coordination without an explicit conjunction.

In the rule in (42), the annotation *CWCONJ* ∈ *o** (in XLE notation: "CWCONJ $ o::*") says that (i) *CWCONJ* is a member of the OT projection; (ii) such a structure is coordinated without an explicit conjunct; and (iii) it should be dispreferred. Allowing any S, IP or NP constituent to be coordinated with another constituent of the same category without an explicit conjunction poses notorious ambiguity and performance problems: It generates a great number of parse possibilities, and sometimes leads to memory, time-out and coverage problems. Thus, the ambiguity caused by this kind of construction had to be addressed during grammar development.

[21] The (↑ SUBJ) = (↓ SUBJ) equation is only needed for asymmetric case where there is no analysis in which the subject can be construed to be outside of the two conjuncts, in which case normal distribution over sets will take care of this.

[22] The symbol *e* is the "epsilon" symbol for an LFG grammar.

Figure 7:
C-structure analysis of coordinated sentences without an explicit conjunction




Currently, the Wolof grammar handles ambiguity caused by coordination without an explicit conjunction by using the STOPPPOINT mark. For performance reasons, all rules dealing with coordination like (42) (ca. 10 rules) are annotated with the OT dispreference mark *CWCONJ*. In the OT configuration of the grammar, *CWCONJ* is inserted to the left of the STOPPOINT mark to consider this expensive construction only when no other analysis is available.

To measure the impact of using this approach, the grammar was run on the test set with and without the application of STOPPOINT. The test runs reveal that the use of the STOPPOINT mark increases the parsing time by 6%. In fact, the approach does not lead to effective advantages in terms of efficiency because XLE has to parse each sentence in several passes. This explains the slight increase in parsing time. However, this approach pays off very well in terms of ambiguity reduction: The comparison of the number of solutions produced by each run reveals that it reduces the ambiguity rate by a factor of 5–6. However, with this approach, there is also a decrease in the pars-

ing quality: In the test set, about 25% of the desired interpretations (relative to coordination without an explicit conjunction) were also eliminated, causing a drop in f-score of ca. 0.94 points.

### 6.5 *Preserving ambiguity due to pro-drop and impersonal passive*

Unlike constructions discussed in the previous sections, where the main goal was to remove some readings, many syntactically ambiguous utterances can be parsed and assigned ambiguous structures. Section 3.3 discussed constructions that exhibit global ambiguity due to pro-drop and impersonal passive, as illustrated in (19b), repeated in (43).

(43) *Gor na-ñu garab g-i.*
cut.down +F-3pl tree cl-DFP
"They cut down the tree."
"The tree was cut down."

As other types of ambiguity, referential and lexical ambiguity can interact, resulting in global ambiguity. The referential ambiguity is raised by the subject marker *nañu*, which implies that the subject is a non-arbitrary referential subject or an arbitrary subject used as a third person plural person in impersonal passive constructions. As with ambiguity due to subcategorization frames, the phrase structure in (43) is the same in each case, but the difference lies in the form of the semantic predicates. Thus, while the sentence has a single c-structure, its semantic structure is ambiguous.

The f-structure analysis for the first reading is shown in Figure 8. This analysis follows the standard LFG treatment of pro-drop, in which the verb specifies that its subject has the PRED value 'pro'. In the impersonal construction, however, the subject PRED and PRON-TYPE are assumed to be null in order to reanalyze the null-subject construction as an arbitrary reading. The analysis for the second reading is given in Figure 9.

As an instance of global ambiguity, the sentence in (43) is not disambiguated at the parsing level. The solution considered is to leave the decision to users, which are accustomed to resolving many types of ambiguity in texts subconsciously and efficiently using common-sense knowledge. Without the common-sense knowledge that is necessary to resolve this kind of ambiguity, the parser is not expected to know

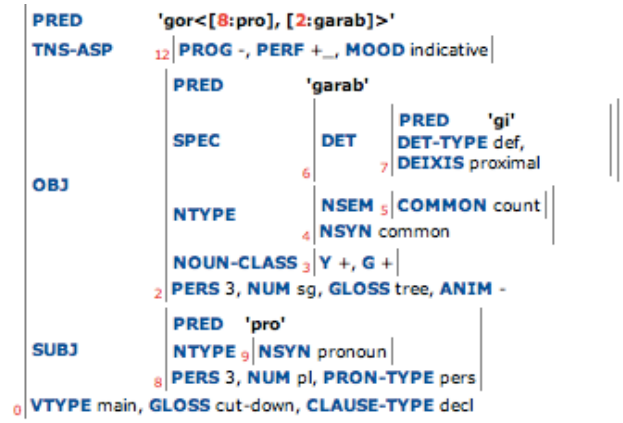



Figure 8:
F-structure analysis of sentence (19b) as an instance of pro-drop constructions

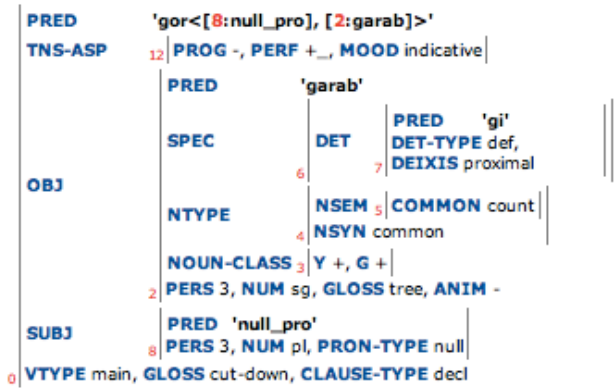



Figure 9:
F-structure of sentence (19b) analyzed as a passivized sentence with a null subject

which of the solutions that it generates can be left ambiguous because they will be disambiguated correctly by the user or because the consequences of misinterpretation are trivial, and which will be genuinely problematic (Chantree 2004). Consequently, automatic resolution of global ambiguity is mostly not desired. It can be very beneficial to allow this kind of ambiguities to remain in the text and to be resolved by users at a later stage (i.e., in post-parsing). For such a task, there are grammar engineering tools available which provide a representation of a set of ambiguous f-structures in a single, packed structure, allow-

ing (non-expert) users to locate specific solutions among the output set straightforwardly and efficiently (e.g., using discriminants).

## 7 GRAMMAR ENGINEERING AND AMBIGUITY

### 7.1 *Ambiguity packing in XLE*

To facilitate ambiguity management, XLE provides a built-in utility for grouping and displaying packed representations of the alternative solutions (King *et al.* 2004). The utility is based on an efficient algorithm for contexted constraint satisfaction that processes ambiguities in a chart-like packed representation (Maxwell III and Kaplan 1996).

Consider Example (43) discussed in Section 6.5. As this ambiguity is global and linguistically appropriate, it will normally be computed and preserved. Accordingly, in the lexicon, the entry for the form *nañu* is provided with two semantic PREDs encoded as disjunctive statements to allow for the two readings of this word. These alternative solutions logically lead to at least two different f-structures. However, with the XLE built-in algorithm for contexted constraint satisfaction, such disjunctive facts are not compiled out and duplicated. The algorithm rather produces a representation as a set of the ambiguous f-structures in a single, packed f-structure, also called f-structure chart, as Figure 10 illustrates.

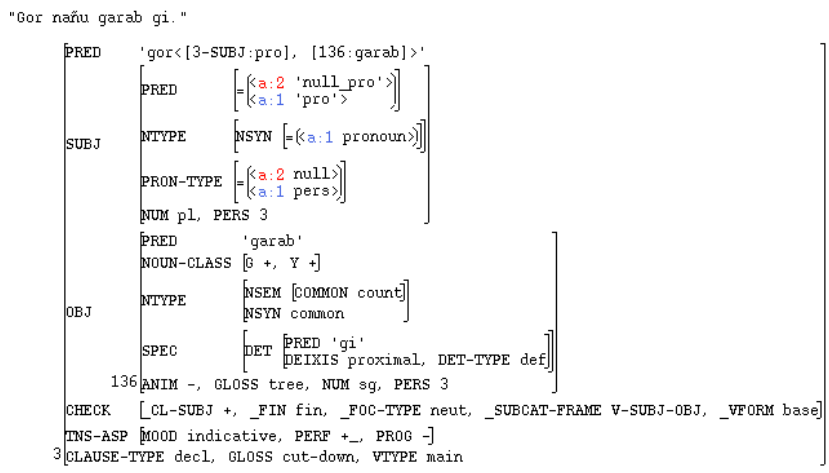



Figure 10: F-structure chart for packed ambiguities in XLE

The f-structure chart window provides a list of choices that are caused by alternative solutions. Hence, this sentence has two analyses,

identical except for the values of the PRED (which may be ‘pro’ or ’null_pro’), pronoun and noun type features. In the packed f-structure chart, attribute-values are indexed with their corresponding context variables, meaning that the two values are displayed as alternatives, labeled with indices a:1 and a:2.

This XLE tool for ambiguity management helps grammar developers to determine the source of the multiple solutions produced by the parser. As Attia (2008, pp. 221) pointed out, “grouping the solutions in packed representations can effectively speed up the process of detection and revision”. For instance, given that the choices in the window in Figure 10 are active, the user can click on a choice and have a solution corresponding to it displayed in the c-structure and f-structure windows. Thus, the use of this facility can avoid the need for the grammar developers to search through the parse forest by examining one solution after the other.

Equally important, the tool provides grammar writers with information on the existence of spurious ambiguities. For instance, vacuous ambiguity of two f-structures, for example, resulting from duplicate lexicon entries for the same word, appears in a specific form (namely, as blank choices) indicating that there is a spurious ambiguity in the grammar with respect to the given sentence. One of the best ways to avoid such vacuous ambiguities is to check the grammar carefully and to make disjunctions exclusive. The elimination of spurious ambiguities proves to be a very effective mechanism for increasing parsing efficiency.

### 7.2 *Removing spurious ambiguities*

Spurious ambiguities as duplicated solutions may arise from different sources, including the morphology, the lexicon, the c-structure and the f-structure. For instance, c-structures may be duplicated if there are two entries under a word for the same category. This particular problem may also arise when there is no obvious disjunction (Crouch *et al.* 2013).

Disjunctions are the alternative paths that a rule can take. “While disjunctive statements of linguistic constraints allow for a transparent and modular specification of linguistic generalizations, the resolution of disjunctive feature constraint systems is expensive, in the worst case exponential” (King *et al.* 2000, pp. 7). If disjunctions are not clearly

defined in order to be mutually exclusive, they can lead to overgeneration. As the sentence length grows, spurious ambiguity can cause an exponential growth in the number of generated solutions.

Thus, grammar writers need to investigate methods of eliminating spurious ambiguities, for example, by verifying that disjunctions in the grammar are mutually exclusive. For example, if an NP's f-description contains the disjuncts in (44), then this NP is required to receive a nominative case value or a third person value. However, the disjunction in (44) is not mutually exclusive, since both can be satisfied at the same time. A good way to avoid spurious ambiguity in this case is to make the disjunction explicit and mutually exclusive, as shown in (45). While in the first disjunct in (45) the attribute person must have a value other than 3, hence the annotation (↑ PERS) ~= 3, in the second disjunct in this example a third person value is required; thus the two disjuncts cannot be satisfied at the same time.

(44) {(↑ CASE) $=_c$ nom | (↑ PERS) $=_c$ 3 }

(45) {(↑ CASE) $=_c$ nom (↑ PERS) ~= 3 | (↑ PERS) $=_c$ 3 }

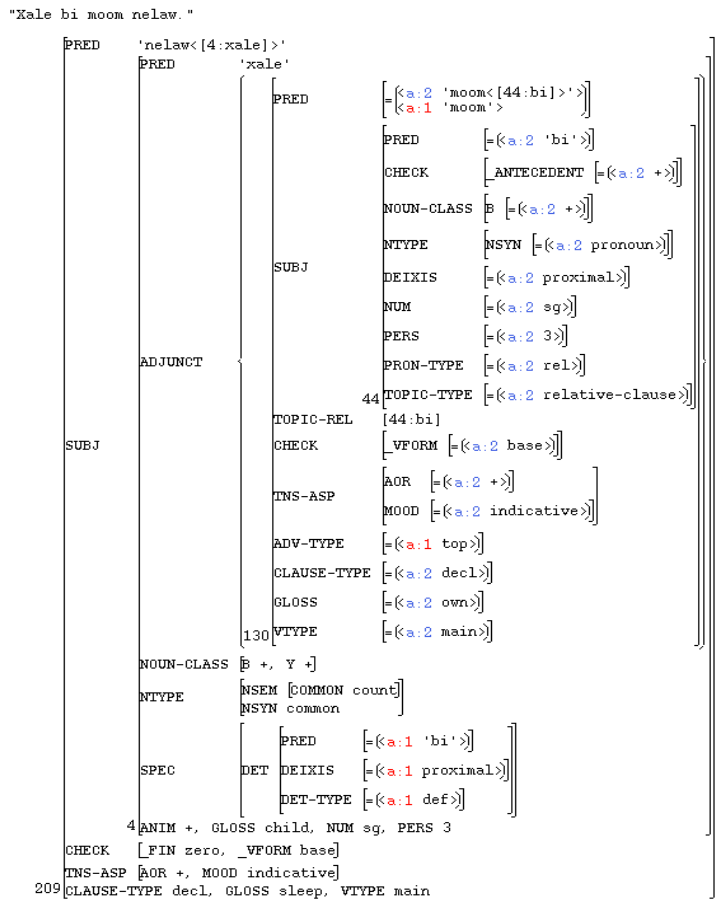
Accordingly, I have thoroughly checked the rules, templates (including verb subcategorization frames) and lexical entries in the Wolof grammar, in order to avoid as many duplicate solutions as possible. This careful review and redesign of the grammar has led to a considerable reduction of spurious ambiguities. Though it might seem like a small detail, removing spurious ambiguities can lead to a great improvement in parsing efficiency: The decrease in parse time observed after taking this measure was more than 50%. Attia (2008, pp. 219) has made a similar observation with respect to the Arabic Grammar, stating that "changing the way a rule was written to avoid a non-exclusive disjunction led to a huge reduction in parse time by 68%. The number of subtrees was reduced then by approximately 10%." These common observations clearly show the effectiveness of spurious ambiguity elimination by making disjunctions mutually exclusive. However, the sources of such ambiguities (e.g., the fact that disjunctions such as the one in Example (44) are not mutually exclusive) are not really obvious for grammar developers and deserve consideration in grammar writing.

## 8 DISAMBIGUATION WITH DISCRIMINANTS

The facility for packing ambiguity provided by XLE is easy to use for disambiguation when there are only a few choices. However, in some contexts, there are many choices. For some other inputs, more than hundreds of analyses are produced. Such a context is illustrated by Example (46) and its related f-structure in Figure 11.

(46) *Xale bi moom nelaw.*
child the TOP.ADV sleep
"The child, for his part, sleeps."

Figure 11: Partial f-structure chart for sentence (46)



When dealing with sentences with many choices such as (46), using this facility requires expert competence in using XLE and detailed knowledge of the grammar (Rosén *et al.* 2005). Consequently, in addition to packing ambiguities, Rosén *et al.* (2005) have implemented discriminants for LFG to facilitate the disambiguation task.

Discriminants are defined as small independent choices which interact to create dozens of analyses (Carter 1997). The idea is based on Carter's (1997) argument "that disambiguation may be achieved quickly and without expert competence if it is based on elementary linguistic properties which the disambiguator may accept or reject independently of other properties" (Rosén *et al.* 2005, pp. 378). On this basis, Rosén *et al.* (2005) implement discriminants for LFG-based parsers, defining a discriminant in LFG terms as "any local property of a c-structure or f-structure that not all analyses share."

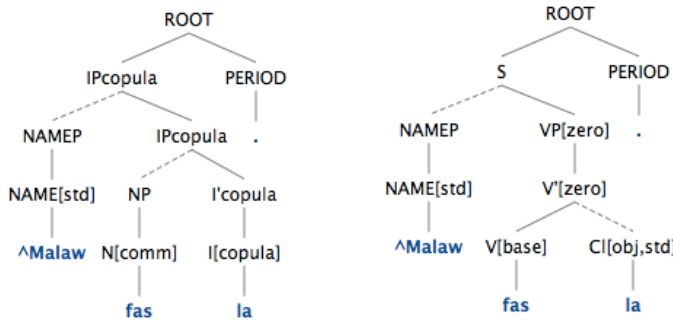
There are four major types of discriminants for LFG grammars (Rosén *et al.* 2007): lexical, morphological, c-structure and f-structure discriminants. Any given discriminant can induce a binary partition on the choice space. The selection of a discriminant (or its complement) amounts to the selection of one of the two partition elements, reducing the choice space accordingly.

The discriminant-based approach provides efficient and elegant support for LFG parse disambiguation. For instance, the Wolof treebank has been established by running test suites through the grammar. To disambiguate the outputs of the parser and to measure the parsing quality, the set of solutions returned by the parser must be manually reviewed, as no gold standard data are available for the language. However, the Wolof parser, like most parsing systems, produced a great number of solutions (tens or hundreds, sometimes even thousands of solutions). In such cases, reviewing the output by hand to see if the intended reading is in the set of the produced parses becomes a time-consuming, tedious and impractical task.

Therefore, I used a semi-automatic, incremental parsebanking approach based on lexical, morphological, c-structure and f-structure discriminants, in order to disambiguate the Wolof data in an efficient and elegant way. By way of illustration, let us consider Example (47). This sentence is associated with two c-structures displayed in Figure 12.

(47) *Malaw* ***fas*** *la.*
Malaw horse/amulet/to.tie COP.3SG/2SG.OBJ
"Malaw is a horse/an amulet." / "Malaw tied you."

Figure 12: Two possible c-structures for the sentence in (47)

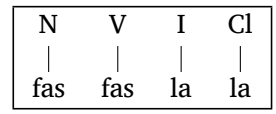



In this sentence, the word form *fas* may be either a verb (V) or a noun (N). Likewise, *la* may be a copular verb (I) or an object clitic (Cl). Because of this, there are two quite different c-structures, as shown in Figure 12. However, choosing the lexical category of either of these words is sufficient to determine which c-structure is the intended one; thus examining these c-structures is no longer necessary. Figure 13 illustrates *lexical discriminants* for the ambiguous words in (47). The traditional part of speech (e.g., N, V, I, Cl) is the lexical category specified in the discriminant. The relevant subtrees containing preterminal and terminal nodes for Example (47) are shown in Figure 14.

Figure 13: Representation of lexical discriminants for *fas* and *la*

| |
|---|
| '*fas*': V |
| '*fas*': N |
| '*la*': I |
| '*la*': Cl |

Figure 14: Subtrees defining lexical discriminants for Example (47)




In (47), the word *fas* is ambiguous between different forms within the same POS (N). It can either mean 'horse' and therefore fits to

the noun class *w* or 'amulet' which belongs to the *g* class. This morphological ambiguity evidences the fact that, in some cases, lexical discriminants are not sufficient for disambiguation. For this reason, Rosén *et al.* (2007) decided to further define a morphological discriminant as "a word with the tags it receives from morphological preprocessing." A *morphological discriminant* for *fas* is illustrated in Figure 15.[23]

| *fas* + Noun + Common + g + Inanim |
|---|
| *fas* + Noun + Common + w + Anim |

Figure 15:
Morphological discriminants for *fas*

Discriminants can be displayed along c- and f-structures using the XLE Web Interface (XLE-Web),[24] as shown in Figure 16.

Originally developed in the TREPIL project (Rosén *et al.* 2009) and now in use for many of the ParGram grammars, the interface is a web-based tool for interactive sentence analysis with XLE. It allows to visualize the mapping from c- to f-structure, and to compactly display packed representations that combine the c- and f-structures of all analyses of a given parse into one c- and one f-structure graph.

Discriminants are presented in a user-friendly form with a sentence and all the parses identified by the parser. In Figure 16, the XLE-Web interface shows possible c- and f-structures for the sentence in (47) as well as lexical, morphological, and syntactic features, allowing binary choices for efficiently selecting the intended discriminant. This example has morphological and lexical discriminants that are reflected in the f-structure. The discriminants are the different values of the NOUN-CLASS, ANIM and GLOSS features. When a discriminant is selected, parses not consistent with that selection are removed from the choice space (and suppressed in the display). Discriminants are not completely independent. Some discriminants are redundant and others eliminate dependent discriminants when selected. Table 5 displays LFG discriminant statistics for the Wolof treebank.

[23] Note that this example refers to the initial analysis of Wolof noun classes prior to the underspecification approach discussed in Section 5.1.

[24] See `http://iness.uib.no/iness/`.

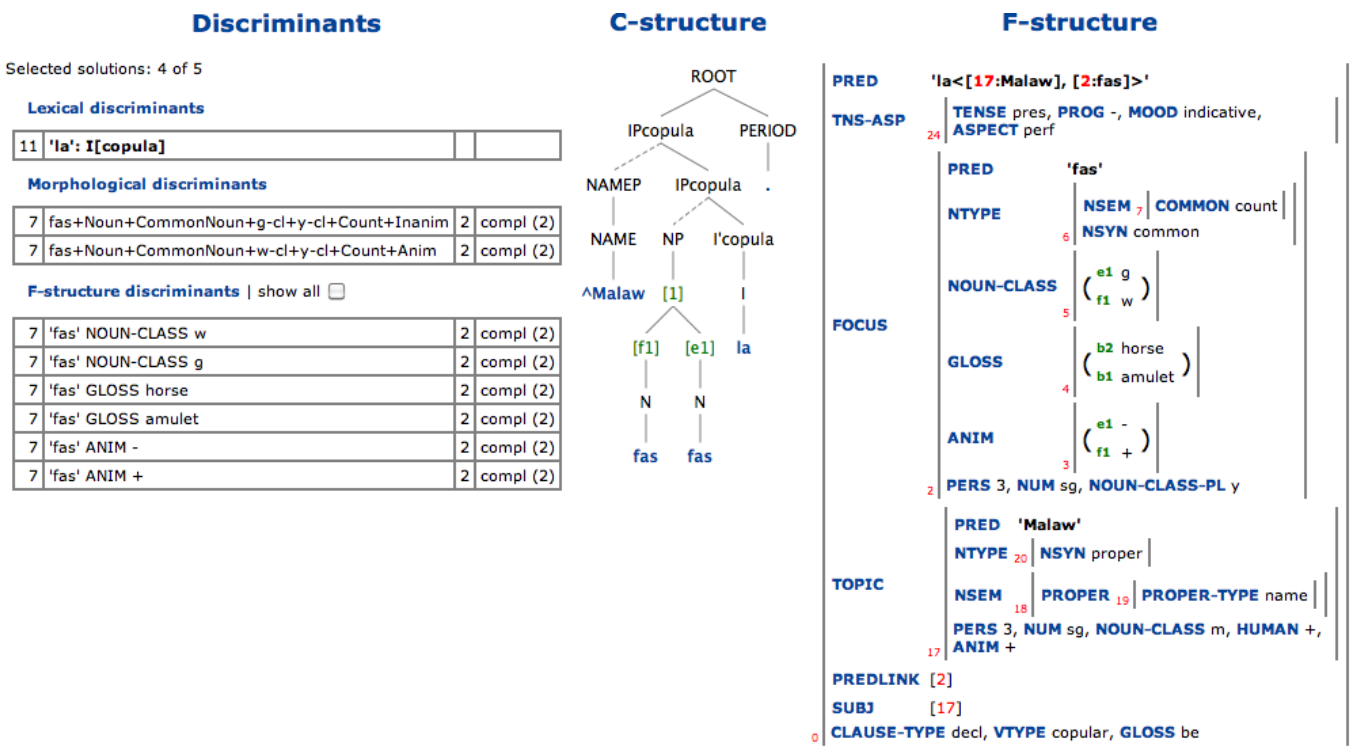



Figure 16: XLE-Web interface showing discriminants

| Discriminant Type | Frequency |
|---|---|
| M: Morphological discriminant | 522 |
| L: Lexical discriminant | 1432 |
| C(R): C-structure rule discriminant | 131 |
| C(C): C-structure constituent discriminant | 568 |
| F: F-structure discriminant | 606 |
| total | 3259 |

Table 5:
LFG discriminants statistics

## 9 CONCLUSION

This work shows that natural languages, with a particular focus on Wolof, are rich in ambiguities of many kinds. It also shows that the wide range of possible interpretations of natural languages and the interaction between the different ambiguity types pose a particular challenge for large-scale, linguistically motivated grammars. In the context of Wolof, the most productive sources of ambiguity in the grammar include noun class syncretism, the use of coverbal ideophones, lexically ambiguous words, lexical ambiguity due to subcategorization frames, structural ambiguity, coordination ambiguity, and ambiguity between pro-drop and impersonal passive constructions. Accordingly, I explored several ambiguity management approaches at various parsing levels. This includes systematic ways of dealing with ambiguity, CG-based disambiguation, c-structure pruning, the application of OT constraints, packing ambiguities and discriminant-based disambiguation.

Systematic disambiguation approaches involve classical ways of underspecification. With the assumption that Wolof nouns typically show no class distinction and often have forms that can be attributed 'indeterminately' to different noun classes, I applied an underspecification approach based on feature indeterminacy to the Wolof noun class system. Following this analysis, Wolof nouns were assigned a feature structure containing a noun class attribute whose value allows specification by means of a separate Boolean-valued attribute. The proposed approach correctly identifies the linguistic aspects triggered by the noun class attribute, allowing to substantially reduce both ambiguity and parse time.

Likewise, ambiguity caused by Wolof ideophones were dealt with in a systematic way. For this word class, I introduced a special c-

structure category based on the main assumption that ideophones behave like verb particles. Using lexical specification, collocational verbs that subcategorize for ideophones were then constrained to specify the lexical form of the particles they select for. Following on from this, a functional template was used to concatenate the arguments represented by the collocational verb and the ideophonic particle. In addition, optimality marks were used to state a preference for the ideophonic reading, when ideophones co-occur with the collocational verb. This helped to control ambiguity caused by the collocational verb and ideophones, resulting in a substantial improvement in parse efficiency.

Disambiguation at the pre-parsing stage includes handling morphological and lexical ambiguities using CG-based approaches. The application of these approaches showed that, with a modest number of CG rules, the average number of readings per token and therefore the large number of lexical and morphological ambiguities can be reduced significantly. Also, the CG-based model proved to be very useful when dealing with less-resourced languages, as it avoids the requirement for a large training corpus. Equally interesting, the CG-based techniques were combined with the c-structure pruning mechanism to tackle ambiguities that arise both at the pre-parsing and at the parsing stages. The application of the c-structure pruning mechanism led to a considerable reduction of structural ambiguity in the grammar. It caused a great decrease of the number of the c-structures built in the XLE chart parser, allowing for a significant improvement in parsing efficiency. In terms of ambiguity reduction and efficiency, techniques based on CG and c-structure pruning proved to be the most effective ones. The experiment results show that the combination of c-structure pruning with CG-based disambiguation can greatly reduce the ambiguity rate by ca. 80% and increase parsing efficiency by 58%, however at the expense of the accuracy of the overall system. The parsing quality decreased by about 3.62 points in f-score. With more training data for c-structure pruning, better results could be expected.

To provide a high-level comparison of disambiguation options, this work has also experimented with optimality marking. The mechanism is used to manage ambiguity caused by ideophonic expressions, asyndetic coordination and verb subcategorization frames. Although OT filtering was originally intended to be effective in filtering syn-

tactic ambiguity, the current findings suggest that the preference constraints are frequently faced with exceptions and counterexamples. Optimality marking seems to have only occasional effects, for instance when disambiguating clear cases like ideophonic expressions or when taking advantage of STOPPOINT effects in certain cases. The results show that, with constructions like asyndetic coordination, using the STOPPOINT mark can prove very beneficial in terms of ambiguity reduction, but also eliminates a substantial number of desired interpretations and decreases parsing efficiency. In the Wolof grammar, the latter approach is currently used as a default option, selected from the explored possible approaches failing an optimal solution.

In addition, this work has discussed grammar engineering utilities that facilitate ambiguity management at the parsing and post-parsing levels. It has shown that XLE provides useful built-in tools that allow for automatic packing of representations of the alternative parse solutions. Such tools are valuable in dealing with global ambiguities where appropriate readings of a construction need to be preserved. For large-scale grammars, one particularly interesting feature of these tools is that they provide grammar writers with very useful information on spurious ambiguities. As the Wolof case has shown, it is crucially important to develop strategies for avoiding vacuous ambiguities, including ways of mending non-exclusive disjunctions originating from duplicate solutions. With a relatively simple disambiguation technique consisting in preventing spurious explorations of the grammar, the performance of the parser could significantly be improved.

More interestingly, the paper shows how efficient and elegant LFG parse disambiguation can be achieved using discriminants. Presented in a user-friendly way, discriminants are easy for humans to judge and are prominent in the XLE-web interface display. The user can disambiguate the sentence by selecting or rejecting discriminants and thereby retaining or rejecting sets of corresponding analyses. The efficiency of this method, as compared to presenting all the full analyses to the user, can be appreciated from the fact that a combination of a small number of local ambiguities can result in a large number of analyses. Applied on the Wolof LFG treebank, this method allowed to efficiently deal with ambiguity using lexical, morphological, c-structure and f-structure discriminants. As with CG, this disambiguation method is particularly attractive, as it does not require much training data.

Table 6: Comparison of the impact of five different disambiguation methods on the Wolof test data

| Disambiguation method | Ambiguity reduction | Parse time | Drop in parsing accuracy |
|---|---|---|---|
| Underspecification | ≈ 8% | reduced by 4% | None |
| Lexical specification | ≈ 4% | reduced by 16% | None |
| CG pre-filtering | ≈ 77% | reduced by 30% | 2.5 |
| C-structure pruning | ≈ 72% | reduced by 36% | 0.97 |
| Optimality marking | ≈ 80% | increased by 6% | 0.94 |

As noted earlier, the various disambiguation methods were applied on different parsing levels and parser versions. The methods have interactions which are very difficult to control systematically. In the same way, it is very tedious to measure all combinations of all techniques. Table 6 gives estimates of the gain or drop that would result from adding some of the techniques.

The table shows the impact of using underspecification, lexical specification for coverbal ideophones, CG-based disambiguation, c-structure pruning and optimality marking. Disambiguation techniques based on the formal encoding of noun class indeterminacy via underspecification and lexical specification apply alternative descriptive devices that reduced both ambiguity and parse time, but otherwise leave the space of analyses unchanged (i.e., they did not lead to a drop in parsing accuracy). CG pre-filtering greatly reduced both ambiguity and parse time, but caused a drop of 2.5 points in f-score. Likewise, with c-structure pruning, the ambiguity rate as well as the parse time were reduced significantly, but the accuracy also decreased by about 0.97 points. Finally, with optimality marking, ambiguity dropped significantly, but there was a slight increase in parse time and a substantial drop in parsing accuracy.

## 10 ACKNOWLEDGEMENTS

This research has received support from the EC under FP7, Marie Curie Actions SP3-People-ITN 238405 (CLARA).

## REFERENCES

Mohammed ATTIA (2008), *Handling Arabic morphological and syntactic ambiguity within the LFG framework with a view to machine translation*, Ph.D. thesis, University of Manchester.

Mariyaama BA (2007), *Bataaxal bu gudde nii (So long a letter)*, Nouvelles Editions Africaines du Sénégal (NEAS), Dakar, Sénégal.

Matthew BAERMAN, Dunstan BROWN, and Greville G. CORBETT (2005), *The syntax-morphology interface: A study of syncretism*, Cambridge University Press.

Marlyse BAPTISTA (1995), On the nature of pro-drop in Capeverdean Creole, Technical report, Harvard Working Papers in Linguistics.

Eckhard BICK (2000), *The parsing system "Palavras": Automatic grammatical analysis of Portuguese in a Constraint Grammar framework*, Aarhus University Press, Denmark.

Eckhard BICK (2009), *Basic Constraint Grammar tutorial for CG-3 (Vislcg3)*, Southern Denmark University, Copenhagen, Denmark, `http://beta.visl.sdu.dk/cg3_howto.pdf`.

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

Miriam BUTT, Tracy Holloway KING, María-Eugenia NIÑO, and Frédérique SEGOND (1999), *A grammar writer's cookbook*, CSLI Publications, Stanford, CA, USA.

Aoife CAHILL, Tracy Holloway KING, and John T. MAXWELL III (2007), Pruning the search space of a hand-crafted parsing system with a probabilistic parser, in *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, pp. 65–72, Prague, Czech Republic.

Aoife CAHILL, John T. MAXWELL III, Paul MEURER, Christian ROHRER, and Victoria ROSÉN (2008), Speeding up LFG parsing using c-structure pruning, in *Proceedings of the Workshop on Grammar Engineering Across Frameworks*, pp. 33–40, Manchester, UK.

David CARTER (1997), The TreeBanker. A tool for supervised training of parsed corpora, in *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, pp. 9–15, Madrid, Spain.

Francis CHANTREE (2004), Ambiguity management in Natural Language Generation, in *Proceedings of the Seventh Annual CLUK Research Colloquium*, pp. 23–28, Birmingham, UK.

Noam CHOMSKY (1981), *Lectures on Government and Binding*, Foris, Dordrecht, The Netherlands.

Mamadou CISSÉ (1994), *Contes Wolof modernes (Modern Wolof tales)*, L'harmattan, Paris, France.

Ann COPESTAKE and Dan FLICKINGER (2000), An open source grammar development environment and broad-coverage English grammar using HPSG, in *Proceedings of the Second International Conference on Language Resources and Evaluation*, Athens, Greece.

Max COPPERMAN and Frédérique SEGOND (1996), Computational grammars and ambiguity: the bare bones of the situation, in *Proceedings of the LFG '96 Conference*, Grenoble, France.

Denis CREISSELS (2001), Setswana ideophones as uninflected predicative lexemes, *Typological Studies in Language*, 44:75–86.

Dick CROUCH, Mary DALRYMPLE, Ron KAPLAN, Tracy Holloway KING, John T. Maxwell III, and Paula NEWMAN (2013), XLE documentation, On-line, Palo Alto Research Center (PARC), `http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html`.

Berthold CRYSMANN (2005), Syncretism in German: a unified approach to underspecification, indeterminacy, and likeness of case, in *Proceedings of the 12th International Conference on Head-Driven Phrase Structure Grammar*, pp. 91–107, Lisbon, Portugal.

Mary DALRYMPLE (2001), *Lexical-Functional Grammar*, volume 34 of *Syntax and Semantics*, Emerald Group Publishing Limited, Bingley, West Yorkshire, UK.

Mary DALRYMPLE and Ronald M. KAPLAN (1997), A set-based approach to feature resolution, in *Proceedings of the LFG '97 Conference*, San Diego, CA, USA.

Mary DALRYMPLE, Tracy Holloway KING, and Louisa SADLER (2009), Indeterminacy by underspecification, *Journal of Linguistics*, 45(01):31–68.

Tino DIDRIKSEN (2003), Constraint Grammar manual, `http://beta.visl.sdu.dk/cg3/vislcg3.pdf`, apS, GrammarSoft.

Cheikh M. Bamba DIONE (2012a), An LFG approach to Wolof cleft constructions, in *Proceedings of the LFG '12 Conference*, pp. 157–176, Stanford, CA, USA.

Cheikh M. Bamba DIONE (2012b), A morphological analyzer for Wolof using finite-state techniques, in *Proceedings of the Eigth International Conference on Language Resources and Evaluation*, Istanbul, Turkey.

Cheikh M. Bamba DIONE (2013a), Handling Wolof clitics in LFG, in Christine Meklenborg SALVESEN and Hans P. HELLAND, editors, *Challenging Clitics*, pp. 87–118, John Benjamins, Amsterdam, The Netherlands.

Cheikh M. Bamba DIONE (2013b), Valency change and complex predicates in Wolof: an LFG account, in *Proceedings of the LFG '13 Conference*, pp. 232–252, Stanford, CA, USA.

Cheikh M. Bamba DIONE (2014), Pruning the search space of the Wolof LFG grammar using a probabilistic and a constraint grammar parser, in *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, Reykjavik, Iceland, ISBN 978-2-9517408-8-4.

Clement M. DOKE (1935), *Bantu Linguistic Terminology*, Longmans, Green and Company, London, England.

Helge DYVIK (2000), Nødvendige noder i norsk. Grunntrekk i en leksikalsk-funksjonell beskrivelse av norsk syntaks. [Necessary nodes in Norwegian. Basic properties of a lexical-functional description of Norwegian syntax.], in Øivin ANDERSEN, Kjersti FLØTTUM, and Torodd KINN, editors, *Menneske, språk og felleskap*, Novus forlag, Oslo, Norway.

Dan FLICKINGER (2000), On building a more efficient grammar by exploiting types, *Natural Language Engineering*, 6(1):15–28.

Anette FRANK (2002), A (discourse) functional analysis of asymmetric coordination, in *Proceedings of the LFG '02 Conference*, pp. 174–196, Athens, Greece.

Anette FRANK, Tracy Holloway KING, Jonas KUHN, and John T. MAXWELL III (1998), Optimality Theory style constraint ranking in large-scale LFG grammars, in *Proceedings of the LFG '98 Conference*, Stanford, CA, USA.

Anette FRANK, Tracy Holloway KING, Jonas KUHN, and John T. MAXWELL III (2001), Optimality Theory style constraint ranking in large-scale LFG grammars, in Peter SELLS, editor, *Formal and Empirical Issues in Optimality Theoretic Syntax*, pp. 367–398, CSLI Publications, Stanford, CA, USA.

Nataali Dominik GARROS, editor (1997), *Bukkeek "perigam" bu xonq: teeñ yi (Hyena and its red wig: the lice)*, Dakar, Senegal: SIL; Paris, France: EDICEF, dr. Moren ak mbootayu "xale dimbale xale". trad. du français en wolof par Momar Touré.

Gerald GAZDAR and Chris MELLISH (1989), *Natural Language Processing in {LISP}*, Addison-Wesley, Boston, MA, USA.

Talmy GIVÓN (1979), *On understanding grammar*, Academic Press, New York, NY, USA.

Pascual Cantos GÓMEZ (1996), *Lexical ambiguity, dictionaries and corpora*, Services de Publicaciones, Universidad de Murcia, Spain.

Ron KAPLAN and Joan BRESNAN (1982), Lexical-Functional Grammar: a formal system for grammatical representation, in Joan BRESNAN, editor, *The Mental Representation of Grammatical Relations*, pp. 173–281, MIT Press, Cambridge, MA, USA.

Ronald M. KAPLAN, John T. MAXWELL III, Tracy Holloway KING, and Richard CROUCH (2004), Integrating finite-state technology with deep LFG grammars, in *Proceedings of the ESSLLI 2004 Workshop on Combining Shallow and Deep Processing for NLP*, Nancy, France.

Fred KARLSSON (1990), Constraint Grammar as a framework for parsing running text, in *Proceedings of the 13th Conference on Computational Linguistics*, pp. 168–173, Helsinki, Finland.

Gerard KEMPEN (1991), Conjunction reduction and gapping in clause-level coordination: an inheritance-based approach, *Computational Intelligence*, 7(4):357–360.

Tracy Holloway KING, Stefanie DIPPER, Anette FRANK, Jonas KUHN, and John MAXWELL (2000), Ambiguity management in grammar writing, in *Proceedings of the ESSLLI 2000 Workshop on Linguistic Theory and Grammar Implementation*, pp. 5–19, Birmingham, UK.

Tracy Holloway KING, Stefanie DIPPER, Anette FRANK, Jonas KUHN, and John T. MAXWELL III (2004), Ambiguity management in grammar writing, *Research on Language and Computation*, 2(2):259–280.

Ekaterini KLEPOUSNIOTOU (2002), The processing of lexical ambiguity: homonymy and polysemy in the mental lexicon, *Brain and Language*, 81(1):205–223.

Nobo KOMAGATA (2004), A solution to the spurious ambiguity problem for practical Combinatory Categorial Grammar parsers, *Computer Speech & Language*, 18(1):91–103.

Maryellen C. MACDONALD, Neal J. PEARLMUTTER, and Mark S. SEIDENBERG (1994), Lexical nature of syntactic ambiguity resolution, *Psychological Review*, 101(4):676–703.

Christopher D. MANNING and Hinrich SCHÜTZE (1999), *Foundations of Statistical Natural Language Processing*, MIT Press, Cambridge, MA, USA.

William MARTIN, Kenneth CHURCH, and Ramesh PATIL (1987), *Preliminary analysis of a breadth-first parsing algorithm: theoretical and experimental results*, Springer, Berlin/Heidelberg, Germany.

John T. MAXWELL and Ronald M. KAPLAN (1993), The interface between phrasal and functional constraints, *Computational Linguistics*, 19(4):571–590.

John T. MAXWELL III and Ronald M. KAPLAN (1996), Unification-based parsers that automatically take advantage of context-freeness, in *Proceedings of the LFG '96 Conference*, Grenoble, France.

Fiona MCLAUGHLIN (2004), Is there an adjective class in Wolof?, in Robert M. W. DIXON and Alexandra Y. AIKHENVALD, editors, *Adjective Classes: A Cross-Linguistic Typology*, pp. 242–262, Oxford University Press, Oxford, UK.

Fiona MCLAUGHLIN (2010), Noun classification in Wolof: When affixes are not renewed, *Studies in African Linguistics*, 26(1):1–28.

Marjorie J. MCSHANE (2005), *A theory of ellipsis*, Oxford University Press, Oxford, UK.

Alan PRINCE and Paul SMOLENSKY (1993), Optimality Theory: constraint interaction in Generative Grammar, Technical report, Rutgers University Center for Cognitive Science, Cambridge, MA, USA.

Stefan RIEZLER, Tracy Holloway KING, Ronald M. KAPLAN, Richard CROUCH, John T. MAXWELL III, and Mark JOHNSON (2002), Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques, in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 271–278, Philadephia, PA, USA.

Victoria ROSÉN, Paul MEURER, and Koenraad DE SMEDT (2005), Constructing a parsed corpus with a large LFG grammar, in *Proceedings of the LFG '05 Conference*, pp. 371–387, Stanford, CA, USA.

Victoria ROSÉN, Paul MEURER, and Koenraad DE SMEDT (2007), Designing and implementing discriminants for LFG grammars, in *Proceedings of the LFG '07 Conference*, pp. 397–417, Stanford, CA, USA.

Victoria ROSÉN, Paul MEURER, and Koenraad DE SMEDT (2009), LFG Parsebanker: a toolkit for building and searching a treebank as a parsed corpus, in *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories*, pp. 127–133, Utrecht, The Netherlands.

David J. SAPIR (1971), West Atlantic: an inventory of the languages, their noun class systems and consonant alternation, *Current Trends in Linguistics*, 7(1):43–112.

William H. TORRENCE (2005), *On the distribution of complementizers in Wolof*, Ph.D. thesis, University of California, Los Angeles, CA, USA.

Hans USZKOREIT, Rolf BACKOFEN, Stephan BUSEMANN, Abdel K. DIAGNE, Elizabeth A. HINKLEMAN, Walter KASPER, Bernd KIEFER, Hans-Ulrich KRIEGER, Klaus NETTER, Günter NEUMANN, *et al.* (1994), DISCO: an HPSG-based NLP system and its application for appointment scheduling, in *Proceedings of the 15th Conference on Computational Linguistics*, pp. 436–440, Kyoto, Japan.

Erhard F. K. VOELTZ and Christa KILIAN-HATZ, editors (2001), *Ideophones*, Typological Studies in Language, John Benjamins, Amsterdam, The Netherlands.

Sylvie VOISIN-NOUGUIER (2002), *Relations entre fonctions syntaxiques et fonctions sémantiques en Wolof (Relations between syntactic functions and semantic functions in Wolof)*, Ph.D. thesis, Université Lumière (Lyon 2), Lyon, France.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*
http://creativecommons.org/licenses/by/3.0/


# Computational modelling of Yorùbá numerals in a number-to-text conversion system

*Olúgbénga O. Akinadé and Ọdẹ́túnjí A. Ọdéjọbí*
Computing and Intelligent Systems Research Group
Department of Computer Science and Engineering
Ọbáfẹ́mi Awólọ́wọ̀ University
Ilé-Ifẹ̀, Nigeria

## ABSTRACT

In this paper, we examine the processes underlying the Yorùbá numeral system and describe a computational system that is capable of converting cardinal numbers to their equivalent Standard Yorùbá number names. First, we studied the mathematical and linguistic basis of the Yorùbá numeral system so as to formalise its arithmetic and syntactic procedures. Next, the process involved in formulating a Context-Free Grammar (CFG) to capture the structure of the Yorùbá numeral system was highlighted. Thereafter, the model was reduced into a set of computer programs to implement the numerical to lexical conversion process. System evaluation was done by ranking the output from the software and comparing the output with the representations given by a group of Yorùbá native speakers. The result showed that the system gave correct representation for numbers and produced a recall of 100% with respect to the collected corpus. Our future study is focused on developing a text normalisation system that will produce number names for other numerical expressions such as ordinal numbers, date, time, money, ratio, etc. in Yorùbá text.

*Keywords: Analysis of numerals, Yorùbá numerals, numbers to text, text normalisation*

## 1 INTRODUCTION

The use of numbers and their power in capturing concepts makes them indispensable in effective communication (Goyvaerts 1980). In any society, the use of numbers is firmly anchored to numerous beliefs and perceived usefulness of the significant philosophy underlying numerical messages (Abímbọ́lá 1977). In fact, key advancement in civilisation can be traced to the conception, invention, representation, and manipulation of numbers to facilitate accurate rendering of measurable objects. This has made the use of numbers an important tool within the society, where it is used in trade, cosmology, mathematics, divination, music, medicine, etc. Early cultures devised various means of number representation, which include body/finger counting (Zaslavsky 1973; Saxe 1981), object counting, Egyptian numerals, Babylonian numerals, Greek numerals, Chinese numerals, Roman numerals, Mayan numerals, Hindu-Arabic numerals, etc. The Hindu-Arabic numeral system, which is considered to be the greatest mathematical discovery (Bailey and Borwein 2011), is still the most commonly used symbolic representation of numbers due to its simplicity and the fact that it requires little memorisation to represent practically any number.

Naming numbers in human languages requires various mathematical and linguistic processes. For example, the number 74 is represented as 70 ($7 \times 10$) increased by 4 in English, whereas it is represented as 60 ($6 \times 10$) increased by 14 ($4 + 10$) in French. In Logo, the number 74 is represented as 10 added to 60 ($20 \times 3$) increased by 4. In Yorùbá, in turn, the same number is derived in a more complex way by adding 4 to 80 ($20 \times 4$) reduced by 10. Table 1 shows the representation of the number 74 in the four languages.

The analysis of number names is important but understudied in human language processing. While it may seem trivial to compute number names in languages like English, it may be difficult to get it

Table 1: Derivation of the number 74 in four languages

| **Language** | **Name** | **Derivation** |
|---|---|---|
| English | *seventy four* | $(7 \times 10) + 4$ |
| French | *soixante-quatorze* | $60 + 14$ |
| Logo | *nyaa̱ na drya mudri drya su* | $(20 \times 3) + 10 + 4$ |
| Yorùbá | *ẹ̀rìnlélàádọ́rin* | $4 + (-10 + (20 \times 4))$ |

right in many other languages, particularly in the Yorùbá language. In this paper, we present a formal description of the Yorùbá numeral system; specifically, the problem of Yorùbá number name transcription is addressed from an engineering perspective, by applying standard theories and techniques to an understudied language. This is part of a wider interest in the development of Text-To-Speech (TTS) synthesis and Machine Translation (MT) systems for the Yorùbá language. In TTS and related applications, text normalisation is often the first task, in which Non-Standard Words (NSW) such as numbers, abbreviations, acronyms, time, date, etc. are correctly identified and expanded into their textual forms (Sproat 1996). The expansion of numerical expressions in text is thus a key task in such applications because numbers occur more frequently in varying forms within a block of text. These forms include cardinal numbers, ordinal numbers, telephone numbers, date, time, percentages, monetary value, address, etc.

The rest of this paper is structured as follows: Section 2 gives an analysis of the Yorùbá numeral system and its associated number naming rules. Section 3 discusses the system design and implementation, while Section 4 discusses the results. Section 5 presents the system evaluation and Section 6 concludes the paper with areas of further study.

## 2 THE YORÙBÁ NUMERALS

The Yorùbá language (ISO 639.3 yor), which belongs to the West Benue-Congo branch of the Niger-Congo African languages family, is spoken by about 19,000,000 speakers in the South-Western Nigeria (Owólabí 2006). The language is also spoken in other West African countries such as Central Togo, the East-Central part of the Republic of Benin, and Creole population of Sierra Leone. Outside Africa, Yorùbá (called Nagô, Aku, or Lukumi; Lovejoy and Trotman 2003) is spoken in Brazil, Cuba, and Trinidad and Tobago.

Without a formal method of documenting literature, the Yorùbá community developed a complex numeral system that extensively uses subtraction throughout its system (Verran 2001). This has attracted many linguistic scholars to investigate the reasons why this community has developed an intricate numeral system. Certainly, knowledge of the Yorùbá numeral system has been passed from generation to

generation by means of oral literature. Young language learners, in particular, are made to undergo drills of reciting rhymes with numbers ranging from 1 to 10.

In an early study of the Yorùbá numeral system, Mann (1887) shows how large numbers could be represented as an arithmetic combination of the basic number units and reveals that the subtraction operation plays an important role in number naming. The peculiarity in the Yorùbá numerals was highlighted as follows:

> *"Very different is the framework of the Yorùbá, it can boast of a greater number of radical names of numerals, and to a large extent makes use of subtraction..." (Mann 1887, p. 60)*

A fact worth noting is that some systems illustrate a pervasive use of the subtractive techniques. Examples of such systems are the clock system and the Roman numeral system. In the conventional clock system, when the minute part of time is greater than 30 minutes, the spoken representation can be derived by employing the subtractive technique. For instance, four canonical representations of 2:30 PM are:

(i) Half two (half hour past two)
(ii) Two thirty (2 o'clock + 30 min)
(iii) Thirty minutes after two (2 o'clock + 30 min)
(iv) Thirty minutes to three (3 o'clock − 30 min)

All four representations in (i) to (iv) are acceptable and none has precedence over the other. The form in (iv) is used to a large extent in our daily lives without any difficulty. Similarly, *halb zwei* in German means 'half of the second hour', which is 'half one'. So, the Yorùbá's use of subtraction is not completely exceptional, but its extensive usage may seem unusual, especially when it is preferred over the simpler addition operation.

Another observable feature of the Yorùbá numeral system is the use of base 20 (vigesimal), which likely stems from the counting of cowry shells as described by Mann:

> *"Here we may explain the origin of this somewhat cumbersome system; it springs from the way in which the large sum of money (cowries) are counted. When a bagful is cast on the floor, the*

> *counting person sits or kneels down beside it, takes 5 and 5 cowries and counts silently, 1, 2, up to 20, thus 100 are counted off, this is repeated to get a second 100, these little heaps each of 100 cowries are united, and a next 200 is, when counted, swept together with the first" (Mann 1887, p. 63)*

However, there are vigesimal systems that do not have any relation to cowry shells. A more obvious reason for vigesimal systems could be that humans have 10 fingers and 10 toes. The use of 20 as a base may seem cumbersome, however, it is not entirely exceptional. In many languages, especially in Europe and Africa, 20 is a base with respect to the linguistic structure of the names of certain numbers. Even so, a consistent vigesimal system based on the powers of 20, i.e.: 20, 400, 8000, etc. is not generally used. Examples of a strict vigesimal numeral system are those of Maya and Dzongkha (the national language of Bhutan). The numeral systems of the Ainu language of Japan and Kaire language of Sudan also rely, to an extent, on base 20 for the representation of numbers. Apart from Yorùbá, other African languages with vigesimal numeral system are: Madingo, Mundo, Logone, Nupe, Mende, Bongo, Efik, Vei, Igbo, and Affadeh (Conant 1896). The study by Conant (1896) highlighted the extent of the mental computation required in the expression and conception of the Yorùbá numerals, and concluded that the Yorùbá numeral system is the most peculiar numeral system in existence. One might then begin to wonder why the Yorùbá language, with a simple syllabic structure, will use such a complex numeral system. The reason for this may not be too clear.

Johnson (1921) conducted an analysis of the Yorùbá numerals by focusing on the derivation processes and the morphophonological rules required, and showed how large numbers are calculated in multiples of 20,000. The study by Abraham (1958) examined the arithmetic skills employed in different Yorùbá numeral groups, and provided a guide into their syntactic representation. A profound study on Yorùbá numerals was done by Ẹ́kúndayọ̀ (1977), where the derivational breakdown of the Yorùbá numerals was discussed and the structural representation of Yorùbá numerals was illustrated. In the study, 16 basic number lexemes which serve as the basic building blocks of the Yorùbá numeral system were identified as presented in Section 2.1.

### 2.1 *Basic numbers in Yorùbá*

The Yorùbá counting system has lexemes for basic numbers from 1 to 10 and six higher numerals (i.e.: 20, 30, 200, 300, 400, and 20,000). These 16 basic number lexemes are:

> *ọ̀kan* (1), *èjì* (2), *ẹ̀ta* (3), *ẹ̀rin* (4), *àrún-ún* (5), *ẹ̀fà* (6), *èje* (7), *èjọ* (8), *ẹ̀sán-án* (9), *ẹ̀wá* (10), *ogún* (20), *ọgbọ̀n* (30), *igba* (200), *ọ̀dúnrún* (300), *irínwó* (400), *ọ̀kẹ́* (20,000) (Ẹkúndayọ̀ 1977)

Abraham (1958) and Ẹkúndayọ̀ (1977) also highlighted another set of basic numerals which are multiples of 20 from 20 to 80. These include:

> *okòó* (20), *òjì* (40), *ọ̀tà* (60), and *ọ̀rìn* (80).

These forms of lexemes are used with multiples of 100 between 200 and 20,000. The lexical representation of 20 has two values, i.e., *ogún* or *okòó*, which are used in different contexts. *Okòó* is the only form used in initial word positions when it is added to (*ó lé*) or subtracted from (*ó dín*) a vigesimal, while *ogún* is used with the multiplication formatives in numerical derivation. To illustrate this, 220 may either be expressed as *igba ó lé ogún* (200 increased by 20) or *okòólérúgba* (20 added to 200) but not as *igba ó lé okòó* or *ogúnlérúgba* although they would represent the same quantity.

Numbers are generated using syntactic combination of these lexemes, and only three of the basic mathematical operators are required to represent an infinite set of numbers within the Yorùbá language. These operators are represented by special position words like *lé* for addition, *dín* for subtraction, and *ọ̀nà* for multiplication. However, it should be pointed out that subtraction has an unusually higher functional load than addition. An exponential represented as *ìlọpo* may be required to express very large numerals as powers of 20,000 (Ọdéjọbí 2003) but this is not generally used in the Yorùbá numeral system.

### 2.2 *Overcounting in Yorùbá numerals*

We have mentioned the use of three of the standard arithmetic operations (i.e., multiplication, subtraction, and addition) in the Yorùbá numeral system. However, it is important to discuss a special mode of subtraction depicted by *ẹẹ́dín* and its variant, *aadín*. The *ẹẹ́dín* phenomenon was well articulated in Ẹkúndayọ̀ (1977), where a detailed

explanation of this concept was given. Overcounting (Menninger 1969) occurs when a numeral is expressed in relation to a higher numeral. Overcounting, thus, becomes inevitable within any numeral system employing subtraction operation in number representation.

In the Yorùbá numerals system, when *ẹẹdín* is used with a number, it implies that the number must be reduced by a certain value. The use of *ẹẹdín* or *aadín* is context-dependent; hence, the value deducted varies depending on the numeral to which it is attached. This is shown in Table 2. When *ẹẹdín*' is used with numbers 20 and 30, 5 is deducted from them to produce 15 (*ẹẹdín ogún* = *ẹ̀ẹ̀dógún*) and 25 (*ẹẹdín ọgbọ̀n* = *ẹ̀ẹ̀dọ́gbọ̀n*) respectively. But if *ẹẹdín* is used with 600, 100 is deducted to produce 500 (*ẹẹdín ẹgbẹ̀ta* = *ẹ̀ẹ̀dẹ́gbẹ̀ta*).

| Variant | Number | Reduction |
|---|---|---|
| *ẹẹdín* | 20 and 30 | 5 (half of 10) |
| *aadín* | 60, 80, ..., 200 | 10 (half of 20) |
| *ẹẹdín* | 600, 800, ..., 2000 | 100 (half of 200) |
| *ẹẹdín* | 4000, 6000, ..., 20000 | 1000 (half of 2000) |

Table 2: '*ẹẹdin*' mode of subtraction

The concept of overcounting is also noticeable in the numeral systems of Ainu and Maya. Danish, an essentially Germanic language, also exhibits a related subtractive phenomenon (Conant 1896) as illustrated below:

a) 50 = *halvtredsindstyve* = half (of 20) from 3 × 20
b) 70 = *halvfierdsindstyve* = half (of 20) from 4 × 20
c) 90 = *halvfemsindstyve* = half (of 20) from 5 × 20

Notably, the process of naming numbers in Danish is similar to Yorùbá. Now, we present the rules used in naming numbers in Yorùbá.

### 2.3 *Yorùbá number naming rules*

There are basic rules that hold in the generation of an infinite set of number names in the Yorùbá language as captured in Figure 1. As observed by Hurford (2001), numeral sequences in human languages show several discontinuities in their patterns of representation. Therefore, it is important to identify numeral groups that exhibit similar derivative process within the Yorùbá numeral system. This is to

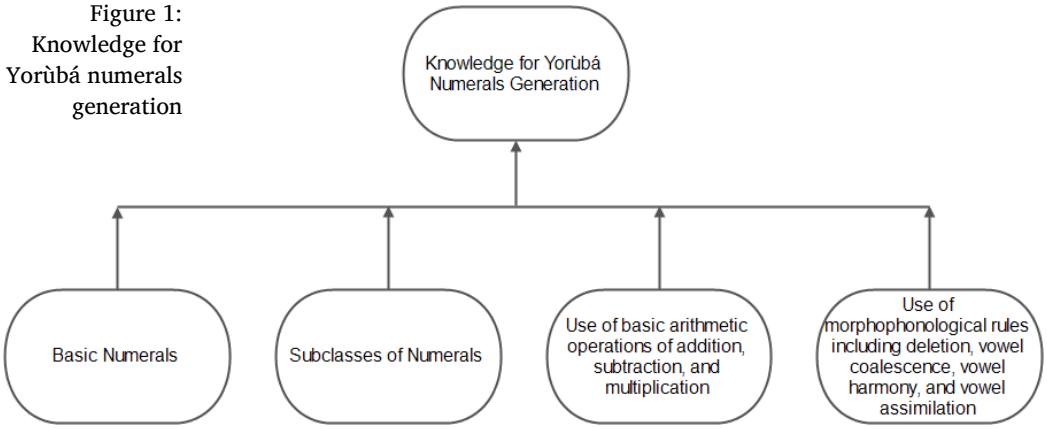



Figure 1: Knowledge for Yorùbá numerals generation

achieve the design of an effective computational model to handle the mathematical and syntactic structure of each group. The groups are:

a) **Basic numbers:** The canonical lexemes in the Yorùbá language have been discussed in Subsection 2.1. This set of lexemes cannot be broken down to simpler forms, and other number names are generated using arithmetic combinations of these lexemes.

b) **Numbers from 11 to 200:** The addition operation is used for deriving numbers from one to four above multiples of 10, while numbers from five to one below such points are obtained through subtraction as illustrated in Figure 2. The Yorùbá lexical representation of number 11 is formed as an additive concatenation of the terms for numbers 1 and 10. This also applies to numbers 12, 13, and 14 as :

   i) 11 = (1+10) = *ọ̀kan lé ẹ̀wá* = *ọ̀kànlá*
   ii) 12 = (2+10) = *èjì lé ẹ̀wá* = *èjìlá*
   iii) 13 = (3+10) = *ẹ̀ta lé ẹ̀wá* = *ẹ̀tàlá*
   iv) 14 = (4+10) = *ẹ̀rìn lé ẹ̀wá* = *ẹ̀rìnlá*

   Note that the lexical representation of '+ 10', i.e., *lé ẹ̀wá* is contracted to *lá*. The Numbers from 15 to 19 are represented as 5 to 1 deducted from 20, respectively.

   i) 15 = 5 from 20 = *àrún-úndínlógún*
   ii) 16 = 4 from 20 = *ẹ̀rindínlógún*
   iii) 17 = 3 from 20 = *ẹ̀tadínlógún*

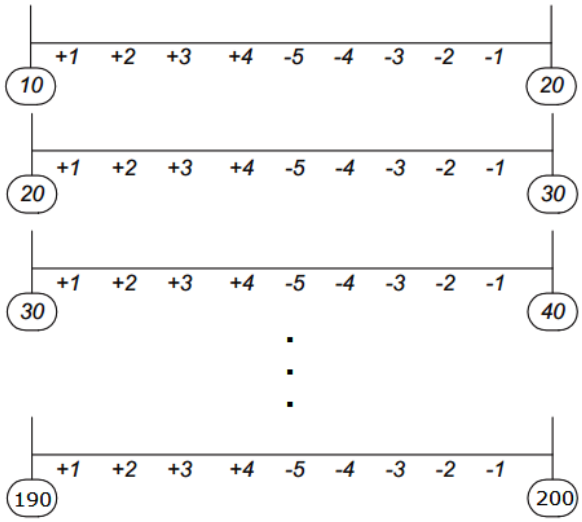



Figure 2: Yorùbá number scale

iv) 18 = 2 from 20 = *èjìdínlógún*

v) 19 = 1 from 20 = *ọ̀kàndínlógún*

Multiples of 20 from 40 to 180 are expressed as such in successive elision and vowel harmony. Numerals 50, 70, 90, 110, 130, 150 and 170 are expressed as 10 deducted (*aadín*) from the next multiple of 20. Again, this is illustrated below:

i) 40 = (20 × 2) = *ogún èjì* = *ogójì*

ii) 50 = 10 less (20 × 3) = *aadín (ogún ẹ̀ta)* = *ààdọ́ta*

iii) 110 = 10 less (20 × 6) = *aadín (ogún ẹ̀fà)* = *ààdọ́fà*

iv) 180 = 20 × 9 = *ogún ẹ̀sàn-án* = *ọgọ́sàn-án*

Notably, a possible representation of 30 is *ààdóji*, which means 10 deducted from 2 twenties ((20 × 2) − 10), but 30 is referred to as *ọgbọ̀n*, which is a generic term in the Yorùbá numeral system.

c) **Numbers From 200 to 2000:** Apart from 400, numbers which are multiples of 200 are derived in multiples of *igba* and the numbers 500, 700, 900, 1100, 1300, 1500, 1700, and 1,900 are derived by 100 deducted (*ẹ̀ẹ̀dín*) from the next multiple of 200.

i) 600 = (200 × 3) = *igba ẹ̀ta* = *ẹgbẹ̀ta*

ii) 1000 = (200 × 5) = *igba àrùn-ún* = *ẹgbẹ̀rún-ún*

iii) $1400 = (200 \times 7) =$ *igba èje* = *egbèje*

iv) $1500 = (1600 - 100) = (200 \times 8) - 100 =$ *ẹẹdín igba èjọ* = *èèdégbèjọ*

v) $2000 = (200 \times 10) =$ *igba ẹ̀wá* = *ẹgbẹ̀wá* = *ẹgbàá*

d) **Numbers Between 2,000 and 20,000:** Numbers in this subgroup are formed from 2,000 as the root word. The multiples of 2,000 within this range are expressed as multiples of *egbàá* and intermediate numbers are formed with the *eedín* that shows a subtraction of 1,000.

i) $6{,}000 = (2{,}000 \times 3) =$ *ẹgbàá èta* = *ẹgbàáta*

ii) $10{,}000 = (2{,}000 \times 5) =$ *ẹgbàá àrún-ún* = *ẹgbàárún-ún*

iii) $15{,}000 = (16{,}000 - 1000) = (2{,}000 \times 8) - 1000 =$ *eedín ẹgbàá èjọ* = *ẹẹdẹgbàájọ*

iv) $20{,}000 = (2{,}000 \times 10) =$ *ẹgbàá ẹ̀wá* = *ẹgbàáwá*. This number is also expressed as *ọ̀kẹ́ kan*.

e) **Numbers 20,000 and above:** Numerals greater than 20,000 are derived as a multiple of 20,000 (*ọ̀kẹ́ kan* = twenty thousand in one place).

i) $40{,}000 = (20{,}000 \times 2) =$ *ọ̀kẹ́ méjì*

ii) $1{,}000{,}000 = (20{,}000 \times 50) =$ *àádọ́ta ọ̀kẹ́*

iii) $800{,}000{,}000 = (20{,}000 \times 20{,}000 \times 2) =$ *ọ̀kẹ́ ona ọ̀kẹ́ méjì*

iv) $8{,}000{,}000{,}000{,}000 = 20{,}000 \times 20{,}000 \times 20{,}000 =$ *ọ̀kẹ́ ọ̀nà ọ̀kẹ́ ọ̀nà ọ̀kẹ́ kan*

Once the number groups are identified, Yorùbá numerals can be represented as a combination of members from each group using the addition and subtraction operations. For example, the number 45,678 will be represented as:

$$45{,}678 = 40{,}000 + 5{,}000 + 600 + 70 + 8 \tag{1}$$

A close observation of these groups shows that certain numbers occur as reference points in the Yorùbá numeral system as proposed by Pollmann and Jansen (1996). An observable trend is that the numbers 20 and 10 play important roles within the Yorùbá numeral system.

## 2.4 The linguistic structure of numerals

In this section, we review two important bibliographic references on the syntactic structure of numerals. The first one is Hurford (1975), which is an extensive study of various numeral systems. The other one is the study conducted by Ẹ́kúndayọ̀ (1977), in which phrase structure rules were proposed for the Yorùbá numeral system.

### 2.4.1 Hurford's generative numeral grammar

A notable work on the application of generative grammar to numerals is the work of Hurford (1975), where the universal phrase structure rules for deriving numerals were presented. A modified version of the phrase structure rules was presented in Hurford (2007), being a significant improvement with respect to well-formed numerals. In this extensive study of numerals, Hurford considered numerals as syntactic structural constructs and proposed a universal constraint on numerals, which he called the packing strategy. The packing strategy helps to make the right choice for a number name from different structural constructs derived by the production rules presented in Definition 1. The packing strategy guides the general constraints on the well-formed nature of numerals and any structure containing an ill-formed structure is itself ill-formed.

**Definition 1 (Hurford's production rules for Yorùbá numerals)** *Hurford's production rules for the Yorùbá numeral system are as follows:*

$$NUM \rightarrow \left\{ \begin{array}{c} DIGIT \\ NP \end{array} \right\} (NUM) \tag{2}$$

$$NP \rightarrow (M)\ NUM \tag{3}$$

$$M \rightarrow 10\left( \left\{ \begin{array}{c} 2 \\ M \end{array} \right\} \right) \tag{4}$$

*Where DIGIT is a set of basic number lexemes, M is a set of multiplicative base lexemes, NUM is a numeral and the start symbol, and NP is a Number Phrase. Rule (2) is interpreted as addition/subtraction, and it can occur in reverse order, i.e., NUM → NUM NP. Rules (3) and (4) are interpreted as multiplication when two constituents are chosen. The curly brace in the production rules shows alternative productions, while parenthesis indicates*

*an optional item. For example, an NP can be formed from a single NUM or a multiplicative combination of M NUM.*

Hurford's generative framework provides an adequate account for the numeral system of most languages including English. However, the grammar proposed for the Yorùbá numeral system was structurally inadequate. It is worth noting that the grammar does not provide an adequate mechanism to differentiate between the addition and subtraction operations in Rule (2). For example, Hurford (1975) presented structures for 46 and 4,600, as shown in Figure 3. In the structure in Figure 3(a), 46 (*ẹ̀rindínlàádọ́ta*) was derived by deducting 4 (*ẹ̀rin*) from 50 (*àádọ́ta*) and 50 was derived by deducting 10 from 60 (*ọ̀gọ́ta*). In Figure 3(b) and (c), representing structures for 4,600, i.e., *ẹgbẹ̀talélogún* (200 × 23) and *ẹgbàájì ó lé ẹgbẹ̀ta* (4,000 + 600) respectively, 23 (*ẹ̀talélogún*) was derived by adding 3 (*ẹ̀ta*) to 20 (*ogún*) and 4,600 was derived by 4,000 plus 600. Rule *NUM → NUM NP* is interpreted as subtraction in Figure 3(a), whereas, it is interpreted as addition in Figures 3(b) and (c). This means that the structure in Figure 3(a) could be misinterpreted as 54, and structures in Figure 3(b) and (c) as 3,400. Therefore, this introduces ambiguity in interpretation. It is also important to point out that Rule (4) results in an incorrect interpretation of the structures of *M*. To illustrate this, the rule represents 20 (*ogún*) as a combination of 10 (*ẹ̀wá*) and 2 (*èjì*), which is structurally incorrect. This is because *ogún* is not formed, by any means, from the combination of *ẹ̀wá* and *èjì*.

The study also acknowledged that multiple structures may exist for some numbers like 4,600, as shown in Figures 3(b) and 3(c), but concluded that the structure in 3(c) was ill-formed, whereas it is a valid structure in Yorùbá. This conclusion could result from a limited expert knowledge in verifying the correctness of these structures, as noted:

> *"Despite the difficulty in finding crucial information in the sources, it is conceivable that some complete account of Yorùbá numerals can be given that is soundly motivated. This language certainly presents the weightiest challenge for a general theory of numerals that we have encountered." (Hurford 1975, p. 232)*

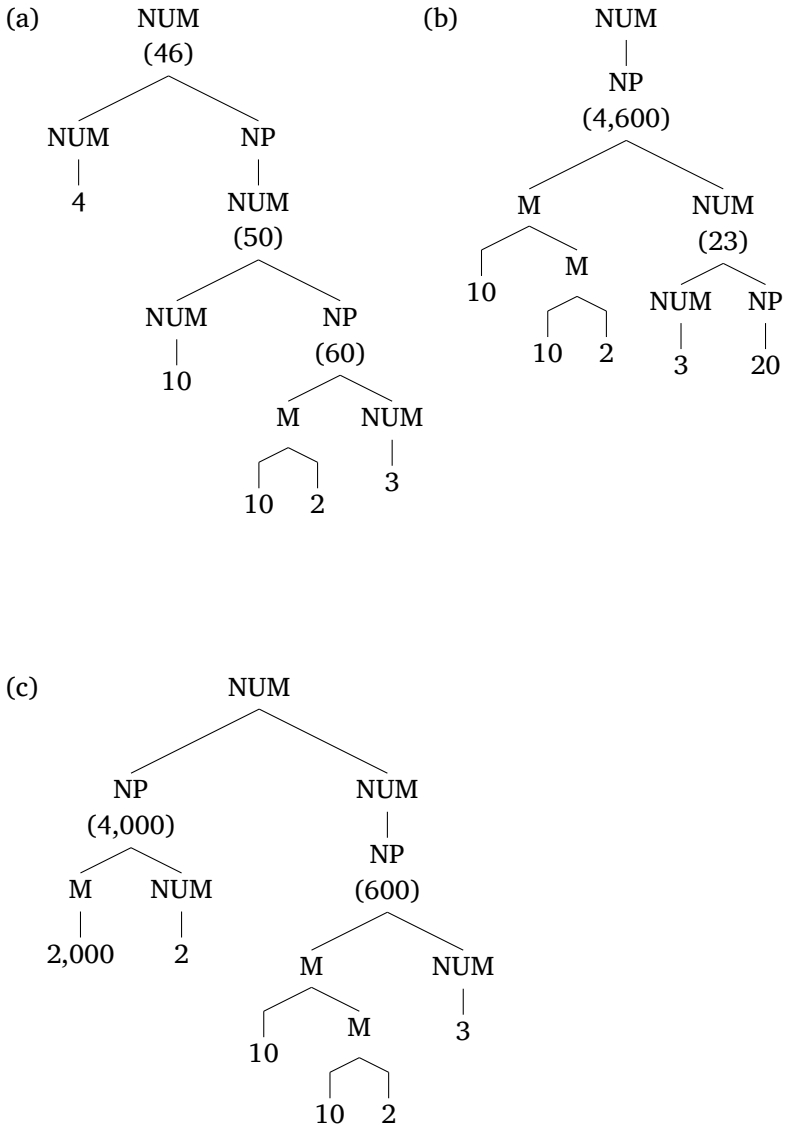



Figure 3:
Parse tree for
(a) 46 derived by 4 deducted from (10 deducted from $(20 \times 3)$)
(b) 4,600 derived by $200 \times (3 + 20)$
(c) 4,600 derived by $(2{,}000 \times 2) + (200 \times 3)$

### 2.4.2 Ẹkúndayọ̀'s phrase structure rules

The study conducted by Ẹkúndayọ̀ (1977) reveals that there exist similarities between the mechanism used in the Yorùbá language for constructing an infinite number of sentences from a finite set of building blocks and constructing an infinite set of numerals from a limited set of basic numbers. This proposition was corroborated into 3 different concepts as shown in Table 3. This shows that all Yorùbá numerals can be sententially represented through the addition, subtraction, and multiplication operators. The study also shows that some numbers have multiple representations in the Yorùbá language, but constraints of correctness are imposed on these representations. These constraints include linguistic and structural plausibilities.

Apart from the concept of infinity, creativity, and paraphrasable representation of numerals, Ẹkúndayọ̀ (1977) demonstrated that a recursive grammar is needed for numeral derivation and representation. It was observed that the recursive rules are not easily established for the Yorùbá numerals, however, a set of phrase structure (PS) rules for the Yorùbá numeral system was given as shown in Definition 2.

Table 3: Comparison of sentence construction and number naming in Yorùbá

| No. | Concept | Language | Yorùbá numerals |
|---|---|---|---|
| 1 | **Infinity** | There is no longest sentence. Any sentence, however long, can be expanded. So with the use of recursive rules, an infinite number of sentences can be constructed. | Numerals are infinitely enumerable. This means that there is no longest numeral. Any numeral, however large can still be increased. So, the concept of recursive rules can be adopted in numerals. |
| 2 | **Creativity** | It is possible to construct and perceive an entirely new sentence that has never been heard before. | Yorùbá numerals also require a high level of creativity as higher numerals must be recreated every time they are used. |
| 3 | **Paraphrase** | A single idea could be represented in several ways. | A single number may also be represented in different forms in Yorùbá numerals. |

**Definition 2 (Ẹkúndayọ̀'s PS rules for Yorùbá numerals)**
*Ẹkúndayọ̀ phrase structure rules for the Yorùbá numeral system are as follows:*

$$NUM \rightarrow NP \tag{5}$$

$$NP \rightarrow \left\{ \begin{array}{l} NP\ S \\ N \\ PRON \end{array} \right\} \tag{6}$$

$$S \rightarrow NP\ VP \tag{7}$$

$$VP \rightarrow V\ NP \tag{8}$$

*Where $NUM$ is a numeral and the start symbol, $NP$ is a noun phrase, $VP$ is a verb phrase, $S$ is a sentence, $N$ is the set of 16 basic number lexemes, $PRON$ is the formative ó ('it'), and $V$ is a verb represented as the operating formatives ọ̀nà (for multiplication), dín (for subtraction), and lé (for addition). NOTE: Rule (8) was presented as $V \rightarrow V\ \ NP$ in the original article but it was modified to make the grammar complete.*

The point of interest here is that verbs are used in number naming, and that numbers are sententially represented in their surface structure. This allows for a distinction between addition and subtraction operations. This is illustrated by the structure of *ẹ̀rinlélàádọ́ta* (54), shown in Figure 4, where the operating formative ($V$) is explicitly represented. Although these PS rules proved useful in the derivation of Yorùbá numerals, they are mostly arithmetic rather than syntactic rules as the positions of the basic lexical numerals and operatives do not correspond to their positions in the surface structure. An example would be the surface structure of *ẹ̀rinlélàádọ́ta* (54) represented in Figure 4 as ((*ọ̀gún ọ̀nà mẹ́ta) ó dín ẹ̀wá) ó lé ẹ̀rin* rather than *ẹ̀rin ó lé (aadín (ọ̀gún ọ̀nà mẹ́ta*)), thereby leading to a misrepresentation of numerals.

Another problem with Ẹkúndayọ̀'s PS rules is that multiplicative bases ($M$) in Hurford's grammar are not captured. The multiplicative bases help to understand which numbers are important milestones in a numeral system. Hence, in this paper, we used knowledge from these two models to capture the essential components of the Yorùbá numerals. The grammar developed captures the multiplicative bases and treats Yorùbá numerals as both arithmetic and syntactic constructs.

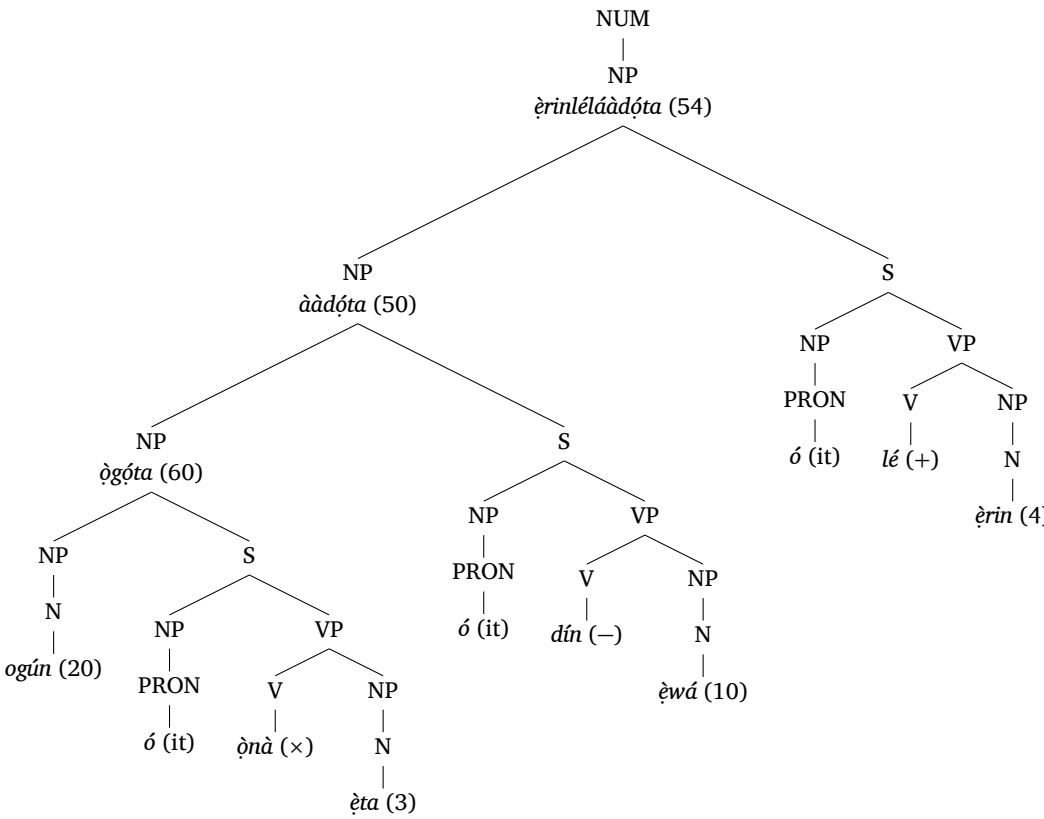



Figure 4: Parse tree for *ẹ̀rinlélàádọ́ta* (54)

## 3 SYSTEM DESIGN AND IMPLEMENTATION

It has been shown that the Yorùbá numeral system is very methodical, thus, an efficient computational system is required to gain accuracy in number representation. Figure 5 presents the block diagram of number transcription in the Yorùbá language. There are four important processes in this model. First, there is the number decomposition process, where numbers are expressed as a sum of smaller numbers in harmony with the sub-grouping discussed in Section 2.3. The output of this process is the magnitude stack. Next, there is a process that generates the possible forms of a single number. This is done by careful combinations of neighbouring elements of the magnitude stack and parsing them with the designed numeral grammar. This is done by using

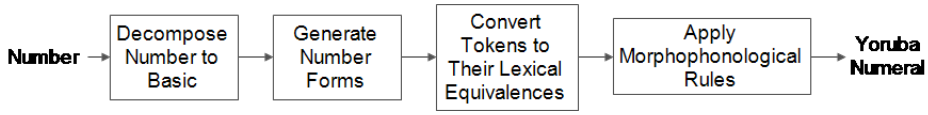



Figure 5: Number to Yorùbá text transcription system. The figure shows the processes involved in converting a cardinal number to Yorùbá text.

the packing strategy to verify whether the structures are well-formed. The third process is where tokens of the number forms are converted to their equivalent lexical forms, and the final process is where the morphophonological rules employed in Yorùbá naming numbers are applied.

### 3.1 *Number decomposition to vigesimal*

Within the Yorùbá numeral system, every number can be represented using a combination of five different smaller terms, each drawn from the possible groups of the Yorùbá numeral system. So, the first process is to generate the magnitude stack from the given number. This generates five new numbers $(d_0, d_1, d_2, d_3, d_4)$ from the given number. So that

$$number = d_4 + d_3 + d_2 + d_1 + d_0 \tag{9}$$

where

a) $d_0$ is 0 or a member of subgroup (a), i.e.,
$d_0$ takes values from 0 to 9, i.e., $d_0 \in DIGIT = \{0, 1, 2, ..., 9\}$.

b) $d_1$ is 0 or a member of subgroup (b), i.e.,
$d_1$ is a multiple of 20 ($d_1 = 20 \times n \mid 0 \leq n < 10$) **or** 10 deducted from a multiple of 20 ($d_1 = (20 \times n) - 10 \mid 2 \leq n \leq 10$).

c) $d_2$ is 0 or a member of subgroup (c), i.e.,
$d_2$ is a multiple of 200 ($d_2 = 200 \times n \mid 0 \leq n < 10$) **or** 100 deducted from a multiple of 200 ($d_2 = (200 \times n) - 100 \mid 2 \leq n \leq 10$).

d) $d_3$ is 0 or a member of subgroup (d), i.e.,
$d_3$ is a multiple of 2,000 ($d_3 = 2{,}000 \times n \mid 0 \leq n < 10$), **or** 1,000 deducted from a multiple of 2,000 ($d_3 = (2{,}000 \times n) - 1{,}000 \mid 2 \leq n \leq 10$).

e) $d_4$ is 0 or a member of subgroup (e), i.e.,
$d_4$ is a multiple of 20,000 ($d_4 = 20000 \times n \mid 0 \leq n < \infty$).

Table 4: Magnitude stack of some numbers

| Number | Magnitude stack |
|---|---|
| 23 | [20, 3] |
| 167 | [160, 7] |
| 3,459 | [3,000, 400, 50, 9] |
| 19,669 | [19,000, 600, 60, 9] |
| 412,987 | [400,000, 12,000, 900, 80, 7] |
| 1,876,234 | [1,860,000, 16,000, 200, 30, 4] |

These new numbers can be derived using Algorithm 1. Any of $d_4, d_3, d_2, d_1, d_0$ that is equal to zero is removed from the magnitude stack. The magnitude stacks of some numbers are presented in Table 4. For example, the magnitude stack generated for number 1,876,234 was:

$$[d_4, d_3, d_2, d_1, d_0] = [1{,}860{,}000,\ 16{,}000,\ 200,\ 30,\ 4]$$

In the next section, we discuss how the representations of single numbers are generated.

### 3.2 *Generating forms of a number*

Once the magnitude stack has been computed, the next task is to generate the possible forms of the number in Yorùbá. All the possible Yorùbá forms of a number are derived by some combinations of neighbouring elements of the magnitude stack. The possible forms for a number with magnitude stack of $[d_4, d_3, d_2, d_1, d_0]$ are listed in Table 5. For example, the magnitude stack for 19,669 is $[d_3, d_2, d_1, d_0] =$ [19,000, 600, 60, 9], and the possible forms are shown in Table 6. All possible forms are then stored in the form stack. However, not all numbers exhibit all these forms. The number of forms largely depends on the values of $d_4, d_3, d_2, d_1$, and $d_0$.

Thereafter, the elements of the form stack are expanded to a form containing only the symbols representing the basic lexical items. The expanded form stack for number 19,669 is presented in Table 7. In these forms, '×' represents multiplication, '—' and '+' represent subtraction and addition within a number phrase respectively; '——' and '++' represent subtraction and addition between number phrases respectively, as discussed in Section 3.3 b(ii). It should be noted that

Algorithm 1: Magnitude generator algorithm

```
Data: number: Input number
Result: magnitudeStack: The magnitude stack
procedure GenerateMagnitude(number)
    d_0, d_1, d_2, d_3, d_4 = 0;
    divisor = 10;
    magnitudeStack = [ ];
    while number ≠ 0 do
        remainder = number % divisor;
        if remainder ≠ 0 then
            magnitudeStack.push(remainder);
        end if
        number = number − remainder;
        divisor = divisor × 10;
    end
    for mag in magnitudeStack do
        if mag < 10 then
            d_0 = d_0 + mag;
        else if mag < 200 then
            d_1 = d_1 + mag;
        else if mag < 2000 then
            d_2 = d_2 + mag;
        else if mag < 20000 then
            d_3 = d_3 + mag;
        else
            d_4 = d_4 + mag − (mag % 20000);
            d_3 = d_3 + (mag % 20000);
        end if
    end
    magnitudeStack = [d_0, d_1, d_2, d_3, d_4];
    return magnitudeStack.reverse();
end procedure
```

arithmetic is mostly done from right to left in the Yorùbá numeral system, i.e., $2-20$ implies 2 removed from 20, which gives 18. In the same way, $(10-(20\times4))$ implies 10 deducted from $(20\times4)$ to give 70.

Table 5: Forms of Yorùbá number

| | Derivation |
|---|---|
| 1 | $[d_4, d_3, d_2 + d_1, d_0]$ |
| 2 | $[d_4, d_3, d_2 + d_1 + 20, d_0 - 20]$ |
| 3 | $[d_4, d_3, d_2 + d_1 - 20, d_0 + 20]$ |
| 4 | $[d_4, d_3, d_2, d_1 + d_0]$ |
| 5 | $[d_4, d_3, d_2 + 100, d_1 + d_0 - 100]$ |
| 6 | $[d_4, d_3 + 1000, d_2 - 1000, d_1 + d_0]$ |
| 7 | $[d_4, d_3 + 1000, d_2 - 1000 + 100, d_1 + d_0 - 100]$ |
| 8 | $[d_4, d_3 + 1000, d_2 + d_1 - 1000, d_0]$ |
| 9 | $[d_4, d_3 + d_2, d_1 + d_0]$ |
| 10 | $[d_4, d_3 + d_2 + 100, d_1 + d_0 - 100]$ |

Table 6: Generation of the forms of 19,669. Item $d_4$ is discarded because $d_4 = 0$

| | Derivation | Form Stack |
|---|---|---|
| 1 | $[d_3, d_2 + d_1, d_0]$ | [19,000, 660, 9] |
| 2 | $[d_3, d_2 + d_1 + 20, d_0 - 20]$ | [19,000, 680, −11] |
| 3 | $[d_3, d_2 + d1 - 20, d_0 + 20]$ | [19,000, 640, 29] |
| 4 | $[d_3, d_2, d_1 + d_0]$ | [19,000, 600, 69] |
| 5 | $[d_3, d_2 + 100, d_1 + d_0 - 100]$ | [19,000, 700, −31] |
| 6 | $[d_3 + 1000, d_2 - 1000, d_1 + d_0]$ | [20,000, −400, 69] |
| 7 | $[d_3 + 1000, d_2 - 1000 + 100, d_1 + d_0 - 100]$ | [20,000, −300, −31] |
| 8 | $[d_3 + 1000, d_2 + d1 - 1000, d_0]$ | [20,000, −340, 9] |
| 9 | $[d_3 + d_2, d_1 + d_0]$ | [19,600, 69] |
| 10 | $[d_3 + d_2 + 100, d_1 + d_0 - 100]$ | [19,700, −31] |

### 3.3 *Context-free grammar for Yorùbá numerals*

We studied the structures of the five numeral groups discussed in Section 2.3, from which some patterns became apparent. We started the design of the CFG by identifying the set of terminal symbols which are:

a) The set of lexemes listed in Section 2.1.

   i) *DIGIT* = {*ọ̀kan* (1), *èjì* (2), *ẹ̀ta* (3), *ẹ̀rin* (4), *àrún-ún* (5), *ẹ̀fà* (6), *èje* (7), *ẹ̀jọ* (8), *ẹ̀sán-án* (9), *ẹ̀wá* (10), *ọgbọ̀n* (30), *ọ̀dún-rún* (300), *irínwó* (400), *okòó* (D20), *òjì* (D40), *ọ̀tà* (D60), *ọ̀rìn* (D80)}, and

   ii) The set of multiplicative bases i.e., $M$ = {*ogún* (20), *igba* (200), *ọ̀kẹ́* (20,000)}

b) The sets of lexical affixes depicting arithmetic operations in Yorùbá numerals. These three sets of operators are:

| No | Form Stack |
|---|---|
| 1 | $[(1{,}000-(2{,}000\times 10))++(D60+(200\times 3))++9]$ |
| 2 | $[(1{,}000-(2{,}000\times 10))++(D80+(200\times 3))--(1+10)]$ |
| 3 | $[(1{,}000-(2{,}000\times 10))++(D40+(200\times 3))++(1-30)]$ |
| 4 | $[(1{,}000-(2{,}000\times 10))++(200\times 3)++(1-(10-(20\times 4)))]$ |
| 5 | $[(1{,}000-(2{,}000\times 10))++(100-(200\times 4))--(1+30)]$ |
| 6 | $[20{,}000--400++(1-(10-(20\times 4)))]$ |
| 7 | $[20{,}000--300--(1+30)]$ |
| 8 | $[20{,}000--(D40+300)++9]$ |
| 9 | $[(200\times(2-(20\times 5)))++(1-(10-(20\times 4)))]$ |
| 10 | $[(100-(200\times(1-(20\times 5))))--(1+30)]$ |

Table 7: Expanded forms of number 19,669. '++' and '−−' are operation formatives found between phrases, while *D*40, *D*60, and *D*80 are multiples of 20 as mentioned in Section 2.1

i) A set of operators that occur within a phrase. This includes *lé ní* (+) for addition and *dín ní* (−) for subtraction. Multiplication within a phrase is implied, which means it is not explicitly represented. Hence, $V = \{lé\ ní,\ dín\ ní\}$.

ii) A set of operators that occur between phrases. This includes *ó lé* (++) for addition, *ó dín* (−−) for subtraction and *ọ̀nà* (×) for multiplication. So we say $VV = \{ó\ lé,\ ó\ dín,\ ọ̀nà\}$.

iii) A set of implied subtraction operators represented by the prefixes *aadín* (reduction by 10) and *ẹẹdín* (reduction by 5, 100, and 1,000), i.e., $REDUCE = \{aadín,\ ẹẹdín\}$.

Thus the set of terminal symbols, $T$, is made up of all elements in: *DIGIT*, $M$, $V$, $VV$, and *REDUCE*.

The start symbol is a numeral which is denoted by *NUM*. Since a CFG is the union of simpler grammars (Sipser 2007), we started by constructing rules for structures of numerals that could occur as a number phrase.

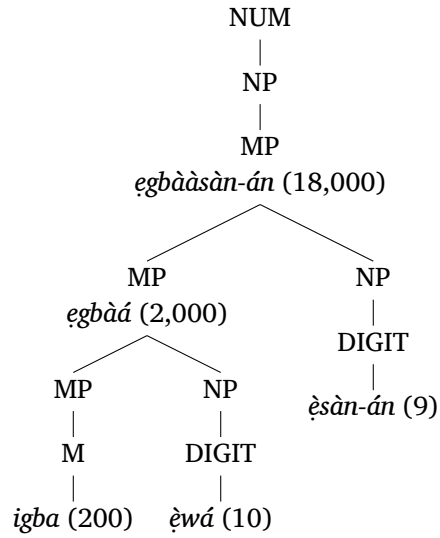
A phrase could be formed as a single *DIGIT* (Ẹkúndayọ̀ 1977) or from the multiplication of $M$ and *DIGIT*. A phrase formed by multiplication is denoted by *MP* (Hurford 2007), i.e.,

$$NP \rightarrow DIGIT \mid MP \tag{10}$$

$$MP \rightarrow M \mid MP\ NP \tag{11}$$

*MP* is formed by a single multiplicative base *M*, or recursively by multiplying *MP* by a number phrase *NP*, e.g. *ọgọ́ta* is formed by multiplying an *MP* (20 – formed by $MP \rightarrow M$) and an *NP* (3 – formed

Figure 6:
Parse tree of
*ẹgbàásàn-án* (18,000)




by *NP → DIGIT*). Also, Rule (11) is recursive to handle multiple levels of multiplication. For example, 18,000 (*ẹgbàásán*) is represented as 2,000 multiplied by 9, and 2,000 is subsequently represented as 200 multiplied by 10, as shown in the parse tree in Figure 6.

We then added a rule to make allowance for the *ẹẹdín/aadín* type of subtraction. The phrase *ẹẹdín/aadín* can only occur as a prefix to a number derived from a multiplication operation. When this is done, the value deducted depends on the number to which it is prefixed (discussed in Subsection 2.2). A further example is 50 (*ààdọ́ta*), which is derived by deducting 10 from 60 (*ọgọ́ta*). Rule (12) captures this as shown in the structure in Figure 7.

$$NP \rightarrow REDUCE\ MP \tag{12}$$

With the inclusion of this rule, it should be pointed out that it has some obvious consequences. The rule overgenerates, that is, it allows the use of *ẹẹdín* or *aadín* without respecting Table 2. We shall devise means of filtering out ill-formed structures using the packing strategy.

The next stage refers to how the operators *V* (Verbs) are represented within a phrase. Within a phrase, the Yorùbás start number presentation with the smaller number (Addend/Subtrahend) rather than the larger number (Augend/Minuend). For instance, number 21

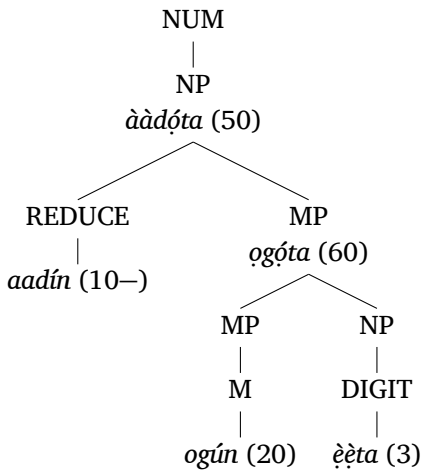



Figure 7:
Parse tree of *ààdọ́ta* (50)

(twenty one) is represented as *ọ̀kànelélógún* (1+20) in Yorùbá. We then considered '1 +' as a verb phrase (*VP*), which is made up of a *DIGIT* and a *V* as presented in Rule (13):

$$VP \rightarrow DIGIT \;\; V \tag{13}$$

A *VP* can then be combined with an *NP* to make up an *NP*, i.e.:

$$NP \rightarrow VP \;\; NP \tag{14}$$

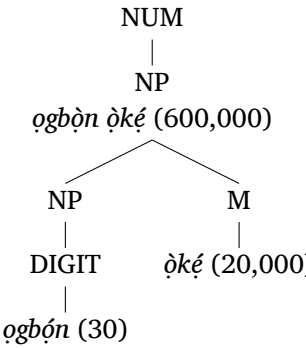
Also, the order in Rule (11) can be reversed to capture the structure of numbers like 600,000 (*ọgbọ̀n ọ̀kẹ́*), which is represented as 30 times 20,000. The multiplicative base is now positioned at the end of the rule, and can only take *ọ̀kẹ́* (20,000) as a value. The outcome of this new rule is a phrase (*NP*), since it cannot be used as a multiplicand (*MP*) to derive higher numerals. For example, 1,200,000 cannot be represented as *ọgbọ̀n ọ̀kẹ́ ọ̀nà mẹ́wàá* ((30 × 20,000) × 10), but as *ọ̀dúnrún ọ̀kẹ́* (300 × 20,000). So we added Rule (15). The structure of number 600,000 is shown in Figure 8.

$$NP \rightarrow NP \;\; M \tag{15}$$

Next, we created rules to connect these phrases together to form a number. So, a number could be formed from a phrase, i.e.:

$$NUM \rightarrow NP \tag{16}$$

Figure 8:
Parse tree of
*ọgbọ̀n ọ̀kẹ́* (600,000)




Also, a number could be formed by combining an existing number with a phrase using the lexical operatives in the set *VV*. We added two rules to capture this as follows:

$$NUM \rightarrow NUM\ S \quad (17)$$

$$S \rightarrow VV\ NP \quad (18)$$

Although multiplication plays an important role in Yorùbá numerals, its lexical representation, *ọ̀nà* does not occur in number names except when more than one 20,000 (*ọ̀kẹ́*) occur within a number phrase. For example, 400,000,000 is represented as 20,000 × 20,000, i.e., *ọ̀kẹ́ ọ̀nà ọ̀kẹ́ kan*, and the structure is also captured using Rule (18) as shown in Figure 9.

Finally, all these rules were merged to make up the production rules of the Yorùbá numeral grammar, as presented in Definition 3.

**Definition 3 (Production rules of the Yorùbá numeral grammar)** *The production rules for the Yorùbá numeral system are as follows:*

$$NUM \rightarrow NP \mid NUM\ S \quad (19)$$

$$S \rightarrow VV\ NP \quad (20)$$

$$NP \rightarrow DIGIT \mid MP \mid VP\ NP \quad (21)$$

$$NP \rightarrow REDUCE\ MP \mid NP\ M \quad (22)$$

$$MP \rightarrow M \mid MP\ NP \quad (23)$$

$$VP \rightarrow DIGIT\ V \quad (24)$$

These phrase structure rules include the verbs which are operating formatives (V and VV) proposed by (Ẹ́kúndayọ̀ 1977). These rules pro-

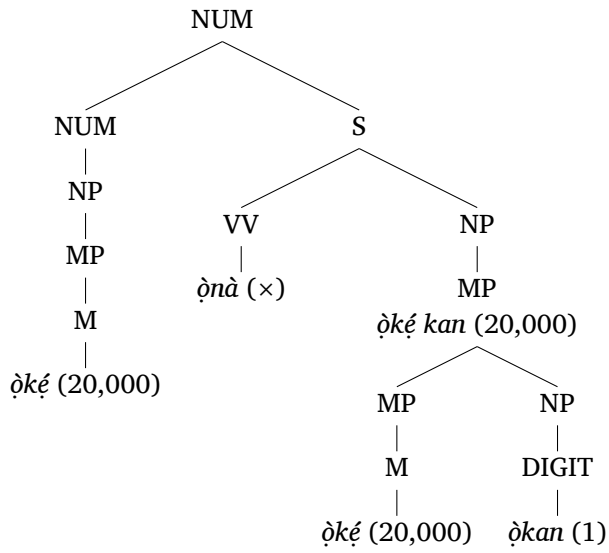



Figure 9:
Parse tree of
*ọ̀kẹ́ ọ̀nà ọ̀kẹ́ kan*
(400,000,000)

duce a single and correct structure for most Yorùbá numerals, however, the rules overgenerate with some numerals. For example, the number 1,000,000 produces 3 structures as presented in Figure 10, but the valid structure is determined using a single packing strategy defined in Definition 4.

**Definition 4 (Packing strategy for the Yorùbá numeral system)**
*The following metarules govern well-formed Yorùbá numeral structures:*

- *(i) Whenever a phrase MP is formed by a multiplicative combination of two numerals, the multiplicand (MP) must be greater than the multiplier (NP).*
- *(ii) Whenever the rule NP → REDUCE MP is used, the lexical item of REDUCE must correspond to the appropriate MP, as shown in Table 2.*
- *(iii) Whenever the rule S → VV NP is used and the VV has the value of ọ̀nà, then NP can only take a value of ọ̀kẹ́, and the resulting S must be used with a multiple of ọ̀kẹ́. (see Figure 9).*

Using the packing strategy for Yorùbá numerals, the well-formedness of the structures in Figure 10 was investigated and only the structure in (c) was well-formed. The analysis is as follows:

(i) In the structure in Figure 10(b), the formation of *MP* from *ogún* (20) disagrees with metarule (i), thereby making the structure in Figure 10(b) ill-formed.

(ii) The structure in Figure 10(a) is not well-formed as *REDUCE* (*aadín* (10–)) is applied to a multiple of *ọ̀kẹ́* (20,000), which

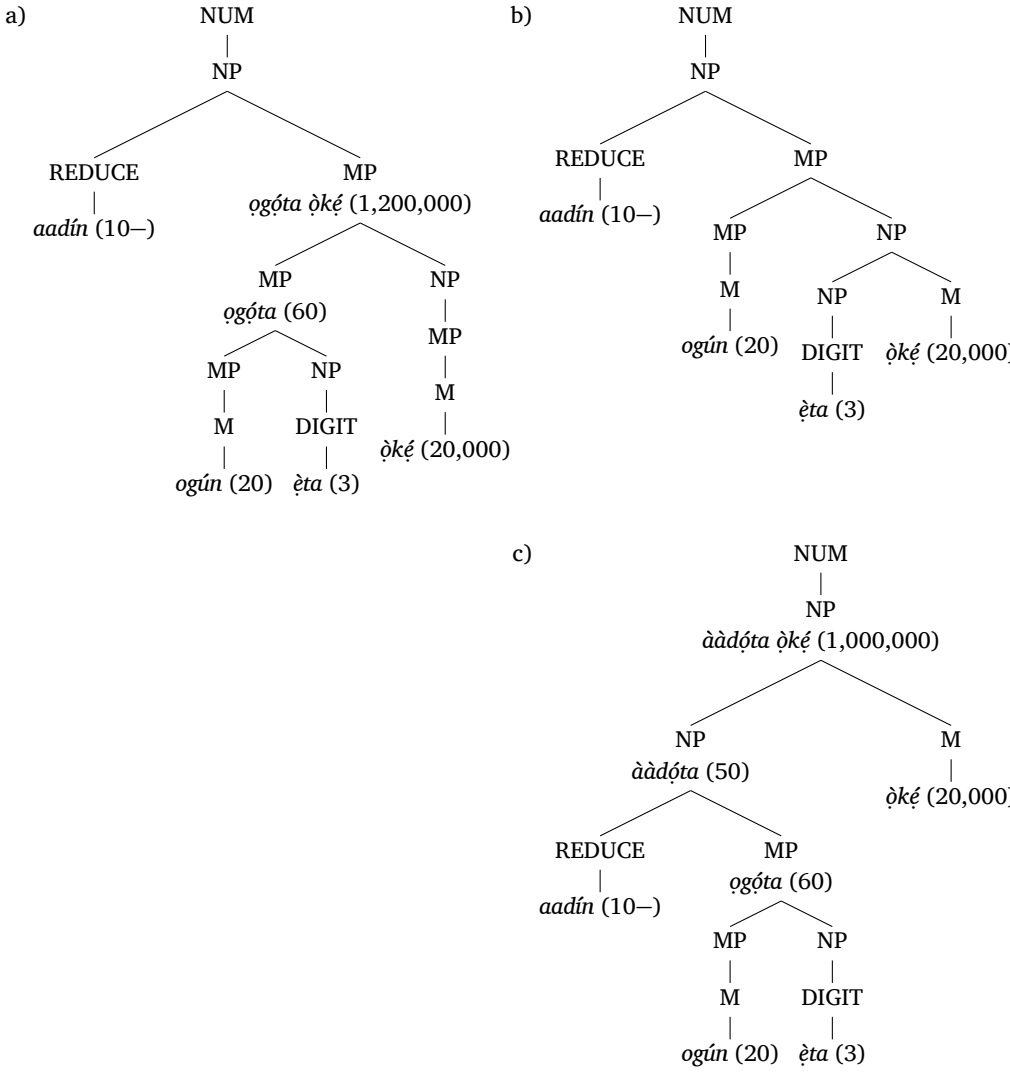



Figure 10: Parse trees of *ààdọ́ta ọ̀kẹ́* (1,000,000)

disagrees with metarule (ii). Hence, only the structure in Figure 10(c) is well-formed.

Once a parse tree is generated, we then convert the tokens to their lexical equivalences followed by the application of morphophonological rules.

## 3.4 *Morphophonological rules in Yorùbá numeral system*

The representation of numbers in Yorùbá is cumbersome due to the fact that a high level of linguistic processing is involved. Therefore, the speakers are required to have adequate knowledge of some morphophonological rules in the Yorùbá language. These morphophonological rules include deletion, vowel coalescence, vowel harmony, and vowel assimilation. These rules will be discussed to show how they are useful in number naming.

### 3.4.1 Deletion

Deletion is a process by which a phrase or word is shortened by completely deleting a segment. Both vowels and consonants can be deleted in Yorùbá. The most commonly deleted consonants are *w* (when it is part of the last syllable) and *g*. Deletion is notable in the contracted form of phrases *dín ní* (less than) and *lé ní* (more than), where *i* is completely deleted and *n* is converted to *l*. This conversion is possible because *n* and *l* are allophones of the same phoneme. For example, the expression for 28, which is derived as 2 from 30 (*èjì dín ní ọgbọ̀n*), is *èjìdínlógbọ̀n*.

A deletion also occurs in naming numbers between 11 and 14. For example, *ókànlá* (11) is formed by adding 1 to 10, i.e., *òkan lé ẹ̀wá*, which is contracted to form *òkanlẹ́wá* by deleting the vowel *é*. The consonant *w* and vowel *ẹ* are then deleted to form *òkànlá*. Another example is *ẹ̀ẹ̀dégbèta*, which is formed from *ẹẹdín ẹgbèta*. This is achieved by completely deleting the vowel *ín*.

### 3.4.2 Vowel coalescence

Coalescence is a phonological process whereby two adjoining segments converge or fuse into one element such that the new segment is

phonologically distinct from the input segments (Bámiṣilẹ̀ 1994). This is illustrated by Equation 25, where $V_1$ is the vowel that ends the first morpheme, $V_2$ is the vowel that begins the second morpheme, '+' is the morpheme boundary, and $V_3$ is the resulting morpheme.

$$V_1 + V_2 \rightarrow V_3 \tag{25}$$

In coalescence, the combining vowels may be phonologically distinct from each other but the resulting vowel must be distinct from the combining vowels, i.e., $V_1 \neq V_3$ and $V_2 \neq V_3$ (Awóbùlúyì 1987).

Vowel coalescence is most notable when two nouns are next to each other. And since Yorùbá numerals are mostly treated as nominal entities, they also use vowel coalescence in naming numbers. For example, vowel coalescence is used in the formation of *ogójì* (40) derived from 20 multiplied by 2, i.e., *ogún èjì*. The vowels *ún* and *e* are combined by coalescence to become *o*. Table 8 shows the possible occurrence of vowel coalescence in the Yorùbá numeral system.

Table 8: Vowel coalescence in Yorùbá numerals. $V_1$ is the vowel that ends the first morpheme, $V_2$ is the vowel that begins the second morpheme, + is the morpheme boundary, and → stands for 'rewritten as'

| $V_1$ | $V_2$ | $V_3$ | **Example** |
|---|---|---|---|
| *ún* | *à* | *ọ́* | *ogún + àrùn-ún → ọgọ́rùn-ún* |
| *ún* | *è* | *ọ́* | *ogún + èjì → ọgọ́jì* |
| *ún* | *ẹ̀* | *ọ* | *ogún + ẹ̀ta → ọgọ́ta* |
| *í* | *i* | *ú* | *èjì dín ní + igba → èjìdínlúgba* |

### 3.4.3 Vowel harmony

Standard Yorùbá has 7 oral vowels, which are: *a, e, ẹ, i, o, ọ, u*. Vowel harmony places a constraint on the occurrence of vowel sequence in words. Archangeli and Pulleyblank (1989) discussed the two classes of Standard Yorùbá oral vowels which are:

i) Advanced Tongue Root (ATR), which are vowels *i, e, o,* and *u*,
ii) Non-ATR, which are vowels *a, ẹ*, and *ọ*

ATR vowels involve drawing forward the root of the tongue so that the pharynx is expanded. In simple Yorùbá words, the last vowel in the word determines the other vowels in the word (Akinlabí 2004). So, if the last vowel in a word is an ATR, the immediately preceding vowel must be an ATR. The high vowels (*i* and *u*) do not participate

in the vowel harmony at all, and they can occur with any vowel. Only the mid vowels (*e, o, ẹ*, and *ọ*) are fully involved in the vowel harmony (Akinlabí 2004). The chart presented in Table 9 shows the permissible and non-permissible sequences of vowels in the Yorùbá language.

| | | $V_2$ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | i | e | ẹ | a | ọ | o | u |
| $V_1$ | i | + | + | + | + | + | + | + |
| | e | + | + | Ø | Ø | Ø | + | + |
| | ẹ | + | Ø | + | + | + | Ø | + |
| | a | + | + | + | + | + | + | + |
| | ọ | + | Ø | + | + | + | Ø | + |
| | o | + | + | Ø | Ø | Ø | + | + |
| | u* | + | + | + | + | + | + | + |

Table 9:
Sequence of vowels in Yorùbá bisyllabic words. The symbols + and Ø indicate the permissible and non-permissible vowel sequence, respectively. $V_2$ is the second oral vowel in the word and $V_1$ indicates the vowel that may precede $V_2$

(Adapted from Archangeli and Pulleyblank (1989))
*The letter u cannot start a word in the Standard Yorùbá language.

These rules also apply to number naming in Yorùbá as illustrated with the following example: The number *ọgọ́ta* (120) is derived as 20 (*ogún*) multiplied by 3 (*ẹ̀ta*), i.e., *ogún ẹ̀ta*. The vowels *ún* and *ẹ̀* are then changed to vowel *ọ́* to form *ogọ́ta* by means of vowel coalescence. Since the last vowel in the last two syllables is *a*, which is a non-ATR, therefore, the immediately preceding vowels must be non-ATR. We will then proceed to check for harmony between the first two syllables. The second vowel *ọ́* is non-ATR, therefore, the first vowel *o* must also be a non-ATR. This will transform *o* to *ọ* by means of the vowel harmony.

### 3.4.4 Vowel assimilation

Vowel assimilation is a process whereby a vowel becomes completely or partially like another vowel (Akinlabí 2004). Vowel assimilation is most notable in Yorùbá numerals when a consonant separating 2 vowels is deleted. This can be illustrated by number 2,000 (*ẹgbàá*). The number 2,000 is actually formed from 200 × 10, i.e., *igba ẹ̀wá*, which will produce *ẹgbẹ̀wá* by vowel deletion. *ẹgbàá* is then formed by deleting the consonant *w* and allowing the vowel *ẹ* to assimilate the form of vowel *a*.

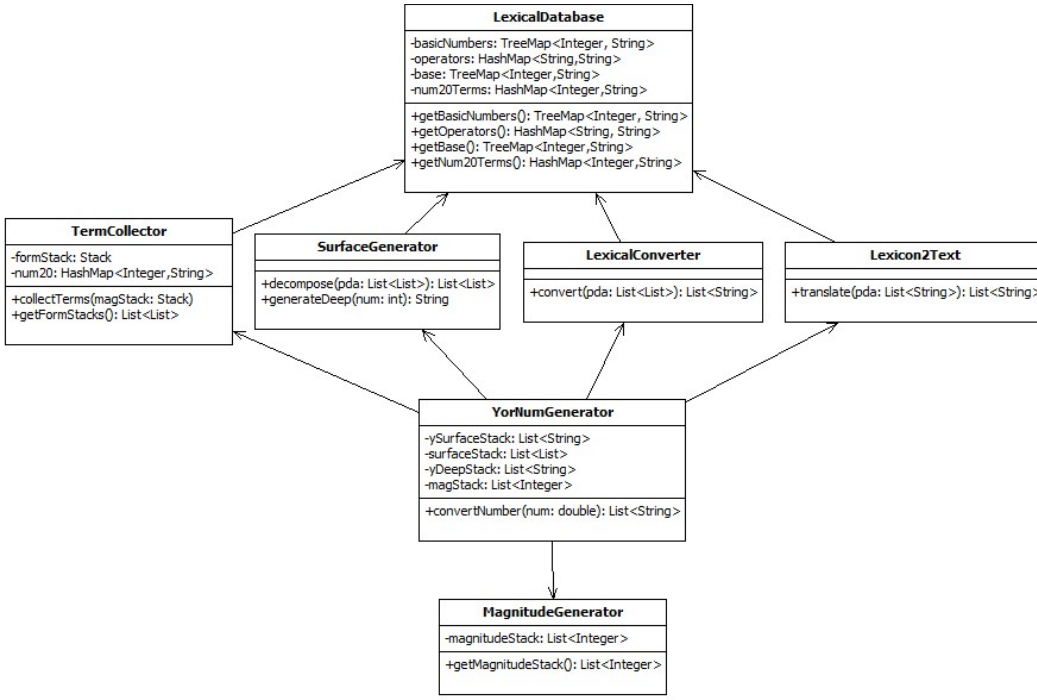



Figure 11: UML class diagram

Vowel assimilation can also occur between vowels separated by a consonant as in the expression for 800 (*ẹgbẹ̀rin*). This expression is derived as 200 (*igba*) multiplied by 4 (*ẹ̀rin*), i.e., *igba ẹ̀rin*. *igbẹ̀rin* is then formed by deleting *a*. The vowel *i* then assimilates *ẹ* to form *ẹgbẹ̀rin*.

### 3.5 *System and implementation*

An object oriented programming (OOP) approach with 7 classes was used during the system design. The UML class diagram and the sequence diagram for the software are as shown in Figure 11 and Figure 12 respectively.

The software implementation was done using Python and Java. The software was implemented following the specifications in the system design. The following software pieces were developed to demonstrate the conversion of numbers to Standard Yorùbá text:

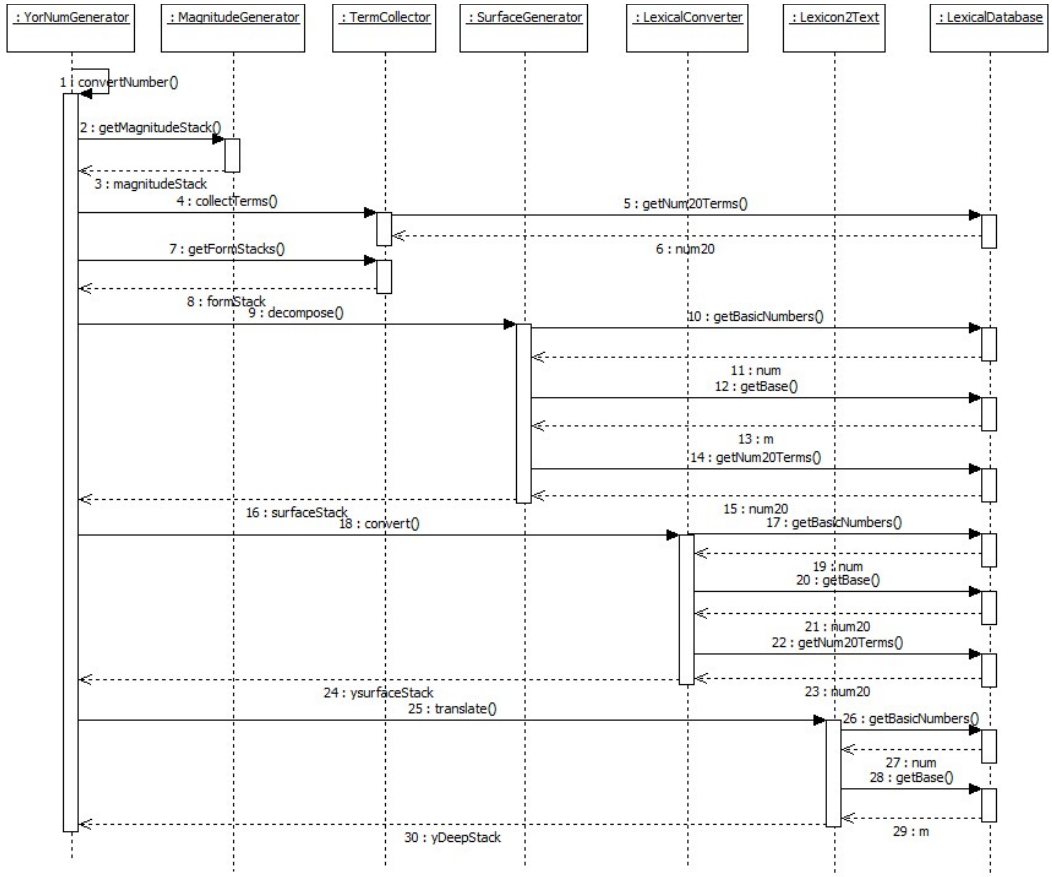



Figure 12: UML sequence diagram

a) **Desktop application:** The desktop application was implemented using PyQt in the Python programming language environment. The combination of Python and Qt makes possible the development of applications that are platform-independent (Summerfield 2008). NLTK (Loper and Bird 2002) was used to implement the grammar designed for the Yorùbá numeral system. It was also used to generate the parse trees of the number forms. The screenshot is as shown in Figure 13 and the software is available for download at `http://www.ifecisrg.org/yorubanumerals`.

b) **Web application:** The web application was implemented using the Google App Engine Python API. The screenshot is as shown in

Figure 14. The application is available at `http://www.num2yor.appspot.com`

c) **Mobile application on Android OS:** The mobile application was ported to Android using Java and the Android Application Development Toolkit (ADT). The screenshot is as shown in Figure 15.

The desktop application has a single document interface with toolbars for all tasks on top, a menu bar duplicating toolbar tasks, and a dockable history and analysis widgets. The analysis widget shows the computational details of a numeral structure. The Onka software has the following features:

a) The history can be saved for future usage.

b) Users can copy the output text to the computer's clipboard and paste it into an editing program or word processor.

c) The output of the software can be printed or saved in Unicode text format.

d) LaTeX users can copy or save the output in the LaTeX format. Also, the parse trees generated can be copied in the qtree (Siskind and Dimitriadis 2008) bracketed syntax for inclusion in TeX documents.

## 4 DISCUSSION

The software produces the correct lexical transcription for numbers in the Yorùbá language. In the following subsections, analysis will be carried out on the structure, computation, and forms of certain numbers. The numbers that will be considered are 240, 969, 19,669, and 40,000,000.

### 4.1 *The number 240*

The software processing of 240 produced two different forms, which are:

a. *òjìlélígba*: This number is computed by the addition of digit 40 to 200, i.e., [D40 + 200]. This representation uses only one addition operation. The parse tree of this representation is shown in Figure 16. This representation contains three terminal symbols, and the depth of the parse tree is 6.

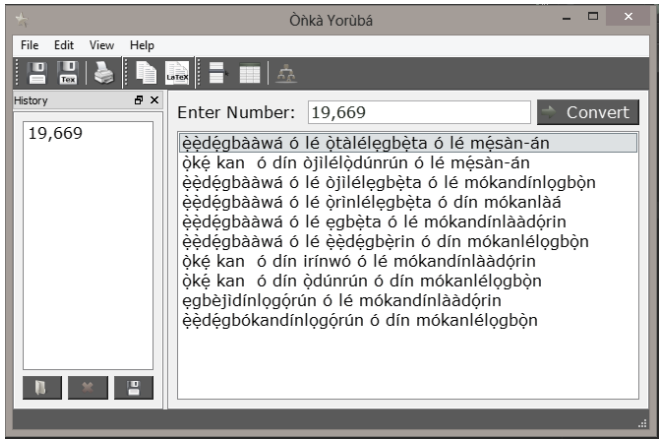



Figure 13: Screenshot of Òǹkà desktop application


Figure 14: Screenshot of Òǹkà web application hosted on Google App engine

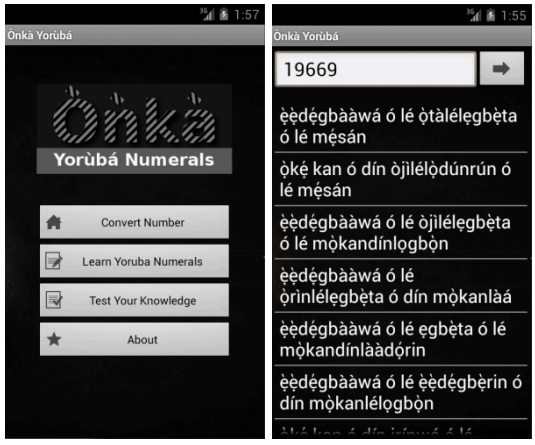



Figure 15: Screenshot of *Òǹkà* Android application

Figure 16:
Parse tree of *òjìlélígba*
(240 = 40 added to 200)
– representation 1

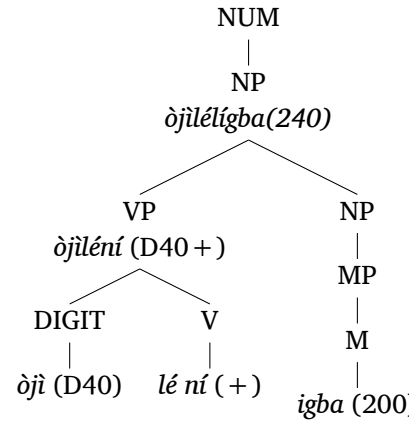
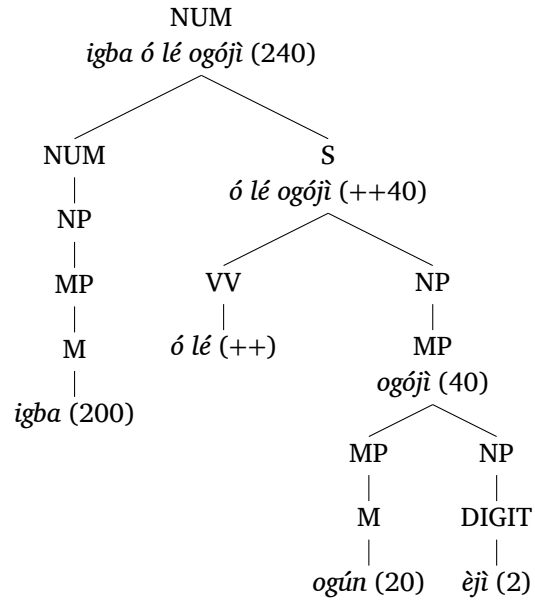



Figure 17:
Parse tree of *igba ó lé ogójì*
(240 = 200 increased by 40)
– representation 2




b. *igba ó lé ogójì*: This representation is presented as a phrase containing two number phrases. The parse tree of this representation is shown in Figure 17. This representation contains four terminal symbols, and the depth of the parse tree is 7.

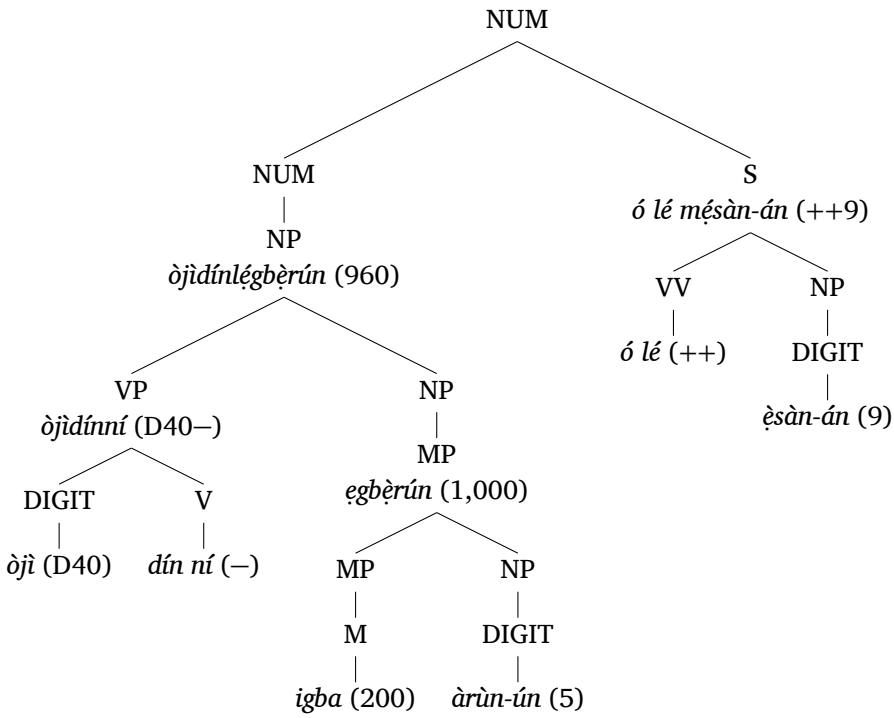



Figure 18: Parse tree for *òjìdínlẹgbẹ̀rún ó lé mẹ́sán* (969)

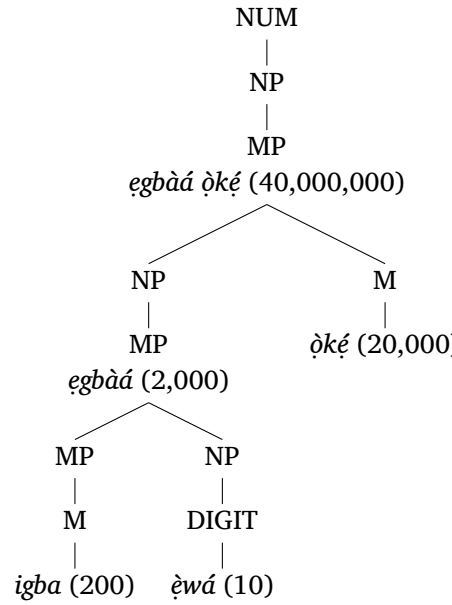
## 4.2 *The number 969*

The software gives 5 representations for number 969 as shown in Table 10. All these representations are valid and none has preference over others. The choice of a representation depends on the mental dexterity of the speaker. The parse tree for the first representation is shown in Figure 18.

Table 10: Representations of number 969

| No | Representation | Derivation |
|---|---|---|
| 1 | *òjìdínlégbèrún ó lé mésàn-án* | $((200 \times 5) - 40) + 9$ |
| 2 | *ọ̀tàdínlégbèrún ó lé mókandínlọgbọ̀n* | $((200 \times 5) - 60) + (-1 + 30)$ |
| 3 | *okòódínlẹgbẹ̀rún ó dín mókanlàá* | $((200 \times 5) - 20) - 11$ |
| 4 | *ẹ̀ẹ̀dégbèrún ó lé mókandínlààdọ́rin* | $((200 \times 5) - 100) + (((20 \times 4) - 10) - 1)$ |
| 5 | *ẹgbèrún ó dín mókanlélọgbọ̀n* | $(200 \times 5) - (1 + 30)$ |

Figure 19:
Parse tree of *ẹgbàá ọ̀kẹ́* (40,000,000)




### 4.3 *The number 19,669*

The output of the software for number 19,669 is shown in Figure 14. Representations 1 to 7 were presented by Ẹkúndayọ̀ (1977) and the developed software produced three more representations (8–10) that are structurally valid.

### 4.4 *The number 40,000,000*

The software gave one representation for 40,000,000 (*ẹgbàá ọ̀kẹ́*), which is derived as a multiple of 20,000 (i.e., $2{,}000 \times 20{,}000$). Next, 2,000 was derived as 200 in 10 places, i.e., $200 \times 10$. The parse tree is shown in Figure 19.

## 5 SYSTEM EVALUATION

In order to determine the accuracy of the system, we analysed and evaluated the output generated using the qualitative evaluation method. However, in these circumstances, it becomes expedient to rank the output of the software when multiple representations are produced. The aim is to order the representations according to the economy of computation.

## 5.1 *Number name ranking*

Although all representations produced are valid, we proposed some heuristic measures for ranking the representations when there are multiple correct expressions for a number. Once the parse tree had been generated for each representation, we computed the values to determine the computational economy of the numeral structure in the following order:

i) **The total number of terminal nodes ($t$):** This represents the number of basic lexical items that make up a Yorùbá numeral. The fewer the number of terminal nodes, the more economical the numeral structure is.

ii) **The height of the parse tree ($h$):** The height of the generated parse tree was determined by using the *height()* function of the package *nltk.tree*. The parse tree with the least height is thus considered the most suitable representation for a number.

iii) **The relative number of subtractions ($r$):** The most natural operations in most numeral systems are addition and multiplication, yet, the Yorùbá numeral system places a higher functional load on subtraction.
The value of $r$ is calculated by dividing the number of subtraction operations by the total number of arithmetic operations as shown in Equation 26. The two possible types of subtraction are the normal subtraction operation and the *ẹẹdín* type of subtraction.

$$r = \frac{Number\ of\ subtraction\ operations}{Total\ number\ of\ arithmetic\ operations} \tag{26}$$

This means that a lower $r$ implies a higher economy.

Once the first measure has been calculated and some structures have the same cost, the second measure, which checks the height of the parse tree in each structure, is used. But, if there is still a tie in values among any of the structures, the last measure (i.e., the relative number of subtraction) is used to determine the most suitable representation for a number. To illustrate this, we used these measures to decide which of the two representations for the number 240 discussed in Section 4.1 is more computationally economical. We started

by picking the representation with the minimum number of terminals. The parse tree in Figure 16 has three terminal symbols compared to four in Figure 17. Thus, the structure in Figure 16 is more computationally economical.

Also, the analysis of the ten representations for the number 19,669 is presented in Table 11. This shows the number of terminal symbols, the depth of the parse tree, and the arithmetic complexity. The computational cost was calculated based on these criteria, and it was used to rank the representations. The representations with the lowest number of terminal symbols and least height are representations 2 and 8 (with 8 terminal symbols and height of 8), however, representation 2 has the lesser relative number of subtractions. Hence, representation 2 (Figure 20) is the most computationally economical. Table 12 presents the most economical representations of selected numbers derived from the software.

Table 11: Representations for the number 19,669 and their ranks. **t** is the number of terminal symbols, **h** is the height of the parse tree as generated by the software, and **r** is the relative number of subtraction operations in the representation

| No | Rank | Yorùbá Text | t | h | r |
|---|---|---|---|---|---|
| 2 | 1 | *ọkẹ ó dín ojìlélọ́dúnrún ó lé mẹ́sán-án* | 8 | 8 | 0.250 |
| 8 | 2 | *ọkẹ ó dín ọ́dúnrún ó dín mọ́kanlélọ́gbọ̀n* | 8 | 8 | 0.500 |
| 7 | 3 | *ọkẹ ó dín irínwó ó lé mọ́kandínlaadọ́rin* | 10 | 8 | 0.500 |
| 10 | 4 | *ẹẹ́dégbọ̀kandínlọ́gọ́rún ó dín mọ́kanlélọ́gbọ̀n* | 10 | 10 | 0.500 |
| 1 | 5 | *ẹẹ́dẹ́gbàáwá ó lé ọ̀talélẹ́gbẹ̀ta ó lé mẹ́sán* | 11 | 9 | 0.143 |
| 9 | 6 | *ẹ́gbèjìdínlọ́gọ́rún ó lé mọ́kandínlaadọ́rin* | 11 | 10 | 0.429 |
| 6 | 7 | *ẹẹ́dẹ́gbàáwá ó lé ẹẹ́dẹ́gbẹ̀rin ó dín mọ́kanlélọ́gbọ̀n* | 12 | 9 | 0.375 |
| 3 | 8 | *ẹẹ́dẹ́gbàáwá ó lé ójìlélẹ́gbẹ̀ta ó lé mọ́kandínlọ́gbọ̀n* | 13 | 9 | 0.250 |
| 4 | 9 | *ẹẹ́dẹ́gbàáwá ó lé ọ̀rinélẹ́gbẹ̀ta dín mọ́kanlàá* | 13 | 9 | 0.250 |
| 5 | 10 | *ẹẹ́dẹ́gbàáwá ó lé ẹ́gbẹ̀ta ó lé mọ́kandínlaadọ́rin* | 13 | 9 | 0.333 |

### 5.2 *Qualitative evaluation*

The Mean Opinion Score (MOS) was used for the qualitative evaluation of the system. Chosen members of the staff of Ọ̀báfẹ́mi Awólọ́wọ̀ University, who are Yorùbá native speakers with adequate knowledge of the Yorùbá language and its orthography, were asked to provide the textual equivalences of some numbers in Yorùbá. Afterwards, their responses were compared to the output from the software.

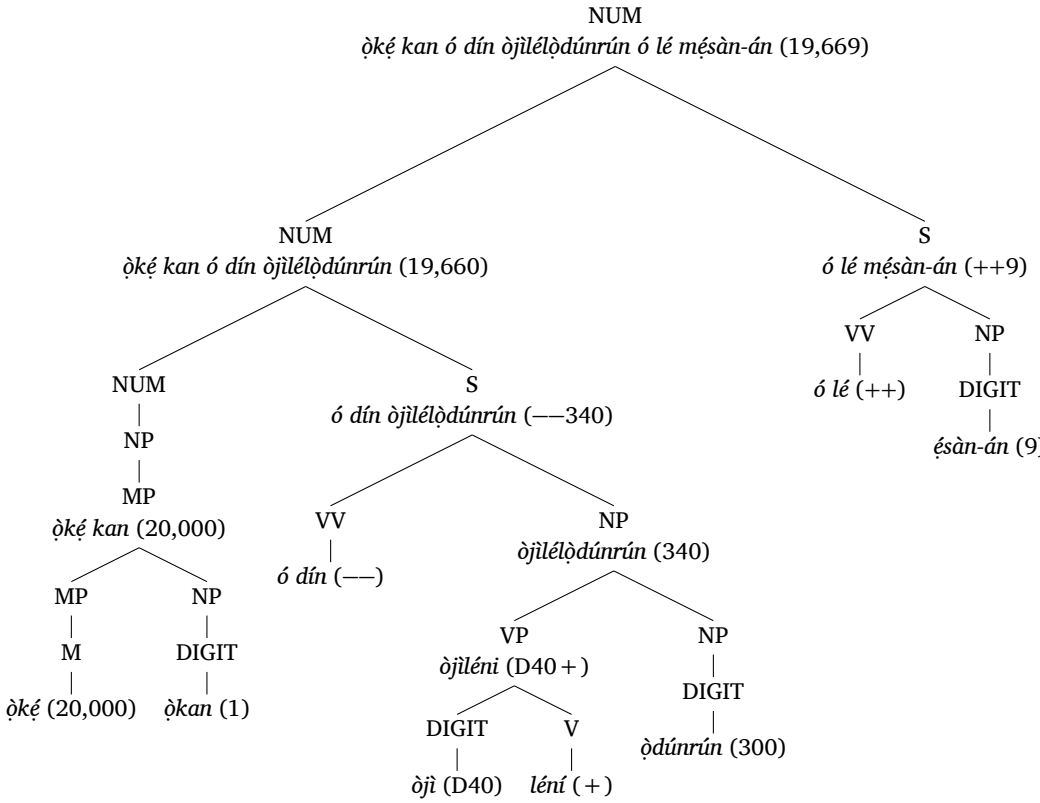



Figure 20: Parse tree for *ọ̀kẹ́ kan ó dín ọ̀ìlélọ̀dúnrún ó lé mẹ́sàn-án* (19,669)

Table 12: Software output for some numbers

| **Number** | **Yorùbá Text** |
|---|---|
| 182 | *ọgọ́sán ó lé méjì* |
| 187 | *ọgọ́sán ó lé méje* |
| 365 | *irinwó ó dín márùndínlógójí* |
| 595 | *ẹgbẹ̀ta ó dín márùn-ún* |
| 666 | *ọ̀tàlélẹgbẹ̀ta ó lé mẹfà* |
| 760 | *ọ̀ìdínlégbẹ̀rin* |
| 777 | *ẹ̀ẹ̀dégbẹ̀rin ó lé mẹ́tàdínlọ́gọ́rin* |
| 815 | *ẹgbẹ̀rin ó lé márùndínlógún* |
| 840 | *ọ̀ìlélégbẹ̀rin* |
| 905 | *ẹ̀ẹ̀dégbẹ̀rún ó lé márùn-ún* |
| 1,247 | *ọ̀ìlélégbèfà ó lé méje* |
| 600,000 | *ọgbọ̀n ọ̀kẹ́* |

A questionnaire was designed and administered to the selected group of 32 respondents. The numbers in the questionnaire were 25, 67, 132, 750, 969, 2,400, 3,000, 19,669, 20,000, 30,000, 1,000,000, and 400,000,000. The MOS evaluation was carried out to capture two important aspects of the Yorùbá numeral system. The first one was the ability of the respondents to give an accurate representation of the numbers in terms of value and orthography, and the second one was to obtain the most suitable representation for the numbers as provided by the respondents.

The numbers used in the questionnaires were chosen based on the following criteria:

i) Numbers 25, 67, and 132 were included to confirm that numbers between 1 and 200 have one standard lexical form.
ii) Numbers 750, 969, 2,400, and 19,669 were included to check whether the respondents are aware that there are multiple representations for these Yorùbá numerals.
iii) The number 20,000 was included to check whether the respondents find 20,000 as a single lexical item or think it is derived from the number 200.
iv) Numbers higher than 20,000 (30,000, 1,000,000, 400,000,000) were included to see if the respondents represent these numbers as multiples of 20,000 or in some other way.
v) Some structurally complex numbers (969 and 19,669) were added to see the most convenient combination of basic lexical numerals used by the respondents to derive these numbers.

The results of the analysis revealed that:

- For numbers 25, 67, and 132, all the respondents gave one correct representation, which matched up with the output from the software. This shows that numbers below 200 have one standard lexical form and that the skills needed to name these numbers are well understood.
- Ten respondents gave a representation for 19,669 but only two of them gave a correct number name (*ẹẹdẹgbàáwá ó lé ọtalélẹgbẹta ó lé mẹsán* and *ẹẹdẹgbàáwá ó lé ẹgbẹta ó lé mọkandínlaadọrin*). The other eight respondents provided number names that do not in any way evaluate to the number 19,669. Twenty two (22) of the

respondents did not give any representation for number 19,669. This shows that few respondents understand that 19,669 needs to be reconstructed and only two respondents were able to carry out the required computations. This result also shows that none of the respondents realised that multiple representations exist for the number 19,669.

- Seven of the respondents gave the correct number names for 20,000, with only two respondents using *ọ̀kẹ́*, and the remaining five using *ẹgbàawá*. This shows that few respondents were able to represent 20,000 in Yorùbá.
- Only three respondents gave the correct names for 1,000,000 (*ààdọ́ta ọ̀kẹ́*), and none of the respondents gave the correct representation for 400,000,000 (*ọ̀kẹ́ ọ̀nà ọ̀kẹ́*). This shows thatYorùbá native speakers may find the computations underlying naming large numbers cumbersome.

From these results, we conclude that the respondents were able to produce correct representations for numbers that are frequently used (number 1 to 200), although most of them were not able to produce names for higher numbers. After comparing the responses of the human evaluators with the system output, we recognise that the software out-performed the human evaluators. This affirms that most native-speakers know the terminologies needed for large numbers but are not familiar with the expression skills required for computing their number names. Without a doubt, modern Yorùbá speakers are losing the numeral generation skills embedded in their language. An obvious reason for this is the overwhelming use of the English numerals within the Yorùbá community.

## 6 CONCLUSION

In this paper, we discussed extensively the computational analysis of the Yorùbá numerals. We started by identifying the basic lexical numerals and the numeral groups. Then, we designed a CFG that was able to capture the structure of the Yorùbá numerals. Furthermore, we implemented a software for converting numbers to their textual equivalences in the Yorùbá language and generating their corresponding parse trees.

In this study, we are able to show that:

1. The Yorùbá number system has a systematic concept underlying it and that this concept can be articulated using modern computing tools and techniques.
2. The Yorùbá numeral system is not fully vigesimal. Elements of decimal (base 10) and quinary (base 5) are used in numeral representation.
3. The system's recall is 100% with respect to the corpus used in this study. This implies that, with carefully constructed computational model, the generation of the Yorùbá numeral system can be fully automated.
4. All the forms of number names produced were valid and the most computationally suitable representation are those in which : (a) the least number of terminal nodes is used, (b) the least height of the parse tree is generated, and (c) the least relative number of subtraction operations is involved. Though these measures are computationally reasonable, an interesting study will be to verify why Yorùbá native speakers sometimes prefer to adopt more complex methods, particularly when generating numerals greater that 200.

The results of this study can be applied in Yorùbá TTS. In any TTS system, numbers must be expanded into their textual forms before the actual speech synthesis is carried out. Thus, the system developed can serve as a sub-system of a Yorùbá TTS to handle the expansion of numbers to their textual equivalences. However, additional heuristic strategies must be employed by the TTS listeners to understand the number being spoken. Without a doubt, an increased usage of the Yorùbá numerals in communication could reduce the mental task needed for number conception.

The software developed in this study has a place in effective teaching and learning of the Yorùbá language. The software can be used in classes to teach the Yorùbá numeral system and its structure. This will allow the students to see the various forms possible for a single number and to visualise the structure (parse tree) of the numerals.

There are certain areas related to this study which we cannot explore. By pointing out these areas, we hope to focus our future study on

them. There is a need to carry out the contextual analysis of the Yorùbá numeral systems which will establish the relationships between numerals and their surrounding words. This will ensure that the expansion of numbers is carried out based on the context (cardinal, ordinal, nominal, currency, percentage, ratio, date, time, etc.) they represent. Also, there is a need to carry out a study on how the textual forms of the Yorùbá numerals could be recognised and converted to numbers. Definitely, the results of these studies could be applied in Yorùbá MT and information retrieval.

## ACKNOWLEDGEMENT

This work is supported by TETFUND Grant TETF/DESS/NRF/OAU/STI /VOL.1/B1.13.9. We would like to thank the editor and anonymous reviewers for their useful comments and suggestions to enhance the quality of the paper. We also acknowledge the support of the African Languages Technology Initiative (Alt-i).

## REFERENCES

Wándé Abímbọ́lá (1977), *Ifa Divinity Poetry*, Traditional African Literature, Nok Pub Intl, New York.

Roy Clive Abraham (1958), *Dictionary of Modern Yorùbá*, University of London Press, London.

Akinbiyi Akinlabí (2004), *Understanding Yorùbá Life and Culture*, chapter The Sound System of Yorùbá, pp. 453–468, Africa World Press, Trenton, NJ 08607.

Diana Archangeli and Douglas Pulleyblank (1989), Yorùbá Vowel Harmony, *Linguistic Inquiry*, 20(2):173–218.

Ọládélé Awóbùlúyì (1987), Towards a Typology of Coalescence, *Journal of West African Languages*, 17(2):5–22.

David Bailey and Jonathan Borwein (2011), The Greatest Mathematical Discovery, *manuscript: Available online:* `http://escholarship.org/uc/item/0sp6t6h5`.

Rẹ̀mí Bámiṣilẹ̀ (1994), Justification for the Survival of Vowel Coalescence as a Phonological Process in Yorùbá, *African Languages and Cultures*, 7(2):133–142.

Levi Leonard Conant (1896), *The Number Concept: Its Origin and Development*, MacMillan, New York.

Samuel Ẹ́KÚNDAYỌ̀ (1977), Vigesimal Numeral Derivational Morphology: Yorùbá Grammatical Competence Epitomized, *Anthropological Linguistics*, 19(9):436–453, http://www.jstor.org/stable/30027551.

Didier GOYVAERTS (1980), Counting in Logo, *Anthropological Linguistics*, 22(8):pp. 317–328, ISSN 00035483, http://www.jstor.org/stable/30027492.

James HURFORD (1975), *The Linguistic Theory of Numerals*, Cambridge University Press, Cambridge, ISBN 9780521133685.

James HURFORD (2001), Numeral Systems, in *International Encyclopedia of the Social & Behavioral Sciences*, pp. 10756–10761, Elsevier Science Ltd.

James HURFORD (2007), A Performed Practice Explains a Linguistic Universal: Counting Gives the Packing Strategy, *Lingua*, 117(5):773–783, doi:10.1016/j.lingua.2006.03.002, http://www.isrl.uiuc.edu/~amag/langev/paper/hurford06packingStrategy.html.

Samuel JOHNSON (1921), *The History of the Yorùbás: From the Earliest Times to the Beginning of the British Protectorate*, Routledge and Kegan Paul, London, reprinted 1966.

Edward LOPER and Steven BIRD (2002), NLTK: The Natural Language Toolkit, in *Proceedings of the ACL02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics*, volume 1, p. 8, http://arxiv.org/abs/cs/0205028.

Paul LOVEJOY and David TROTMAN (2003), *Trans-Atlantic Dimensions of Ethnicity in the African Diaspora*, Continuum: New York.

Adolphus MANN (1887), Notes on the Numeral System of the Yorùbá Nation, *The Journal of the Anthropological Institute of Great Britain and Ireland*, 16:59–64, available online: http://www.jstor.org/stable/2841738.

Karl MENNINGER (1969), *Number Words and Number Symbols: A Cultural History of Numbers*, MIT Press, Cambridge, translated by Paul Broneer for the revised German edition.

Ọdẹ́túnjí Àjàdí ỌDẸ́JỌBÍ (2003), Towards a Formal Specification of Some Computational Concepts in Yorùbá Thoughts, *ODU: Ifẹ̀ Journal of the Institute of Cultural Studies*, 8:87–110.

Kọ́lá OWÓLABÍ (2006), Yorùbá, *Encyclopedia of Language & Linguistics (Second Edition)*, pp. 735–738.

Thijs POLLMANN and Carel JANSEN (1996), The Language User as an Arithmetician, *Cognition*, 59:219–237.

Geoffrey SAXE (1981), Body Parts as Numerals: A Developmental Analysis of Numeration among the Oksapmin in Papua New Guinea, *Child Development*, 51(1):306–316, Blackwell Publishing on behalf of the Society for Research in Child Development.

Michael SIPSER (2007), *Introduction to the Theory of Computation*, Thomas Course Technology, India, $2^{nd}$ edition, ISBN 81-315-0162-0.

Jeffrey Mark SISKIND and Alexis DIMITRIADIS (2008), Qtree, a LaTeX Tree-drawing Package, Available online: `http://www.ling.upenn.edu/advice/latex/qtree/` (Accessed 19 September 2011).

Richard SPROAT (1996), Multilingual Text Analysis for Text-to-Speech Synthesis, in W. WAHLSTER, editor, $12^{th}$ *European Conference on Artificial Intelligence*, pp. 75–80, John Wiley & Sons, Ltd.

Mark SUMMERFIELD (2008), *Rapid GUI Programming with Python and Qt*, Prentice Hall, New Jersey, $1^{st}$ edition.

Helen VERRAN (2001), *Science and an African Logic*, University of Chicago Press, Chicago.

Claudia ZASLAVSKY (1973), *Africa Counts: Number and Pattern in African Cultures*, Lawrence Hill Books, $3^{rd}$ edition.

*This work is licensed under the Creative Commons Attribution 3.0 Unported License.*
`http://creativecommons.org/licenses/by/3.0/`