

Proceedings of FG-MoL 2005:  
The 10th conference on Formal  
Grammar  
and  
The 9th Meeting on  
Mathematics of Language  
Edinburgh

James Rogers (ed.)

5-7 August, 2005

CENTER FOR THE STUDY  
OF LANGUAGE  
AND INFORMATION



---

## Welcome

Welcome to FG/MoL05, the tenth conference on Formal Grammar and the ninth Meeting on Mathematics of Language. This year's conference includes nineteen papers covering, as usual, a wide range of areas of formal grammar and mathematical linguistics. In addition to the papers included in this volume there will be invited talks by

- Nicholas Asher, Department of Philosophy, University of Texas at Austin,
- Uwe Mönnich, Seminar für Sprachwissenschaft, Universität Tübingen,
- and Mark Steedman, School of Informatics, University of Edinburgh.

We are grateful to the Association for Computational Linguistics, the Natural Science and Engineering Research Council of Canada, The Institute for Research in Cognitive Science, University of Pennsylvania and Earlham College for financially supporting the conference. We are extremely grateful to ICCS/HCRC at University of Edinburgh for making this conference possible through local organization.

We are also all indebted to the Program Committee and outside referees for their labors in reviewing and ranking the submitted papers: Anne Abeille (Paris 7), Tilman Becker (DFKI), Pierre Boullier (INRIA), Gosse Bouma (Groningen), Chris Brew (Ohio State University), Wojciech Buszkowski (Poznan), Miriam Butt (Universitaet Konstanz), Tim Fernando (Trinity College, Dublin), Christophe Fougere (Paris 13), Nissim Francez (Haifa), Philippe de Groote (LORIA, Nancy), Aravind Joshi (UPenn), Makoto Kanazawa (National Institute of Informatics, Tokyo), Ruth Kempson (London), Andras Kornai (Metacarta), Uli Krieger (DFKI), Geert-Jan Kruijff (DFKI), Jonas Kuhn (University of Texas at Austin), Shalom Lappin (King's College, London), Alain Lecomte (Grenoble), Carlos Martin-Vide (Tarrag-

ona), Jens Michaelis (Universitaet Potsdam), Guido Minnen (Daimler-Chrysler AG), Mehryar Mohri (AT&T), Uwe Moennich (Universitaet Tuebingen), Michael Moortgat (Utrecht), Drew Moshier (Chapman), Larry Moss (Indiana), Stefan Mueller (Universitaet Bremen), Mark-Jan Nederhof (Rijksuniversiteit Groningen), Richard Oehrle (Berkeley, CA), Owen Rambow (Columbia), Christian Retore (INRIA & LaBRI, Bordeaux), Robert van Rooij (Amsterdam), Giorgio Satta (University of Padua), Ed Stabler (UCLA), Mark Steedman (Edinburgh), and Hans Joerg Tiede (Illinois Wesleyan).

And we are, of course, indebted to all of the authors who submitted papers to the meeting, both those that were accepted and those we could not fit, and to all of you, the participants in the meeting.

Enjoy the conference and enjoy Edinburgh.

Gerhard Jaeger, University of Bielefeld

Paola Monachesi, OTS Utrecht

Gerald Penn, University of Toronto

James Rogers, Earlham College

Shuly Wintner, University of Haifa

---

# Contents

Welcome    iii

Contributors    ix

- 1    **Underspecification and Neutrality: a unified approach to syncretism**    1  
BERTHOLD CRYSMANN
- 2    **From Semantic Restrictions to Reciprocal Meanings**    13  
SIVAN SABATO AND YOAD WINTER
- 3    **Events from temporal logic to regular languages with branching**    27  
TIM FERNANDO
- 4    **On the formal semantics of begin and end of states in a model theory for temporal DRT**    39  
PETRA DÜNGES
- 5    **How to Define Simulated Annealing for Optimality Theory?**    49  
TAMÁS BÍRÓ
- 6    **Finite Presentations of Pregroups and the Identity Problem**    61  
ALEXA H. MATER AND JAMES D. FIX

- 7 On rigid NL Lambek grammars inference from  
generalized functor-argument data 71**  
DENIS BÉCHET AND ANNIE FORET
- 8 Two Type 0-Variants of Minimalist Grammars 81**  
GREGORY M. KOBELE AND JENS MICHAELIS
- 9 Learnability of Some Classes of Optimal Categorical  
Grammars 93**  
JACEK MARCINIEC
- 10 An Additional Observation on Strict Derivational  
Minimalism 101**  
JENS MICHAELIS
- 11 Modular Grammar Design with Typed Parametric  
Principles 113**  
RALPH DEBUSMANN, DENYS DUCHIER AND ANDREAS  
ROSSBERG
- 12 What feature co-occurrence restrictions have to do  
with type signatures 123**  
WIEBKE PETERSEN AND JAMES KILBURY
- 13 Bias decreases in proportion to the number of  
annotators 139**  
RON ARTSTEIN AND MASSIMO POESIO
- 14 The Proper Treatment of Coordination in Peirce  
Grammar 149**  
HANS LEISS
- 15 Strong connectivity hypothesis and generative  
power in TAG 167**  
ALESSANDRO MAZZEI, VINCENZO LOMBARDO AND  
PATRICK STURT
- 16 Inessential Features, Ineliminable Features, and  
Modal Logics for Model Theoretic Syntax 183**  
HANS-JÖRG TIEDE

- 17 Well-Nested Drawings  
as Models of Syntactic Structure 195**  
MANUEL BODIRSKY, MARCO KUHLMANN AND  
MATHIAS MÖHL
- 18 A Linearization-based approach to Gapping 205**  
RUI P. CHAVES
- 19 Further Properties of Path-Controlled  
Grammars 219**  
CARLOS MARTÍN-VIDE AND VICTOR MITRANA
- 20 Scope-marking constructions in type-logical  
grammar 231**  
WILLEMIJN VERMAAT





---

## Contributors

- BERTHOLD CRYSMANN DFKI GmbH and Saarland University
- SIVAN SABATO Technion - Israel Institute of Technology  
Computer Science Department
- YOAD WINTER Technion - Israel Institute of Technology  
Computer Science Department
- TIM FERNANDO Computer Science Department  
Trinity College  
Dublin 2, Ireland
- PETRA DÜNGES Computational Linguistics  
University of the Saarland  
`duenges@mail.coli.uni-sb.de`,  
`http://www.coli.uni-saarland.de/~duenges/`
- TAMÁS BÍRÓ Humanities Computing, CLCG  
University of Groningen  
`t.s.biro@rug.nl`, `birot@nytud.hu`  
`http://www.let.rug.nl/~birot`
- ALEXA H. MATER University of Pennsylvania  
Department of Mathematics  
`matera@alumni.reed.edu`
- JAMES D. FIX Reed College, Mathematics Department  
`jimfix@reed.edu`
- DENIS BÉCHET LINA—CNRS and Université de Nantes  
`Denis.Bechet@univ-nantes.fr`
- ANNIE FORET IRISA—Université de Rennes1  
`Annie.Foret@irisa.fr`

x / FG-MoL 2005

- GREGORY M. KOBELE University of California Los Angeles  
Department of Linguistics  
kobele@humnet.ucla.edu
- JENS MICHAELIS Universität Potsdam, Institut für Linguistik  
michaelis@ling.uni-potsdam.de
- JACEK MARCINIEC Faculty of Mathematics and Computer Science  
Adam Mickiewicz University  
Poznań, Poland  
jacmar@amu.edu.pl
- RALPH DEBUSMANN Programming Systems Lab  
Saarland University, Saarbrücken, Germany
- DENYS DUCHIER IRI and LIFL—CNRS, Lille, France
- ANDREAS ROSSBERG Programming Systems Lab  
Saarland University, Saarbrücken, Germany
- WIEBKE PETERSEN Computational Linguistics  
Institute of Language and Information  
University of Düsseldorf, Germany  
petersew@uni-duesseldorf.de
- JAMES KILBURY Computational Linguistics  
Institute of Language and Information  
University of Düsseldorf, Germany  
kilbury@ling.uni-duesseldorf.de
- RON ARTSTEIN Department of Computer Science  
University of Essex  
artstein [at] essex.ac.uk
- MASSIMO POESIO Department of Computer Science  
University of Essex  
poesio [at] essex.ac.uk
- HANS LEISS Universität München  
Centrum für Informations und Sprachverarbeitung
- ALESSANDRO MAZZEI Dip. Informatica, Università di Torino  
mazzei@di.unito.it
- VINCENZO LOMBARDO Dip. Informatica, Università di Torino  
vincenzo@di.unito.it
- PATRICK STURT Dep. of Psychology, University of Glasgow  
patrick@psy.gla.ac.uk

- HANS-JÖRG TIEDE Dept. of Mathematics and Computer Science  
Illinois Wesleyan University  
Bloomington, IL, USA  
`htiede@iwu.edu`
- MANUEL BODIRSKY Institut für Informatik  
Humboldt-Universität zu Berlin
- MARCO KUHLMANN Department of Computer Science  
Saarland University, Saarbrücken, Germany
- MATHIAS MÖHL Department of Computer Science  
Saarland University, Saarbrücken, Germany
- RUI P. CHAVES Computation of Lexical and  
Grammatical Knowledge Research Group  
CLUL  
Complexo Interdisciplinar da Universidade de Lisboa  
Lisbon, Portugal  
`ruichaves@clul.ul.pt`
- CARLOS MARTÍN-VIDE Research Group in Mathematical Linguistics  
Rovira i Virgili University, Tarragona, Spain  
`carlos.martin@urv.net`
- VICTOR MITRANA Faculty of Mathematics and Computer Science  
University of Bucharest  
and Research Group in Mathematical Linguistics  
Rovira i Virgili University, Tarragona, Spain  
`vmi@urv.net`
- WILLEMIJN VERMAAT UiL-OTS (Utrecht Institute of Linguistics)  
Utrecht University, The Netherlands  
`Willemijn.Vermaat@let.uu.nl`



---

# Underspecification and Neutrality: a unified approach to syncretism

BERTHOLD CRYSMANN <sup>†</sup>

## Abstract

In this paper I discuss the phenomenon of syncretism in German and show that current type-based approaches are unable to combine the treatment of feature indeterminacy with the virtues of underspecification. I will then propose a revised organisation of the inflectional type hierarchies suggested by Daniels (2001), drawing on a systematic distinction between inherent and external (case) requirements. Finally, I will show how likeness constraints operating over a subset of the inflectional dimensions can be expressed by means of typed lists that abstract out the relevant dimension from the combined case/number/gender hierarchies suitable for syncretism.

**Keywords** SYNCRETISM, UNDERSPECIFICATION, INDETERMINACY, HPSG, GERMAN

Nouns, adjectives and determiners in German inflect for case, number and gender. However, as is typical for inflectional languages, these morphosyntactic feature dimensions are not expressed by discrete, individually identifiable affixes. Rather, affixes realise complex feature combinations. Although four case, three gender and two number specifications can clearly be distinguished, the morphological paradigms of the language are characterised by heavy syncretism. Often, syncretism can-

---

<sup>†</sup>I would like to thank Stefan Müller and Michael Jellinghaus for fruitful discussion of several aspects of this work. Many thanks also to the three anonymous reviewers for their invaluable comments.

The work presented in this article was partially supported by research grants from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI project Quetal (FKZ 01 IW C02).

not be resolved to disjunctive specification or underspecification within a single feature, but it cuts across the three inflectional dimensions. However, since disjunctions are in general much harder to process than type inference, type-based underspecification of case/number/gender specifications appears to be the key towards an efficient and concise treatment of syncretism.<sup>1</sup>

Ambiguous nominal forms in German are also subject to indeterminacy. Again, indeterminacy is not restricted to individual inflectional dimensions, but rather follows the patterns of syncretism. Although the notions of ambiguity and indeterminacy are intimately related, there is currently no analysis at hand that is capable of combining the machinery necessary to cover feature indeterminacy with the benefits of underspecification.

In this paper I will propose an entirely type-based approach to syncretism that will successfully reconcile Daniels (2001)'s approach to feature indeterminacy with morphosyntactic underspecification across features. Furthermore, I will show how list types can be fruitfully put to use to abstract out individual featural dimensions from combined case/number/gender type hierarchies, permitting the expression of likeness constraints in coordinate structures. As a result, the current proposal presents an entirely disjunction-free approach to syncretism, addressing indeterminacy, underspecification and likeness constraints.

### 1.1 Feature neutrality

It has been argued by Ingria (1990) that the phenomenon of feature neutrality in coordination constitutes a severe challenge for unification-based approaches to feature resolution and concludes that unification should rather be supplanted by feature compatibility checks.

- (1) Er findet und hilft Frauen.  
he finds.A and helps.D women.A/D  
'He finds and helps women.'
- (2) \*Er findet und hilft Kindern.  
he finds.A and helps.D children.D
- (3) \*Er findet und hilft Kinder.  
he finds.A and helps.D children.A

Unification-based frameworks such as LFG or HPSG have taken up the challenge, refining the representation of feature constraints in such

---

<sup>1</sup>See the Surrey Morphology Group syncretism database for a cross-linguistic overview (<http://www.surrey.ac.uk/LIS/SMG/>).

a way that neutrality can be modelled without any substantial changes to the underlying formalism. For HPSG, Daniels (2001) proposed to address these problems by means of enriching the type hierarchy to include neutral types, an idea originally due to Levine et al. (2001).<sup>2</sup>

Daniels (2001) has also discussed cases where the potential for feature indeterminacy does not only involve the values of a single feature: as illustrated in (4), a masculine noun like *Dozenten* can express any cell of the case/number paradigm except nominative singular. Accordingly, one and the same form can be subject to feature indeterminacy regarding number, gender, or even case.

- (4)    der Antrag des            oder der            Dozenten  
       the petition Def.G.Sg or    Def.G.Pl lecturer.G/D/A+N.Pl  
       ‘the petition of the lecturer(s)’
- (5)    der            oder die            Abgeordnete  
       Def.N.M.Sg or    Def.N.F.Sg representative.N.Sg.M/F  
       ‘the male or female representative’
- (6)    Er findet    und hilft    Dozenten.  
       he finds.A and helps.D lecturers.A/D  
       ‘He finds and helps lecturers.’

A determiner like *der* is neutral between nominative singular masculine and genitive/dative plural. However, indeterminacy with respect to number is not independent of case, as illustrated by (7), where the unavailability of a nominative singular reading for *Dozenten* is responsible for the illformedness of the sentence.

- (7)    \* der                                    Dozenten                                    ist hier  
       the.N.Sg.M+G/D.Sg.F+G.Pl lecturer.G/D/A+N.Pl is here

To incorporate the issue of neutrality across features, Daniels suggests to combine values of different inflectional features into an overarching type hierarchy, the nodes of which are essentially derived by building the Cartesian product of the types within each inflectional dimension.

## 1.2 Underspecification

Combined type hierarchies across different inflectional feature dimensions have also been fruitfully put to use in the context of efficient

---

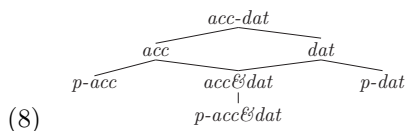
<sup>2</sup>Within LFG, a technically different, though conceptually similar approach has been developed by Dalrymple and Kaplan (2000). See Levy and Pollard (2001) for a comparison.

grammar engineering. In the LinGO ERG (Flickinger, 2000), person and number are represented as values of a single feature PNG, permitting the expression of, e.g., non-3rd-singular agreement without the use of negation or disjunction.

In the context of more strongly inflecting languages, such as German, where syncretism is the norm rather than the exception, underspecification of inflectional features across different dimensions is even more pressing: a typical noun such as *Computer* can express any case/number combination, except genitive singular and dative plural, i.e. 6 in total. Using combined case/number/gender hierarchies, the syncretism between nominative/dative/accusative singular and nominative/genitive/accusative plural can be represented compactly as one entry. The very same holds for German determiners and adjectives. Intuitively, it would make perfect sense to try and exploit the combined type hierarchies required for the treatment of neutrality in order to arrive at a more concise and efficient representation of syncretism.

### 1.3 The Problem

Although both feature indeterminacy and ambiguity do call for type hierarchies combining different inflectional dimensions, these two approaches have not yet received a unified treatment to date: it has been recognised as early as Zaenen and Karttunen (1984) that in unification-based formalisms feature neutrality cannot be reduced to underspecification. The apparent incompatibility of neutrality and underspecification is even more surprising, as these two notions are intimately related: i.e., the ambiguity of a form between two values is a necessary prerequisite for this form to be embeddable in a neutral context.



Taking as starting point the case hierarchy proposed by Daniels (2001), one might be tempted to assign a case-ambiguous form like ‘Frauen’ a supertype of both *acc* and *dat*, e.g. *acc-dat*, which can be resolved to *p-acc* (‘die Frauen’) or *p-dat* (‘den Frauen’), depending on context. However, to include feature-neutrality, it must also be possible to resolve it to the neutral type *acc&dat*. Suppose now that a form like *die* ‘the’ is itself ambiguous, i.e. between nominative and accusative, representable by a type *nom-acc*, again a supertype of *acc*. Unification of the case values of *die* ‘the’ and *Frauen* ‘women’ will yield *acc*, which



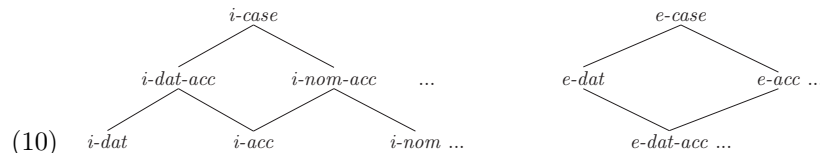
will still be a supertype of the neutral type  $acc\mathcal{E}dat$ , erroneously licensing the unambiguously non-dative *die Frauen* ‘the women’ in the neutral accusative/dative context of *findet und hilft* ‘finds and helps’.

- (9) \* Er findet und hilft [die Frauen]  
 he finds.A and helps.D [the women].A

Thus, under Daniels’s account, lexical items are explicitly assigned leaf type values, so-called “pure types”. While successful at resolving the issue of indeterminacy, this approach in fact drastically increases the amount of lexical ambiguity, having to postulate distinct entries for type-resolved pure accusative, pure dative, pure nominative, pure genitive, as well as all pair-wise case-neutral variants of a single form like *Frauen* ‘women’. Ideally, all these different readings should be representable by a single lexical entry, if only underspecification could be made to work together with indeterminacy.

#### 1.4 A Solution

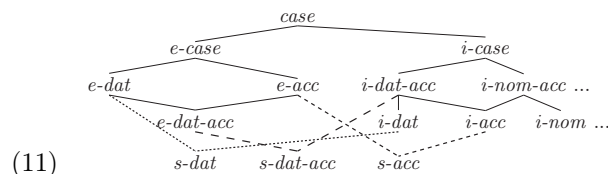
The reason for the apparent incompatibility of underspecification and feature neutrality lies with the attempt to address both aspects within a single type hierarchy. Instead, I shall argue to draw a principled distinction between inherent inflectional feature values, where unification specialises from underspecified or ambiguous types to unambiguous types, and external or subcategorised feature values where unification proceeds from non-neutral, though generally unambiguous to neutral types. As a result we will have two partially independent hierarchies, one for ambiguity (*i-case*) and an inverse one for neutrality (*e-case*).<sup>3</sup>



Inherent case specifications of dependents will be types in the *i-case* subhierarchy (for inherent case), whereas case requirements imposed by a subcategorising head will be values in the *e-case* subhierarchy (for external case). Unification of internal case specifications will result in disambiguation of underspecified case values, whereas unification of external case requirements will result in feature indeterminacy. To illustrate this, take the examples in (1) and (2): case ambiguous *Frauen*

<sup>3</sup>In essence, the inverse layouts of the two subhierarchies correspond quite closely to the different behaviour of functor and argument categories with respect to strengthening/weakening in the approach of Bayer and Johnson (1995).

will be specified *i-dat-acc*, whereas unambiguous *Kindern* will carry the more specific value *i-dat*. Likewise, the verbs *finden* and *helfen* will subcategorise for an *e-acc* and *e-dat* complement, respectively. Coordination of the two lexical verbs will lead to unification of CAT values (Pollard and Sag, 1994),<sup>4</sup> and hence, valence lists, “overspecifying” the case requirement as *e-dat-acc*.



In order to permit satisfaction of any subcategorised case by some inherent case, all we need to do is define the greatest lower bound for any pair of internal and external case specification.

Thus, underspecified internal cases will unify with a corresponding neutral case, whereas specific internal cases will only unify with their corresponding non-neutral cases. As depicted above, more specific types in one hierarchy will be compatible with less specific types in the other, and vice versa. Returning to our example above, underspecified *i-dat-acc*, as in *Frauen* unifies with overspecified *e-dat-acc*, as required by the coordination *findet und hilft*, whereas unambiguous *Kindern* does not, since no greatest lower bound is defined for *i-dat* and *e-dat-acc*. Thus, disambiguation of *i-case* values will always reduce the potential for neutrality, as required. On a more conceptual level, these cross-classifications between the two hierarchies embody the logical link between underspecification and neutrality.

### 1.5 Likeness constraints in coordination

It has been argued by Müller (p.c.) that one of the main obstacles for exploiting combined case-number-gender hierarchies to provide an entirely disjunction-free representation of German syncretism surfaces in certain coordinate structures. It is a well-known fact about German that likeness of category in coordinate structures includes likeness of case specification, but excludes, as a rule, requirements concerning the likeness of gender or number specifications in the conjuncts, a pattern which is quite neatly predicted by HPSG’s segregation of HEAD features and INDEX features. However, in free word order languages like German, case arguably serves not only a categorial function, but also a

<sup>4</sup>For an overview of the treatment of coordination in HPSG, see Crysmann (to appear).

semantic one, thereby supporting the originally morphological motivation towards organising all agreement features into a single hierarchy (see also Kathol (1999) for a similar proposal). Moreover, the mere existence of indeterminacy across case and index features makes combined hierarchies almost inevitable.

Müller discusses syncretive pronominals in German, such as *der*, which is ambiguous, *inter alia*, between nominative singular masculine, as shown in (12), and dative singular feminine, as illustrated in (13).

(12) Der schläft.  
the.N.S.M sleeps  
'That one sleeps.'

(13) Ich helfe der.  
I help the.D.S.F  
'I help that one.'

This ambiguity could be represented by a type  $n-s-m+d-s-f$ .<sup>5</sup> Subcategorisation for nominative singular (type  $n-s-g$ ) or dative (type  $d-n-g$ ) will disambiguate these forms accordingly.<sup>6</sup>

In coordinate structures, however, we observe that likeness of case equally eliminates one of the possible gender specifications for *der*, as witnessed by the disambiguation (14). Thus, we must be able to distribute the case requirement over the two conjuncts in such a way that it can exert its disambiguatory potential, without actually unifying the entire case/number/gender specifications of the two conjuncts.

(14) Ich helfe der und dem Mann.  
I help the.D.S.F and the.D.S.M man  
'I help this one and the man.'

In Daniels (2001), this problem was partly anticipated: he suggests to address the issue of likeness of case by means of a relational constraint *same-case/2*, which restricts the two arguments to satisfy identical type requirements. This type equality is essentially imposed by disjunctive enumeration of the four possible subcategorised case values. In typed

---

<sup>5</sup>As a convention, I am using the following nomenclature of combined c(ase)-n(umber)-g(ender) types: the three inflectional dimensions are specified in the above order, separated by a hyphen. In the first slot, *c* represents the most general case "value", *n, g, d, a* the most specific. "Disjunctive values" are represented as combinations of case specifications. The very same holds number and gender specifications.

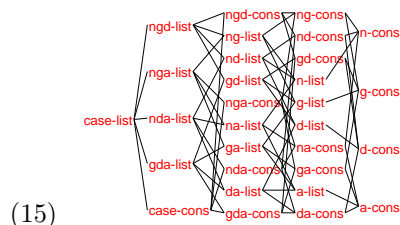
<sup>6</sup>For ease of exposition, I am abstracting away from the internal/external distinction, which is immaterial here, since we are only dealing with underspecification, not indeterminacy.

feature formalisms without relational constraints, his solution may be mimicked by means of unfolding the relevant phrase structure schemata into case-specified variants. In both cases, a greater part of the efficiency gains achieved by underspecification may get eaten up by this disjunctive approach to case similarity.

An alternative, though not fully satisfactory solution would involve retaining a HEAD feature CASE along-side the combined AGR feature. While this move will be at least effective in ruling out unacceptable surface strings, it will fail to impose the disambiguation potential of the subcategorising head onto the individual conjuncts.

What is really needed here is a data structure that may serve to both express the appropriate case-requirements in terms of a combined hierarchy, and permit arbitrarily many specific instantiations of the case constraint. Fortunately, typed feature formalisms do provide for such a data structure, namely typed lists.

To start with, we will set up a hierarchy of case list types, as depicted in figure (15)<sup>7</sup>, where each list type immediately subsumes at least one subtype representing a non-empty list of the same case type.



Types in the combined case-number-gender hierarchy will now restrict their CASE value to an appropriate list type, as given in (16).<sup>8</sup>

$$(16) \quad nda-n-g \rightarrow [CASE \quad nda-list]$$

Non-empty case lists bear a type constraint restricting the FIRST value to the corresponding agreement type in the combined case/number/gender hierarchy. Actually, thanks to type inference in the hierarchy of case lists, we only need to do this for the 4 immediate subtypes of *case-cons*, namely *ngd-cons*, *nga-cons*, *nda-cons*, and *gda-cons*. In order to propagate the case specification onto all elements of the open list, the tail is constrained to the corresponding list type (see (17)).

<sup>7</sup>The type hierarchy has been exported from the LKB: supertypes are on the left, subtypes are on the right.

<sup>8</sup>Recall that, according to our naming convention, the type *nda-n-g* represents all case specification except genitive. Number and gender are fully underspecified.

$$(17) \quad nda-cons \rightarrow \langle nda-n-g \mid nda-list \rangle$$

Now that we have a data structure that enables us to encode likeness of case for arbitrary instances of case/number/gender types, all we need to do is refine our existing coordination schemata to distribute the case restriction imposed on the coordinate structure onto the individual conjuncts. In the implemented German grammar we are using, coordinate structures are licensed by binary phrase structure schemata. Thus, all we have to do is to constrain the AGR feature of the left conjunct daughter to be token-identical to the first element on the mother's AGR|CASE list, and percolate the rest of this list onto the (recursive) righthand conjunct daughter's AGR|CASE value:

$$(18) \quad coord-phr \rightarrow \left[ \begin{array}{l} SS \mid L \mid AGR \mid CASE \langle \boxed{1} \mid \boxed{2} \rangle \\ \text{COORD-DTRS} \left\langle \left[ \begin{array}{l} SS \mid L \mid AGR \boxed{1} \\ SS \mid L \mid AGR \mid CASE \boxed{2} \end{array} \right] \right\rangle \end{array} \right]$$

Coordinating conjunctions, which combine with a conjunct by way of a head-complement rule, will equate their own AGR|CASE|FIRST value with the AGR value of their complement, percolating the case constraint onto the last conjunct.

$$(19) \quad \left[ \begin{array}{l} SS \mid L \left[ \begin{array}{l} AGR \mid CASE \langle \boxed{1} \mid list \rangle \end{array} \right] \\ VAL \mid COMPS \left\langle \left[ \begin{array}{l} L \mid AGR \boxed{1} \end{array} \right] \right\rangle \end{array} \right]$$

Besides coordination, the current approach to likeness constraints across syncretive forms can also be applied to case/gender agreement in German constructions involving the phrase *ein- nach d- anderen* 'one after the other', a set of phenomena discussed by Höhle (1983) and Müller (1999):

$$(20) \quad \text{Wir}_i \quad \text{helfen ihnen}_j \quad [\text{einem} \quad \text{nach dem} \quad \text{anderen}]_{*i/j}$$

we.NOM help them.dat one.DAT.M after the.M other  
'We help them one after the other.'

$$(21) \quad \text{Wir}_i \quad \text{helfen ihnen}_j \quad [\text{einer} \quad \text{nach der} \quad \text{anderen}]_{*i/j}$$

we.NOM help them.dat one.DAT.F after the.F other  
'We help them one after the other.'

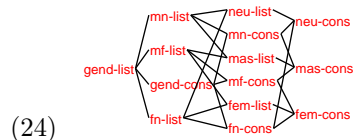
$$(22) \quad \text{Wir}_i \quad \text{helfen ihnen}_j \quad [\text{einer} \quad \text{nach dem} \quad \text{anderen}]_{i/*j}$$

we.NOM help them.dat one.NOM.M after the.M other  
'We help them one after the other.'

- (23) Wir<sub>i</sub> helfen ihnen<sub>j</sub> [eine nach der anderen]<sub>i/\*j</sub>  
 we.NOM help them.DAT one.NOM.F after the.F other  
 ‘We help them one after the other.’

As illustrated by the data in (20–23) above, agreement between antecedent and the phrase *ein- nach d- anderen* ‘one after the other’ proceeds along two inflectional dimensions: case and gender. Within the phrase *ein- nach d- anderen-*, we find gender agreement between the two pronominal *ein-* and the NP *d- anderen*. Case of the latter is invariantly dative, since it is governed by the preposition *nach*. The important aspect of this construction now is that the gender agreement between the pronominals partially disambiguates the case specification: e.g., the pronominal *einer* displays syncretism between nominative masculine and dative feminine (singular). As witnessed by the contrasts in (21) and (22), disambiguation of case syncretism by means of grammatical gender reduces the semantic attachment potential of the entire phrase, precluding attachment to the subject in (21), and to the object in (22).

The situation we encounter here is actually highly parallel to the one we found earlier with likeness of case in coordinate structures: again, agreement only targets a subset of the inflectional dimensions (case and gender) to the exclusion of others (person and number). What is therefore needed, is, again, a mechanism to abstract out the relevant dimensions from our syncretism types. While we can directly reuse our list-valued CASE feature to implement case agreement, we have to provide an analogous abstraction of the gender dimension, a step, which is very much straightforward:



$$(25) \quad c-n-mn \rightarrow \left[ \text{GEND} \quad mn\text{-list} \right]$$

$$(26) \quad mn\text{-cons} \rightarrow \left\langle c-n-mn \mid mn\text{-list} \right\rangle$$

Again, we need a hierarchy of list types, and connect it — via type constraints — to appropriate types in the combined *c-n-g* hierarchy.

Having established the required abstraction of gender alongside case, we are now in a position to capture the interaction of case and gender agreement. All it needs, is to require that, in the phrase *ein- nach d- anderen*, the PP *nach d- anderen*, which exhibits gender agreement with the pronoun *ein-*, will equate the first element of its GEND list

with the AGR value of *ein-*, either constructionally, or via a selection feature, e.g. MOD.

As a result, the entire AGR value of *ein-* will be disambiguated to a *c-n-g* specification compatible with the PP's gender. The AGR value of the entire construction, which represents an aggregate of singular entities, will be the unification of a constructional plural specification (*c-p-g*) with the first elements on both CASE and GEND of *ein-*. This AGR value will then be unified with that of the antecedent.<sup>9</sup>

$$(27) \left[ \begin{array}{l} \text{SS | L | AGR } c-n-p \wedge \boxed{1} \wedge \boxed{2} \\ \left. \begin{array}{l} \left[ \begin{array}{l} \text{PH } \langle \text{einer} \rangle \\ \text{SS | L | AGR } n-s-mn+d-s-f \wedge \boxed{3} \left[ \begin{array}{l} \text{CASE | FIRST } \boxed{1} \\ \text{GEND | FIRST } \boxed{2} \end{array} \right] \end{array} \right] \\ \text{DTRS } \left\langle \left[ \begin{array}{l} \text{PH } \langle \text{nach der anderen} \rangle \\ \text{SS | L | AGR } d-s-f \wedge \left[ \begin{array}{l} \text{CASE } d\text{-list} \\ \text{GEND } \left[ \begin{array}{l} fem\text{-cons} \\ \text{FIRST } \boxed{3}c\text{-n-f} \end{array} \right] \end{array} \right] \end{array} \right] \end{array} \right\rangle \end{array} \right] \end{array} \right.$$

To conclude, we have seen that the approach to likeness of case in coordinate structures can be extended, in a principled way, to other phenomena displaying partial agreement, i.e. agreement involving only a subset of inflectional dimensions. Furthermore, as illustrated by our analysis of the overlapping of gender and case agreement, the combination of dimensions in partial agreement can essentially be reduced to abstracting out each dimension individually and having them interact by means of unification.

## 1.6 Conclusion

In this paper we have argued for an extension to Daniels (2001) original approach to feature indeterminacy in HPSG which makes it possible to combine the empirical virtues of his type-based approach to the phenomenon with the advantages of underspecified representation of syncretism across features, namely generality of specification and efficiency in processing. We have further shown how likeness constraints abstracting out a particular inflectional dimension from a combined inflectional type hierarchy can still be expressed concisely by means of typed lists.

<sup>9</sup>In order to make the lexical specification of case/number/gender information more transparent, I have left the unification of values in (27) unresolved.

## References

- Bayer, S. and M. Johnson. 1995. Features and agreement. In *Proceedings of the 33rd Annual Meeting of the ACL*, pages 70–76.
- Crysmann, Berthold. to appear. Coordination. In K. Brown, ed., *Encyclopedia of Language and Linguistics*. Oxford: Elsevier, 2nd edn.
- Dalrymple, Mary and Ron Kaplan. 2000. Feature indeterminacy and feature resolution. *Language* 76(4):759–798.
- Daniels, Michael. 2001. On a type-based analysis of feature neutrality and the coordination of unlikes. In *Proceedings of the 8th International Conference on Head-Driven Phrase Structure Grammar*, CSLI Online Proceedings, pages 137–147. Stanford: CSLI Publications.
- Flickinger, Daniel P. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering* 6(1):15–28.
- Höhle, Tilman. 1983. Topologische Felder. ms., University of Cologne.
- Ingria, R. J. P. 1990. The limits of unification. In *Proceedings of the 28th Annual Meeting of the ACL*, pages 194–204.
- Kathol, Andreas. 1999. Agreement and the syntax-morphology interface in HPSG. In R. Levine and G. Green, eds., *Studies in Contemporary Phrase Structure Grammar*, pages 209–260. Cambridge and New York: Cambridge University Press.
- Levine, Robert, Thomas Hukari, and Michael Calcagno. 2001. Parasitic gaps in English: Some overlooked cases and their theoretical implications. In P. Culicover and P. Postal, eds., *Parasitic Gaps*, pages 181–222. Cambridge, MA: MIT Press.
- Levy, Roger and Carl Pollard. 2001. Coordination and neutralization in HPSG. In *Proceedings of the 8th International Conference on Head-Driven Phrase Structure Grammar*, CSLI Online Proceedings, pages 221–234. Stanford: CSLI Publications.
- Müller, Stefan. 1999. An HPSG analysis of free relative clauses in German. *Grammars* 2:53–105.
- Pollard, Carl and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. Stanford: CSLI and University of Chicago Press.
- Zaenen, Annie and Lauri Karttunen. 1984. Morphological non-distinctiveness and coordination. In *Proceedings of the First Eastern States Conference on Linguistics (ESCOL)*, pages 309–320.



---

# From Semantic Restrictions to Reciprocal Meanings

SIVAN SABATO AND YOAD WINTER <sup>†</sup>

## Abstract

This paper proposes a new approach to the interpretation of reciprocal expressions using the Strongest Meaning Hypothesis of Dalrymple et al. (1998). We propose a system in which reciprocal meanings are derived directly from semantic restrictions using the SMH, and characterize this derivation process. We present methods to construct a linguistic test for the availability of a reciprocal meaning, or otherwise to prove that a specific meaning is not available for reciprocals. These methods are then used to analyze two controversial reciprocal meanings.

**Keywords** STRONGEST MEANING HYPOTHESIS, RECIPROCAL EXPRESSIONS, SEMANTIC RESTRICTIONS

## 2.1 Introduction

The interpretation of reciprocal expressions (*each other*, *one another*) exhibits a remarkably wide variation, which is affected in intricate ways by the predicate in the scope of the reciprocal. For example, sentence (1) entails that each person in the group likes every other person in the group, while sentences (2) and (3) do not entail an analogous claim.

---

<sup>†</sup>Participation in the conference was partly supported by the Computer Science Department of the Technion - Israel Institute of Technology.

- (1) These three people like each other.
- (2) The three planks are stacked on top of each other.
- (3) The 3rd grade students gave each other measles.

In an attempt to explain this phenomenon, Dalrymple et al. (1998) (henceforth DKKMP) introduced the *Strongest Meaning Hypothesis* (SMH). According to this principle, the reading associated with the reciprocal in a given sentence is *the strongest available reading which is consistent with relevant information supplied by the context*. This allows sentence (2) to be felicitous even though it is impossible for each of the three planks to be stacked on top each of the other planks. A similar weakening occurs in (3), since one cannot get measles from more than one person.

DKKMP postulate an array of reciprocal meanings which the SMH has to choose from, independently of the SMH itself and the semantic properties of predicates. This paper proposes a new system for predicting the interpretation of reciprocals in a given sentence. In this system, the SMH is implemented as a mapping from semantic restrictions on the predicate's denotation into the interpretation of the reciprocal, with no independent assumptions about available reciprocal meanings. We present methods to construct a test for the availability of a reciprocal meaning, or otherwise to prove that a specific meaning is not available for reciprocals. These methods are then used to analyze two previously suggested reciprocal meanings.

## 2.2 Semantic Restrictions and Reciprocal Meanings

In this section we define the notion of *semantic restriction* and show its relevance in delimiting the range of interpretations available for a reciprocal in a given sentence. Then we define the notion of *reciprocal meaning*, imposing on it natural restrictions from generalized quantifier theory. We subsequently show that for every reciprocal interpretation there is exactly one minimal meaning that extends it, thereby proposing a method for attesting reciprocal meanings using natural language sentences. The implications of this method are studied in the next sections.

### 2.2.1 Notation

Let  $R, R' \subseteq E^2$  be binary relations over  $E$ , let  $A \subseteq E$  be a subset of  $E$ , and let  $\alpha, \beta \subseteq \wp(E^2)$  be sets of binary relations over  $E$ . We use the following notation:

- ▷ The identity relation:  $I \stackrel{def}{=} \{(x, x) \mid x \in E\}$ .

- ▷  $R$  restricted to  $A$ :  $R|_A \stackrel{def}{=} R \cap A^2$ .
- ▷  $R$  restricted to  $A$  and disregarding identities:  $R \downarrow_A \stackrel{def}{=} R|_A \setminus I$ .
- ▷  $R \subseteq_A R' \iff R \downarrow_A \subseteq R' \downarrow_A$ , and similarly for  $R =_A R'$ ,  $R \neq_A R'$ , etc.
- ▷  $\alpha \subseteq_A \beta \iff \{R \downarrow_A \mid R \in \alpha\} \subseteq \{R \downarrow_A \mid R \in \beta\}$ , and similarly for  $\alpha =_A \beta$ ,  $\alpha \neq_A \beta$ , etc.
- ▷  $\min(\alpha) \stackrel{def}{=} \{R \in \alpha : \forall R' \in \alpha [R' \subseteq R \Rightarrow R' = R]\}$
- ▷ Let  $X$  and  $Y$  be sets, and let  $D \subseteq X \times Y$  be a binary relation.  
For any  $x \in X, y \in Y$ :
  - $D(x, y)$  holds if and only if  $(x, y) \in D$ , and
  - $D(x)$  is the image of  $x$  under  $D$ :  $D(x) \stackrel{def}{=} \{y \in Y \mid D(x, y)\}$

### 2.2.2 Semantic Restrictions

We first take a closer look at the informal concept of ‘relevant information’ which is used by DKKMP in their formulation of the SMH. Clearly, not all contextual information allows weakening of the reciprocal meaning. Otherwise, according to the SMH by DKKMP, the two sentences in (4) below would not be contradictory, since the information given in the first sentence would cause the reciprocal in the second sentence to require weaker truth conditions.

- (4) # John and Bill don’t know each other. John, Bill and Dan know each other.

To eliminate such undesired consequences, we propose to only consider *semantic restrictions* of the binary predicate in the scope of the reciprocal, along the lines of Winter (2001). A semantic restriction of a binary predicate  $P$  over the domain of entities  $E$  is a set  $\Theta_P$  of binary relations over  $E$ :  $\Theta_P \subseteq \wp(E^2)$ . This is the set of relations that are possible as denotations of the predicate. For example, the denotation of the predicate *stare at* is limited to relations that are also (possibly partial) functions, since one cannot stare at more than one person at a time. Therefore  $\Theta_{\text{stare at}}$  is the set of binary relations over  $E$  which are (possibly partial) functions.

We consider reciprocal sentences of the form *NP P each other*, where  $NP$  denotes a set of entities and  $P$  denotes a binary relation  $R$  over entities. The denotation of the reciprocal expression *each other* is accordingly assumed to be a relation between sets of entities and binary relations. Obviously, the denotation of the reciprocal expression in a given sentence cannot be determined for binary relations outside the semantic restriction of  $P$ . Thus, given a semantic restriction  $\Theta$ , the *interpretation* of a reciprocal expression relative to  $\Theta$  is a binary relation

$\mathcal{I}_\Theta \subseteq \wp(E) \times \Theta$ . The *reciprocal interpretation domain* of  $\Theta$ , denoted  $\text{RECIP}_\Theta$ , is the set of all possible reciprocal interpretations relative to  $\Theta$ :  $\text{RECIP}_\Theta \stackrel{\text{def}}{=} \wp(\wp(E) \times \Theta)$ .

It is known that the SMH is most easily attested with spatial predicates such as *sit alongside* and *stand on top*. Very often the SMH does not affect kinship relations as well as some other types of relations. The following contrast demonstrates this:

- (5) The two chairs are stacked on top of each other.
- (6) #Ruth and Beth are each other's mother.

A weakening effect allows sentence (5) to be felicitous, but a similar effect does not occur in sentence (6), although world knowledge precludes both two-way stacking and two-way mothering. We conjecture that semantic restrictions are not always an exact representation of world knowledge, and are more refined for some classes of predicates than for others. The reasons for this differentiation are poorly understood and require further research.

### 2.2.3 Reciprocal Meanings

The interpretation of a reciprocal relative to a semantic restriction, as defined above, is a novel notion and central to our analysis of reciprocals in general. However, different *meanings* for reciprocals have been suggested and debated upon extensively in the literature. In contrast with a reciprocal interpretation, a reciprocal meaning is defined for *all* binary relations and not only for relations in a given semantic restriction. As a preliminary to our analysis of the meanings available for reciprocals, we propose a formal definition of the notion of reciprocal meaning. The definition captures the properties that a reciprocal meaning must have, though it does not require that the meaning manifest itself in an actual reciprocal expression.

A reciprocal meaning is a relation  $\Pi \subseteq \wp(E) \times \wp(E^2)$ . Thus, reciprocal meanings are all in the domain  $\text{RECIP}_\Theta$  with  $\Theta = \wp(E^2)$ . We assume that a reciprocal meaning must be conservative on its first argument,<sup>1</sup> as expected of any natural language determiner (Keenan and Westerstahl, 1996). Furthermore, reciprocal meanings are never sensitive to relations between identical pairs.<sup>2</sup> In addition, all reciprocal meanings suggested so far in the literature are upward monotonic in the second argument,<sup>3</sup> and we expect this to be true in general. These three properties are all subsumed by the following single property of *argument*

<sup>1</sup>Formally,  $\forall A \subseteq E, R \subseteq E^2 [\Pi(A, R) \iff \Pi(A, R \cap A^2)]$

<sup>2</sup>Formally,  $\forall A \subseteq E, R \subseteq E^2 [\Pi(A, R) \iff \Pi(A, R \setminus I)]$

<sup>3</sup>Formally,  $\forall R, R' \subseteq E^2 [(\Pi(A, R) \wedge R \subseteq R') \Rightarrow \Pi(A, R')]$

*monotonicity*:

**Definition 1** A binary relation  $D \subseteq \wp(E) \times \beta$ , where  $\beta \subseteq \wp(E^2)$ , is *argument-monotonic* if and only if the following holds:

$$\forall A \subseteq E [\forall R, R' \in \beta [(D(A, R) \wedge R \subseteq_A R') \Rightarrow D(A, R')]]$$

Argument monotonicity is therefore used as the underlying property of reciprocal meanings:

**Definition 2** A *reciprocal meaning* over a domain  $E$  is a relation  $\Pi \subseteq \wp(E) \times \wp(E^2)$  that is argument-monotonic.

For similar reasons to the ones listed above, we assume that like reciprocal meanings, reciprocal interpretations in natural language are also argument-monotonic.

#### 2.2.4 When is a Reciprocal Meaning Attested?

When presented with a potential reciprocal meaning, we would like to find out in which settings we can test whether this meaning is indeed available. In other words: what semantic restrictions of binary predicates would allow us to attest a given reciprocal meaning? Formally, we define the notion of *congruence* between a reciprocal meaning and a reciprocal interpretation  $\mathcal{I}_\Theta \in \text{RECIP}_\Theta$ , for a semantic restriction  $\Theta$ :

**Definition 3** Let  $\Theta$  be a semantic restriction over  $E$ . A reciprocal meaning  $\Pi$  over  $E$  is *congruent* with a reciprocal interpretation  $\mathcal{I}_\Theta \in \text{RECIP}_\Theta$  if  $\Pi$  is a minimal reciprocal meaning that extends  $\mathcal{I}_\Theta$ . Formally,  $\Pi$  satisfies:

1.  $\forall A \subseteq E, R \in \Theta [\mathcal{I}_\Theta(A, R) \iff \Pi(A, R)]$ , and
2. Any reciprocal meaning  $\Pi'$  that satisfies 1, also satisfies  $\Pi \subseteq \Pi'$ .

Because of the semantic restrictions on the denotation of two-place predicates in natural language, we cannot always directly extract a meaning for a reciprocal expression using the truth-conditions of reciprocal sentences. Consider for instance the following sentence:

- (7) Proposals 1 through  $n$  are similar to each other.

Given that the predicate *be similar* is symmetric, the interpretation of the reciprocal in (7) is in  $\text{RECIP}_{SYM}$  where  $SYM$  is defined by:  $SYM = \{R \subseteq E^2 \mid \forall x, y \in E [R(x, y) \Rightarrow R(y, x)]\}$ . Since (7) is true only if every proposal is similar to every other proposal, the interpretation of *each other* in (7) is the relation  $\mathcal{I}_{SYM}^0 \in \text{RECIP}_{SYM}$  defined by:  $\mathcal{I}_{SYM}^0 \stackrel{def}{=} \{(A, R) \in \wp(E) \times SYM \mid \forall x, y \in A [x \neq y \Rightarrow R(x, y)]\}$ . This interpretation can be extended by at least two reciprocal meanings

proposed in the literature: Both *Strong Reciprocity*<sup>4</sup> (SR) from Langendoen (1978) and *Strong Alternative Reciprocity*<sup>5</sup> (SAR) from DKKMP match. But SR is congruent with  $\mathcal{I}_{SYM}^0$  while SAR is not. More generally, we claim that any meaning associated with reciprocals should be congruent with the interpretation of the reciprocal in at least one natural language sentence. In this case we say that this sentence *attests* the meaning in question.

According to the following two propositions, if  $\mathcal{I}_\Theta$  is argument-monotonic, it is congruent with exactly one reciprocal meaning.

**Proposition 1** *For every semantic restriction  $\Theta$  over  $E$  and a reciprocal interpretation  $\mathcal{I}_\Theta \in \text{RECIP}_\Theta$ , there is at most one reciprocal meaning  $\Pi$  over  $E$  that is congruent with  $\mathcal{I}_\Theta$ .*

**Proposition 2** *For every semantic restriction  $\Theta$  over  $E$  and an argument-monotonic reciprocal interpretation  $\mathcal{I}_\Theta \in \text{RECIP}_\Theta$ , there exists a reciprocal meaning  $\Pi$  over  $E$  that is congruent with  $\mathcal{I}_\Theta$ .*

Here and henceforth, proofs are omitted in the body of the paper. Selected proofs can be found in the appendix.

By Propositions 1 and 2, for any semantic restriction  $\Theta$  over  $E$  and an argument-monotonic reciprocal interpretation  $\mathcal{I}_\Theta \in \text{RECIP}_\Theta$ , there is a unique reciprocal meaning that is congruent with  $\mathcal{I}_\Theta$ . On the empirical side, this result means that when given a sentence with a reciprocal expression, such as sentences (1)-(3), when  $\Theta$  is the semantic restriction of the predicate in the sentence, the important semantic decision concerns the *interpretation* of the reciprocal chosen from the domain  $\text{RECIP}_\Theta$ . The meaning of the reciprocal can be uniquely determined by this choice. In the following section we propose a new way of choosing a reciprocal interpretation according to the SMH.

### 2.3 The Interpretation of the Reciprocal

We propose that the SMH is realized as a *local maximality* principle: a reciprocal sentence is consistent with models in which no pairs in the antecedent set can be added to the denotation of the predicate within its semantic restriction. Formally:

**Definition 4** Let  $\Theta$  be a semantic restriction over  $E$ . The *SMH-based interpretation* of the reciprocal is the relation  $R_\Theta \in \text{RECIP}_\Theta$ , defined as follows:

$$\forall A \subseteq E, R \in \Theta [R_\Theta(A, R) \iff \forall R' \in \Theta [(R \subseteq_A R') \Rightarrow (R =_A R')]]$$

---

<sup>4</sup> $\forall A \subseteq E, R \subseteq E^2 [SR(A, R) \iff \forall x, y \in A [x \neq y \Rightarrow R(x, y)]]$

<sup>5</sup> $\forall A \subseteq E, R \subseteq E^2 [SAR(A, R) \iff \forall x, y \in A [x \neq y \Rightarrow (R(x, y) \vee R(y, x))]]$

This definition allows a correct prediction of the meaning of sentences presented in DKKMP and analyzed there using their system. Let us review examples (1)-(3). According to our system, the interpretation of the reciprocal in each sentence is determined by the semantic restrictions of the predicate. In sentence (1), the predicate *like* has no restrictions:  $\Theta_{\text{like}} = \wp(E^2)$ . Hence,  $R_{\Theta_{\text{like}}}(A, R) \iff R \supseteq A^2 \setminus I$ , i.e. the sentence is deemed true only if each person in the antecedent set likes each of the others. In sentence (2), we assume that the semantic restriction of the predicate *stack on top* is the set  $\Theta_{\text{stack on top}}$  that includes all the relations  $R \subseteq E^2$  such that  $R$  and  $R^{-1}$  are (possibly partial) functions, and  $R$  is acyclic. Consequently,  $R_{\Theta_{\text{stack on top}}}(A, R)$  holds if and only if the elements of  $A$  are arranged into one sequential stack, as expected. In sentence (3), the predicate *give measles* may only denote acyclic relations which are the inverse of a function: one cannot get measles twice or give measles before getting measles. Using this semantic restriction, we find that the sentence is predicted to be true if and only if each 3rd grade student is connected to each other 3rd grade student by the transitive and symmetric closure of the denotation of *give measles*. This is in fact the expected meaning of this sentence. Unlike DKKMP, this proposal also gives a correct prediction of the truth conditions for the following sentence:

(8) The pirates are staring at each other.

The system proposed by DKKMP expects this sentence to be consistent with *Intermediate Reciprocity* (Langendoen, 1978), which requires all pirates to be connected via the transitive closure of the *stare at* relation. However, as they observe, the actual truth conditions of this sentence match the weaker *One-way Weak Reciprocity*, which only requires that each pirate stares at some other pirate. In the present proposal, we derive this interpretation of the reciprocal assuming that  $\Theta_{\text{stare at}}$  is the set of (possibly partial) functions over  $E$ .

From Definition 4, it is clear that  $R_{\Theta}$  is argument-monotonic for any semantic restriction  $\Theta$ . Therefore, by Propositions 1 and 2, for each semantic restriction  $\Theta$  there is exactly one reciprocal meaning congruent with  $R_{\Theta}$ . In the following section we use the proposed framework and the definition of  $R_{\Theta}$  to examine the possibility of attesting two controversial meanings that have been suggested for reciprocals.

## 2.4 Predicting the Existence of Reciprocal Meanings

In this section we study the implications of our method for two meanings of reciprocals that were proposed in the literature. Section 2.4.1 shows two general lemmas that are useful in characterizing the semantic

restriction  $\Theta$  for which  $R_\Theta$  is congruent with a given reciprocal meaning  $\Pi$ . In sections 2.4.2 and 2.4.3 we apply these lemmas in studying congruence with the reciprocal meanings *Weak Reciprocity* (Langendoen, 1978) and *Inclusive Alternative Ordering* (Kański, 1987).

#### 2.4.1 Characterizing the Congruence Relation

In this section we present two lemmas which provide general methods for analyzing the possibility of attesting a given reciprocal meaning. Though presented here for finite domains, these lemmas are also provable for infinite domains, as long as the reciprocal meaning conforms to an additional (reasonable) requirement, which we do not elaborate upon here.

Lemma 3 below provides a characterization of the congruence relation between the interpretation  $R_\Theta$  of a given semantic restriction  $\Theta$ , and a given reciprocal meaning. This characterization may then be used to check which semantic restrictions attest a reciprocal meaning in question. If a natural language predicate with one of these semantic restrictions is found, it is then possible to devise a reciprocal sentence which attests the given meaning.

**Lemma 3** *Let  $\Theta$  be a semantic restriction over a finite domain  $E$ , and let  $\Pi$  be a reciprocal meaning over  $E$ . Then  $R_\Theta$  is congruent with  $\Pi$  if and only if  $\forall A \subseteq E [R_\Theta(A) =_A \min(\Pi(A))]$ .*

The following lemma shows that in order to check whether there is *any* semantic restriction that attests a given reciprocal meaning  $\Pi$ , it is enough to check one semantic restriction determined by  $\Pi$ , which we denote  $M_\Pi$ :  $M_\Pi \stackrel{def}{=} \bigcup_{A \subseteq E} \min(\Pi(A))$ .

**Lemma 4** *Let  $\Pi$  be a reciprocal meaning over a finite domain  $E$  that is congruent with  $R_\Theta$  for some semantic restriction  $\Theta$ . Then  $\Pi$  is congruent with  $R_{M_\Pi}$ , where  $M_\Pi$  is the semantic restriction defined above.*

#### 2.4.2 Weak Reciprocity

*Weak Reciprocity* (Langendoen, 1978) defines for any given domain  $E$  the reciprocal meaning WR specified by:

$$\forall A \subseteq E, R \subseteq E^2 [WR(A, R) \iff \forall x \in A [\exists y \in A [y \neq x \wedge R(x, y)] \wedge \exists y \in A [y \neq x \wedge R(y, x)]]]$$

In words, WR requires that each member of the set  $A$  participates in the relation both as the first and as the second argument. WR was suggested in Langendoen (1978) as a possible reciprocal meaning. However, this view is rejected on empirical grounds by DKKMP, where it is claimed that all the examples in the literature that had been claimed to demonstrate WR are in fact consistent with other known reciprocal



meanings as well. DKKMP point out that the predicates used in those examples are all symmetric. We show that according to the current system, it is in fact impossible to attest WR with *any* semantic restriction except for very small domains.

**Proposition 5** *For a domain  $E$  such that  $|E| \geq 6$ , there is no semantic restriction  $\Theta$  over  $E$  such that WR is congruent with  $R_\Theta$ .*

Since WR is defined for any given domain  $E$ , showing that the reciprocal meanings it provides for some domains are unattestable disqualifies Weak Reciprocity as a generator of reciprocal meanings.

### 2.4.3 Inclusive Alternative Ordering

DKKMP include in their system the operator *Inclusive Alternative Ordering* (IAO) (Kański, 1987), defined by:

$$\forall A \subseteq E, R \subseteq E^2 [IAO(A, R) \iff \forall x \in A [\exists y \in A [x \neq y \wedge (R(x, y) \vee R(y, x))]]]$$

IAO is proposed in DKKMP as the weakest meaning available for reciprocal expressions. It requires that each member of the antecedent set participate in the relation as either the first or the second argument. IAO thus allows a “partitioning” of the antecedent set into subsets not connected by  $R$ . The following sentence is claimed by DKKMP to exemplify IAO:

- (9) He and scores of other inmates slept on foot-wide planks stacked atop each-other.

This sentence is true if there are several disjoint stacks of planks, a configuration that is allowed by IAO but not by other reciprocal meanings in the system of DKKMP.

Using Lemma 3, we can characterize the semantic restrictions attesting IAO. Let  $\Theta_{IAO}$  be the set of binary relations  $R \subseteq E^2$  such that (1)  $R$  is anti-symmetric; and (2) there are no paths longer than 2 edges in the underlying undirected graph induced by  $R$ .

**Proposition 6** *IAO is congruent with  $R_{\Theta_{IAO}}$ .*

$\Theta_{IAO}$  is not the only semantic restriction  $\Theta$  for which IAO is congruent with  $R_\Theta$ . However, by Lemma 3, for any semantic restriction  $\Theta$  such that IAO is congruent with  $R_\Theta$ ,  $\forall A \subseteq E [R_\Theta(A) =_A R_{\Theta_{IAO}}(A)]$ . Consequently, for any semantic restriction  $\Theta$  such that IAO is congruent with  $R_\Theta$ , all relations in  $\Theta$  must satisfy the conditions given for relations in  $\Theta_{IAO}$ . In addition,  $\Theta$  must allow any element in  $E$  to stand in the relation with any given number of other elements in  $E$ . We submit that although it is theoretically possible to construct a case for

attesting IAO, a binary predicate with the sort of semantic restriction required for such a test is unlikely to be found in natural language.

We propose a different explanation to the truth condition of (9). We claim that the “partitioning” effect in (9) is external to the reciprocal and not part of its meaning. The following contrast exemplifies the effect of such “external partitioning”:

(10) The planks are stacked atop each other.

(11) Planks 1, 2, 3, and 4 are stacked atop each other.

Sentence (10) is felicitous if there are four planks arranged in two stacks of two planks each. This is in contrast with the infelicity of (11) in the same situation. Winter (2000) observes that partitioning effects occur with plural definites, but not with proper name conjunction. We follow this line and claim that partitions in reciprocal sentences are external and not inherent to the reciprocal interpretation.

## 2.5 Summary

This paper presents a novel approach to the systematic analysis of reciprocal meanings according to the Strongest Meaning Hypothesis. The system we propose derives reciprocal interpretations directly from the operation of the SMH on the semantic restrictions of the predicate. The logical restrictions affecting reciprocal meanings were spelled out, and it was shown that they uniquely determine a meaning from an interpretation of the reciprocal. Principles for the examination of meanings and the construction of appropriate linguistic tests for attesting them were defined and exemplified, and some negative and positive conclusions on the availability of previously suggested reciprocal meanings were shown to follow from these criteria.

## 2.6 Appendix: Selected Proofs

**Proposition 1** *For every semantic restriction  $\Theta$  over  $E$  and a reciprocal interpretation  $\mathcal{I}_\Theta \in \text{RECIP}_\Theta$ , there is at most one reciprocal meaning  $\Pi$  over  $E$  that is congruent with  $\mathcal{I}_\Theta$ .*

**Proof** Assume for contradiction that there are two reciprocal meanings  $\Pi_1$  and  $\Pi_2$  such that  $\Pi_1$  and  $\Pi_2$  are both congruent with  $\mathcal{I}_\Theta$ . Then the relation  $\Pi_3$ , defined by  $\Pi_3 \stackrel{def}{=} \Pi_1 \cap \Pi_2$  is also a reciprocal meaning. It extends  $\mathcal{I}_\Theta$ , and it is stronger than at least one of  $\Pi_1$  and  $\Pi_2$ . Therefore at least one of  $\Pi_1$  and  $\Pi_2$  is not congruent with  $\mathcal{I}_\Theta$ , a contradiction.  $\square$

**Proposition 2** *For every semantic restriction  $\Theta$  over  $E$  and an argument-monotonic reciprocal interpretation  $\mathcal{I}_\Theta \in \text{RECIP}_\Theta$ , there exists a reciprocal meaning  $\Pi$  over  $E$  that is congruent with  $\mathcal{I}_\Theta$ .*

**Proof** Let  $\Omega$  be the set of reciprocal meanings that extend  $\mathcal{I}_\Theta$ . First, we show that  $\Omega \neq \emptyset$ : Let  $\Pi \subseteq \wp(E) \times \wp(E^2)$  be the relation such that

$$\forall A \subseteq E, R \subseteq E^2 [\Pi(A, R) \iff \exists S \in \mathcal{I}_\Theta(A) [S \subseteq_A R]]$$

$\Pi$  is a clearly argument monotonic, and is therefore a reciprocal meaning.  $\Pi$  also extends  $\mathcal{I}_\Theta$ :  $\forall A \subseteq E, R \in \Theta [\mathcal{I}_\Theta(A, R) \iff \Pi(A, R)]$ . The left-to-right implication trivially follows from the definition of  $\Pi$ , and the right-to-left implication follows from the definition of  $\Pi$  and the argument-monotonicity of  $\mathcal{I}_\Theta$ . Hence  $\Omega \neq \emptyset$ .

Let  $\Pi_\cap \subseteq \wp(E) \times \wp(E^2)$  be the relation defined by:

$$\forall A \subseteq E, R \subseteq E^2 [\Pi_\cap(A, R) \iff \forall \Pi \in \Omega [\Pi(A, R)]]$$

$\Pi_\cap$  is argument monotonic, therefore it is a reciprocal meaning. By the definition of  $\Pi_\cap$ , there is no reciprocal meaning stronger than  $\Pi_\cap$  that extends  $\mathcal{I}_\Theta$ . Therefore  $\Pi_\cap$  is congruent with  $\mathcal{I}_\Theta$ .  $\square$

**Lemma 3** *Let  $\Theta$  be a semantic restriction over a finite domain  $E$ , and let  $\Pi$  be a reciprocal meaning over  $E$ . Then  $R_\Theta$  is congruent with  $\Pi$  if and only if  $\forall A \subseteq E [R_\Theta(A) =_A \min(\Pi(A))]$ .*

**Proof** “Only If”: Suppose  $\Pi$  is congruent with  $R_\Theta$ . We first prove that  $\forall A \subseteq E [R_\Theta(A) \supseteq_A \min(\Pi(A))]$ . Assume for the sake of contradiction that there is a set  $B \subseteq E$  such that  $R_\Theta(B) \not\subseteq_B \min(\Pi(B))$ , and let  $R_0$  be a relation such that  $R_0 \in \min(\Pi(B)) \setminus \{R \downarrow_B \mid R_\Theta(B, R)\}$ . We define the reciprocal meaning  $\Pi_1$  as follows:

$$\Pi_1 \stackrel{def}{=} \Pi \setminus \{(B, R) \mid R \downarrow_B = R_0\}$$

$\Pi_1$  is indeed argument-monotonic: Let  $R, R'$  be relations such that  $\Pi_1(B, R)$  and  $R \subseteq_B R'$  hold. We need to show that  $\Pi_1(B, R')$  holds.  $\Pi(B, R)$  holds, hence by argument-monotonicity  $\Pi(B, R')$  and  $\Pi(B, R \downarrow_B)$  hold. In addition,  $R_0 \in \min(\Pi(B))$ . Therefore  $R \downarrow_B \not\subseteq R_0$ , and consequently  $R' \downarrow_B \neq R_0$ . Hence  $\Pi_1(B, R')$  holds.

$\Pi_1$  also extends  $R_\Theta$ : By our choice of  $R_0$ , for any relation  $R$  such that  $R_\Theta(B, R)$  holds,  $R \downarrow_B \neq R_0$ . Hence, for all  $R \in \Theta$ :

$$\begin{aligned} \Pi_1(B, R) &\iff \Pi(B, R) \wedge R \downarrow_B \neq R_0 \iff \\ &R_\Theta(B, R) \wedge R \downarrow_B \neq R_0 \iff R_\Theta(B, R) \end{aligned}$$

We conclude that  $\Pi_1$  is stronger than  $\Pi$  and extends  $R_\Theta$ . Therefore  $\Pi$  is not congruent with  $R_\Theta$ , contradicting the assumption. This concludes the proof that  $\forall A \subseteq E [R_\Theta(A) \supseteq_A \min(\Pi(A))]$ .

Let us now show that also  $\forall A \subseteq E [R_\Theta(A) \subseteq_A \min(\Pi(A))]$ . Let  $A \subseteq E$  be a set and  $R$  be a relation such that  $R_\Theta(A, R)$  holds.  $\Pi$  extends  $R_\Theta$ , therefore  $\Pi(A, R)$  holds. Let  $S$  be a relation such that  $S \in \min(\Pi(A))$  and  $S \subseteq R$ .  $S$  surely exists since the domain is finite.  $R_\Theta(A) \supseteq_A \min(\Pi(A))$ , therefore  $R_\Theta(A, S)$  holds. Hence, by the definition of  $R_\Theta$ ,  $R \subseteq_A S$  holds. Therefore  $S =_A R$ , and thus indeed the inclusion holds. This concludes the proof of the ‘‘only if’’ direction.

‘‘If’’: Suppose the right-hand-side holds. We show that the two conditions for congruence with  $R_\Theta$  hold for  $\Pi$ .

1.  $\Pi$  extends  $R_\Theta$ :
  - (a)  $\forall R \in \Theta [R_\Theta(A, R) \Rightarrow \Pi(A, R)]$ : Let  $R \in \Theta$  be a relation such that  $R_\Theta(A, R)$  holds. Then by the supposition, there is a relation  $S$  such that  $S \in \min(\Pi(A))$  and  $R =_A S$ . By the argument-monotonicity of  $\Pi$ ,  $\Pi(A, R)$  holds.
  - (b)  $\forall R \in \Theta [\Pi(A, R) \Rightarrow R_\Theta(A, R)]$ : Let  $R \in \Theta$  be a relation such that  $\Pi(A, R)$  holds. Let  $S$  be a relation such that  $S \in \min(\Pi(A))$  and  $S \subseteq R$ .  $S$  surely exists since the domain is finite. By the supposition, there is a relation  $T \in \Theta$  such that  $T =_A S$  and  $R_\Theta(A, T)$  holds.  $T \subseteq_A R$ , therefore by argument-monotonicity of  $R_\Theta$  over  $\Theta$ ,  $R_\Theta(A, R)$  holds.
2. Let  $\Pi_1$  be a reciprocal meaning that extends  $R_\Theta$ . We show that  $\Pi \subseteq \Pi_1$  holds: Let  $A \subseteq E$  be a set and  $R$  be a relation such that  $\Pi(A, R)$  holds. Let  $S$  be a relation such that  $S \in \min(\Pi(A))$  and  $S \subseteq R$ .  $S$  surely exists since the domain is finite. By the supposition, there is a relation  $T \in \Theta$  such that  $S =_A T$  and  $R_\Theta(A, T)$  holds.  $\Pi_1$  extends  $R_\Theta$ , therefore  $\Pi_1(A, T)$  holds. By argument-monotonicity of  $\Pi_1$ ,  $\Pi_1(A, R)$  holds, hence  $\Pi \subseteq \Pi_1$ .  $\square$

**Lemma 4** *Let  $\Pi$  be a reciprocal meaning over a finite domain  $E$  that is congruent with  $R_\Theta$  for some semantic restriction  $\Theta$ . Then  $\Pi$  is congruent with  $R_{M_\Pi}$ , where  $M_\Pi$  is the semantic restriction defined by  $M_\Pi \stackrel{def}{=} \bigcup_{A \subseteq E} \min(\Pi(A))$ .*

**Proof** Assume for contradiction that  $\Pi$  is not congruent with  $R_{M_\Pi}$ . By Lemma 3, there is a set  $A \subseteq E$  such that  $R_{M_\Pi}(A) \neq_A \min(\Pi(A))$ . We use the same lemma to contradict the congruence of  $\Pi$  with  $R_\Theta$ . Consider the following two cases:

1. If there is a relation  $S \in \min(\Pi(A))$  such that  $R_{M_\Pi}(A, S)$  does not hold, then by the definition of  $M_\Pi$ ,  $S \in M_\Pi$ . Hence by the definition of  $R_{M_\Pi}$ , there is a relation  $R \in M_\Pi$  such that  $S \subsetneq_A R$ . Let  $B \subseteq E$  be a set such that  $R \in \min(\Pi(B))$ . Since  $\Pi$  is congruent with  $R_\Theta$ , by Lemma 3  $R_\Theta(B) =_B \min(\Pi(B))$ . Therefore there is a relation  $R' \in \Theta$  such that  $R_\Theta(B, R')$  holds and  $R' =_B R$ . Since  $R \in \min(\Pi(B))$ ,  $R \downarrow_B = R$ . Therefore  $R \subseteq R'$ . It follows that  $S \subsetneq_A R'$ . Consequently,  $\forall S' [S' =_A S \Rightarrow \neg R_\Theta(A, S')]$ .  $S \in \min(\Pi(A))$ , therefore  $R_\Theta(A) \neq_A \min(\Pi(A))$ .
2. Otherwise, there is a relation  $R \in M_\Pi$  such that  $R_{M_\Pi}(A, R)$  holds and  $\forall S \in \min(\Pi(A)) [R \neq_A S]$ . Let  $S$  be a relation such that  $S \in \min(\Pi(A))$ . Let  $B \subseteq E$  be a set such that  $R \in \min(\Pi(B))$ .  $\Pi$  is congruent with  $R_\Theta$ , therefore  $R_\Theta(B) =_B \min(\Pi(B))$ . Hence there is a relation  $R' \in \Theta$  such that  $R' =_B R$  and  $R_\Theta(B, R')$  holds. As above,  $R \subseteq R'$ . Since  $R_{M_\Pi}(A, R)$  holds,  $R \not\subseteq_A S$  and thus  $R' \not\subseteq_A S$ . Since the domain is finite, there exists a relation  $T \in \Theta$  such that  $R' \subseteq_A T$  and  $R_\Theta(A, T)$ . In addition,  $\forall S \in \min(\Pi(A)) [T \neq_A S]$ . Hence  $R_\Theta(A) \neq_A \min(\Pi(A))$ .

In both cases, the conditions of Lemma 3 do not hold for  $\Theta$ , and therefore  $\Pi$  is not congruent with  $R_\Theta$ , contradicting the assumption.  $\square$

**Proposition 5** *For a domain  $E$  such that  $|E| \geq 6$ , there is no semantic restriction  $\Theta$  over  $E$  such that  $WR$  is congruent with  $R_\Theta$ .*

**Proof** According to Lemma 4, it suffices to show that  $WR$  is not congruent with  $R_{M_{WR}}$ . We show a set  $A \subseteq E$  and a relation  $R \in M_{WR}$  such that  $R_{M_{WR}}(A, R)$  holds but  $WR(A, R)$  does not hold. It follows that  $WR$  does not extend  $M_{WR}$ , hence it is not congruent with  $R_{M_{WR}}$ .

Let  $B \subseteq E$  be a set such that  $|B| = 6$ . We denote the elements of  $B$  by  $\{a, b, c, d, e, f\}$ . Let  $R$  be the following relation (See figure 1):

$$R \stackrel{def}{=} \{(a, b), (b, a), (c, a), (d, b), (e, c), (e, d), (e, f), (f, e)\}$$

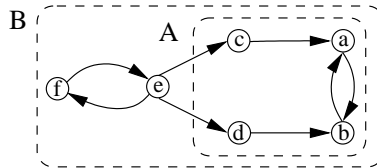


FIGURE 1: The relation  $R$

It is easily verified that  $R \in \min(WR(B))$ . Hence  $R \in M_{WR}$ . Let  $A$  be the set  $A \stackrel{def}{=} \{a, b, c, d\}$ .  $WR(A, R)$  does not hold. We show that  $R_{M_{WR}}(A, R)$  holds.

Assume for contradiction that  $R_{M_{WR}}(A, R)$  does not hold. Then there is some relation  $R_1 \in M_{WR}$  such that  $R \subsetneq_A R_1$ . Let  $(z, w) \in A^2 \setminus I$  be a pair in  $(R_1 \setminus R) \downarrow_A$ . By the definition of  $R$ , there is an element  $t \in A$  such that  $(z, t) \in R$ .  $(z, w) \notin R$ , therefore  $w \neq t$ . By definition of  $M_{WR}$ , there is a set  $C \subseteq E$  such that  $R_1 \in \min(WR(C))$ . Let us define the relation  $R_2 \stackrel{def}{=} R_1 \setminus \{(z, t)\}$ . We show that  $WR(C, R_2)$  holds, contradicting  $R_1 \in \min(WR(C))$ .  $WR(C, R_1)$  holds, therefore:

$$\begin{aligned} & \forall x \in C \setminus \{z\} [\exists y \in C [y \neq x \wedge (x, y) \in R_2]] \wedge \\ & \forall x \in C \setminus \{t\} [\exists y \in A [y \neq x \wedge (y, x) \in R_2]] \end{aligned}$$

We only have left to prove that:

1.  $[\exists y \in C [y \neq z \wedge (z, y) \in R_2]]$  and
2.  $[\exists y \in A [y \neq t \wedge (y, t) \in R_2]]$

$(z, w) \in R_1$ , therefore  $(z, w) \in R_2$ , hence formula 1 holds.  $(z, t) \in R$ , therefore  $t = a$  or  $t = b$ . In both cases there is another pair  $(v, t)$  left in  $R$ . Therefore formula 2 holds. We conclude that  $WR(C, R_2)$  holds, a contradiction. Thus the proof is complete.  $\square$

## References

- Dalrymple, M., M. Kanazawa, Y. Kim, S.A. Mchombo, and S. Peters. 1998. Reciprocal expressions and the concept of reciprocity. *Linguistics and Philosophy* 21(2):159–210.
- Kański, Z. 1987. Logical symmetry and natural language reciprocals. In *Proceedings of the 1987 Debrecen Symposium on Language and Logic*, pages 49–68.
- Keenan, E. and D. Westerståhl. 1996. Generalized quantifiers in linguistics and logic. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*. Amsterdam: Elsevier.
- Langendoen, D. Terence. 1978. The logic of reciprocity. *Linguistic Inquiry* 9:177–197.
- Winter, Y. 2000. Distributivity and dependency. *Natural Language Semantics* 8:27–69.
- Winter, Y. 2001. Plural predication and the Strongest Meaning Hypothesis. *Journal of Semantics* 18:333–365.

---

# Events from temporal logic to regular languages with branching

TIM FERNANDO

## Abstract

Events in natural language semantics, conceived as strings of observations, are extracted from formulas of linear temporal logic, and collected in regular languages. Infinite strings of sets of atomic formulas (fully specifying truth) are truncated and partialized, in line with the bounded temporal extent and descriptive content of events. Branching from that line, counterfactual events are analyzed as b(ranching)-strings accepted by finite b-automata. These structures are compared and contrasted to those of Computational Tree Logic.

**Keywords** EVENTS, TEMPORAL LOGIC, REGULAR LANGUAGES, BRANCHING

## 3.1 Introduction

Priorean temporal logics have attracted considerable attention in efforts to verify computational systems (e.g. Emerson (1992), Clarke et al. (1999)). Has that attention been matched by the formal natural language semantics community? Perhaps in the past (e.g. Thomason (1984)). But if in recent years that popularity has waned, some of the blame must be put down to a reluctance to mix events with temporal logic.<sup>1</sup> Freely appealing to worlds and times, the very influential book Dowty (1979) refers to events only informally. Subsequent works such as Parsons (1990), Kamp and Reyle (1993) and Asher and Las-

---

<sup>1</sup>That said, it is clear from papers such as Blackburn et al. (1996), Condoravdi (2002) and Bennett and Galton (2004) that not everyone shies away from such a mix.

carides (2003), however, attest to natural language semantics’ enduring interest in events as full-fledged theoretical entities. It is against this background that the present work explores the possibility of extracting events from temporal logic formulas, the truth of which they witness. We start in section 2 with the simple case of propositional linear temporal logic given by discrete future operators — LTL, for short. The infinite timelines suitable for analyzing non-terminating reactive systems (in Emerson (1992) and Clarke et al. (1999)) are noted to be at odds with the bounded temporal extent events have (according say, to Reichenbach or Vendler).<sup>2</sup> Similarly, full specification of atomic formulas true at a moment is incompatible with the partial descriptive content of events relative to any fixed moment. Accordingly, the infinite strings of sets of atomic formulas determined by timeline-valuation pairs are truncated and filled with non-atomic formulas, allowing for lazy evaluation. Events are conceptualized as strings of sets of formulas, and an event-type formulated (following Fernando (2004a)) as a set of such strings that may be accepted by a finite automaton — that is, a regular language.

The conception of an event-type as a regular language is extended in section 3 to branching time, motivated to no small degree by sentences such as

(12) Pat stopped the car before it hit the tree.

A natural reading of (12) diverges from the *before* operator in Emerson (1992) (§3.2.2), carrying (as noted in Heinämäki (1972) and Beaver and Condoravdi (2003)) the implication

(13) The car did not hit the tree, but it may well have.

We must (amongst other things) be careful to interpret the words “may well have” in (13) over possibilities ruled out by the first clause in (13). With this in mind, we augment a finite automaton with a binary relation on states to form a finite b-automaton, providing an alternative to the existential operator E in CTL\* (e.g. Emerson (1992), §4.2). Whereas finite automata accept strings, finite b-automata accept b(ranching)-strings, constituting regular b-languages.

### 3.2 Portraying LTL formulas by regular languages

Following for the most part the notation of Emerson (1992) and Clarke et al. (1999), we build the formulas of LTL from a set  $P$  of *atomic propositions* and interpret these relative to a set  $S$  of *states*, a *timeline*

---

<sup>2</sup>Henceforth, events in this paper are to be understood from the perspective of natural language semantics, as opposed to programming language theories.



$x : \mathbb{N} \rightarrow S$  giving an infinite sequence  $x(0), x(1), \dots$  of states, and a *valuation*  $\mathfrak{l} : S \rightarrow 2^P$  specifying the set  $\mathfrak{l}(s) \subseteq P$  of atomic propositions true at  $s \in S$ . We lift the notion  $\mathfrak{l}$  of truth at states to timelines  $x$

$$x \models_{\mathfrak{l}} p \quad \text{iff} \quad p \in \mathfrak{l}(x(0)) \quad \text{for } p \in P$$

and formulas  $\varphi, \psi$  generated from conjunction  $\wedge$ , disjunction  $\vee$

$$\begin{aligned} x \models_{\mathfrak{l}} \varphi \wedge \psi & \quad \text{iff} \quad x \models_{\mathfrak{l}} \varphi \text{ and } x \models_{\mathfrak{l}} \psi \\ x \models_{\mathfrak{l}} \varphi \vee \psi & \quad \text{iff} \quad x \models_{\mathfrak{l}} \varphi \text{ or } x \models_{\mathfrak{l}} \psi \end{aligned}$$

and three temporal operators that are interpreted by defining for every timeline  $x$  and  $n \in \mathbb{N}$  the timeline  $x^n : \mathbb{N} \rightarrow S$  mapping  $m \in \mathbb{N}$  to  $x(n+m)$ . We have *next*  $\mathbf{X}$ , *until*  $\mathbf{U}$  and *release*  $\mathbf{R}$

$$\begin{aligned} x \models_{\mathfrak{l}} \mathbf{X}\varphi & \quad \text{iff} \quad x^1 \models_{\mathfrak{l}} \varphi \\ x \models_{\mathfrak{l}} \varphi \mathbf{U} \psi & \quad \text{iff} \quad (\exists n \geq 0) (x^n \models_{\mathfrak{l}} \psi \text{ and } (\forall m < n) x^m \models_{\mathfrak{l}} \varphi) \\ x \models_{\mathfrak{l}} \varphi \mathbf{R} \psi & \quad \text{iff} \quad (\forall n \geq 0) (x^n \models_{\mathfrak{l}} \psi \text{ or } (\exists m < n) x^m \models_{\mathfrak{l}} \varphi) . \end{aligned}$$

Negation is defined through a map  $\bar{\cdot} : P \rightarrow P$  on atomic propositions with  $\bar{\bar{p}} = p$  (doubling  $P$ , if necessary, to  $P \times \{+, -\}$  with  $(p, +) = (p, -)$  and  $(p, -) = (p, +)$ ), and the dual (De Morgan) pairs  $(\wedge, \vee)$ ,  $(\mathbf{U}, \mathbf{R})$ ,  $(\mathbf{X}, \mathbf{X})$ , where

$$\overline{\varphi \wedge \psi} = \bar{\varphi} \vee \bar{\psi} \quad \overline{\varphi \vee \psi} = \bar{\varphi} \wedge \bar{\psi}$$

etc. Henceforth, we require of a valuation  $\mathfrak{l}$  that for all  $s \in S$  and  $p \in P$ ,

$$p \in \mathfrak{l}(s) \quad \text{iff} \quad \bar{p} \notin \mathfrak{l}(s).$$

We write  $\Phi$  for the set of LTL formulas, with a designated tautology  $\top$  and contradiction  $\perp = \bar{\top}$ , setting  $x \models_{\mathfrak{l}} \top/\perp$  for all/no  $x, \mathfrak{l}$ .

Next, we give a few instances of a language  $L$  over the alphabet  $2^\Phi$  portraying a formula  $\varphi \in \Phi$ , which we presently systematize:

$$\begin{aligned} \boxed{p, q} + \boxed{r} & \quad \text{portrays} \quad (p \wedge q) \vee r \\ \boxed{p} \boxed{q, r} & \quad \text{portrays} \quad p \wedge \mathbf{X}(q \wedge r) \\ \boxed{p}^* \boxed{q} & \quad \text{portrays} \quad p \mathbf{U} q . \end{aligned}$$

We adopt the notation of regular languages (with non-deterministic choice  $+$ , Kleene star  $^*$ , etc) and enclose a set of formulas by a box rather than by curly braces  $\{\cdot\}$  when that set is understood as a symbol. We extend  $\models_{\mathfrak{l}}$  to strings  $s = \alpha_1 \cdots \alpha_n \in (2^\Phi)^*$  conjunctively

$$x \models_{\mathfrak{l}} \alpha_1 \cdots \alpha_n \quad \text{iff} \quad \text{whenever } 1 \leq i \leq n \text{ and } \psi \in \alpha_i, \quad x^{i-1} \models_{\mathfrak{l}} \psi .$$

A language  $L \subseteq (2^\Phi)^*$  is then defined to *portray* a formula  $\varphi \in \Phi$  if  $L$  provides witnesses for the truth of  $\varphi$  in the sense that for all timelines

$x$  and valuations  $\mathfrak{l}$ ,

$$x \models_{\mathfrak{l}} \varphi \quad \text{iff} \quad (\exists \mathfrak{s} \in L) x \models_{\mathfrak{l}} \mathfrak{s}.$$

Notice that by allowing the symbols in  $L$  to be subsets of  $\Phi$  and not just of  $P$ , we ensure that for every  $\varphi \in \Phi$ , there is a regular language portraying  $\varphi$ . Take  $\boxed{\varphi}$ . But for (say)  $\varphi = p\mathbf{U}q$ ,  $\boxed{p}^*\boxed{q}$  is far better than  $\boxed{p\mathbf{U}q}$  at bringing out how  $p\mathbf{U}q$  may stretch over more than one moment. And if we are to view languages as event-types (i.e. sets of events), then it is noteworthy that  $\boxed{p\mathbf{U}q}$  has only one event/string (and a rather poorly drawn out one at that) whereas  $\boxed{p}^*\boxed{q}$  has infinitely many. Each of the strings  $\boxed{p}^n\boxed{q}$  is a more plausible sequence of snapshots than  $\boxed{p\mathbf{U}q}$  given that the truth of  $p$  and  $q$  depends on states (in isolation), unlike the compound  $p\mathbf{U}q$ .

To pick out portrayals  $L$  of  $\varphi$  of the sort given by  $\boxed{p}^*\boxed{q}$ , let us restrict the alphabet of  $L$  a bit, defining the *basis of*  $\varphi$  to be the set  $B(\varphi)$  of formulas

$$\begin{aligned} B(p) &= \{p\} & B(\varphi \bullet \psi) &= B(\varphi) \cup B(\psi) \text{ for } \bullet \in \{\wedge, \vee, \mathbf{U}\} \\ B(\mathbf{X}\varphi) &= B(\varphi) & B(\varphi\mathbf{R}\psi) &= B(\varphi) \cup B(\psi) \cup \{\perp\mathbf{R}\psi\} \end{aligned}$$

and  $B(\varphi) = \emptyset$  for  $\varphi \in \{\top, \perp\}$ . We set  $B_0(\varphi) = B(\varphi) \cap P$ , and note that for  $\varphi$  containing *no* occurrences of  $\mathbf{R}$ ,  $B(\varphi) = B_0(\varphi)$ . Membership of  $\perp\mathbf{R}\psi$  in  $B(\varphi\mathbf{R}\psi)$  hints at a certain irreducibility of  $\varphi\mathbf{R}\psi$  over finite strings. We have

**Theorem 1.** *For all  $\varphi \in \Phi$  and  $n > 0$ , there is a language  $L \subseteq (2^{B_0(\varphi)})^n (2^{B(\varphi)})^*$  that portrays  $\varphi$ .*

Let us define for every  $\varphi \in \Phi$  a language  $\mathcal{L}(\varphi)$  meeting the specification of Theorem 1. The cases of  $p \in P, \vee, \mathbf{X}, \top$  and  $\perp$  are easy:

$$\begin{aligned} \mathcal{L}(p) &= \boxed{p}\square^{n-1}\square^* & \mathcal{L}(\mathbf{X}\varphi) &= \square\mathcal{L}(\varphi) \\ \mathcal{L}(\varphi \vee \psi) &= \mathcal{L}(\varphi) + \mathcal{L}(\psi) & \mathcal{L}(\top) &= \square^n\square^* \end{aligned}$$

and  $\mathcal{L}(\perp) = \emptyset$ . For conjunction, take

$$\mathcal{L}(\varphi \wedge \psi) = \mathcal{L}(\varphi) \& \mathcal{L}(\psi)$$

where the *superposition*  $L\&L'$  of languages  $L, L'$  over the alphabet  $2^\Phi$  is the componentwise union of strings in  $L$  and  $L'$  of the same length

$$L\&L' = \bigcup_{n \geq 0} \{(\alpha_1 \cup \alpha'_1) \cdots (\alpha_n \cup \alpha'_n) \mid \alpha_1 \cdots \alpha_n \in L \text{ and } \alpha'_1 \cdots \alpha'_n \in L'\}$$

(Fernando, 2004a). As for  $\mathbf{U}$ , we cannot generalize the portrayal of  $p\mathbf{U}q$  by  $\boxed{p}^*\boxed{q}$  by equating  $\mathcal{L}(\varphi\mathbf{U}\psi)$  with  $\mathcal{L}(\varphi)^*\mathcal{L}(\psi)$ . (Take  $\varphi = \mathbf{X}p$

and  $\psi = p$ .) Instead, let us define (simultaneously with  $\mathcal{L}$ ) languages  $\mathcal{L}_0(\varphi), \mathcal{L}_1(\varphi), \dots$  that portray  $\varphi^0 = \top, \varphi^1 = \varphi \wedge X\varphi^0, \dots$

$$\begin{aligned}\mathcal{L}_0(\varphi) &= \Box^* \\ \mathcal{L}_{k+1}(\varphi) &= \mathcal{L}(\varphi) \ \& \ \Box \mathcal{L}_k(\varphi) \\ &= \mathcal{L}(\varphi^{k+1}) \quad \text{where } \varphi^{k+1} = \varphi \wedge X\varphi^k\end{aligned}$$

and set

$$\mathcal{L}(\varphi \mathbf{U} \psi) = \sum_{k \geq 0} (\mathcal{L}_k(\varphi) \ \& \ \Box^k \mathcal{L}(\psi))$$

where  $\sum$  is  $\cup$ , just as  $+$  is  $\cup$ . For  $\mathbf{R}$ , the idea is that for a fixed  $n \geq 0$ ,

$$\boxed{q}^* \boxed{p, q} + \boxed{q}^n \boxed{\perp \mathbf{R} q} \quad \text{portrays} \quad p \mathbf{R} q$$

which generalizes to

$$\mathcal{L}(\varphi \mathbf{R} \psi) = \sum_{k \geq 0} (\mathcal{L}_k(\psi) \ \& \ \Box^k (\mathcal{L}(\varphi) \ \& \ \mathcal{L}(\psi))) + (\mathcal{L}_n(\psi) \ \& \ \Box^n \boxed{\perp \mathbf{R} \psi}).$$

This completes the definition of  $\mathcal{L}$ .

Are all the languages  $\mathcal{L}(\varphi)$  regular? No (as the sums  $\sum$  for  $\mathbf{U}$  and  $\mathbf{R}$  are infinite). For  $p, q, r \in P$ , let

$$\hat{\varphi} = (p \wedge (\top \mathbf{U} q)) \ \mathbf{U} \ r$$

and note (after a moment's reflection) that every string in  $\mathcal{L}(\hat{\varphi})$  which happens to be in  $\boxed{p}^+ \boxed{r \ q}^+$  must have no more  $q$ 's than  $p$ 's (as a  $q$  in a string from  $\mathcal{L}(\hat{\varphi})$  pairs up with a  $p$ ). That is,

$$\mathcal{L}(\hat{\varphi}) \cap \boxed{p}^+ \boxed{r \ q}^+ = \sum_{i \geq 1} \boxed{p}^i \boxed{r} \sum_{1 \leq j \leq i} \boxed{q}^j$$

which is non-regular (by a pumping argument). Thus,  $\mathcal{L}(\hat{\varphi})$  cannot be regular.

To form only regular languages, we must do something about  $\mathcal{L}(\varphi \mathbf{U} \psi)$ . Let us modify  $\mathcal{L}$  to  $\hat{\mathcal{L}}$ , retaining the portrayals of formulas *without*  $\mathbf{U}$  or  $\mathbf{R}$

$$\begin{aligned}\hat{\mathcal{L}}(\top) &= \Box^n \Box^* & \hat{\mathcal{L}}(\varphi \vee \psi) &= \hat{\mathcal{L}}(\varphi) + \hat{\mathcal{L}}(\psi) \\ \hat{\mathcal{L}}(\perp) &= \emptyset & \hat{\mathcal{L}}(X\varphi) &= \Box \hat{\mathcal{L}}(\varphi) \\ \hat{\mathcal{L}}(p) &= \boxed{p} \Box^{n-1} \Box^* & \hat{\mathcal{L}}(\varphi \wedge \psi) &= \hat{\mathcal{L}}(\varphi) \ \& \ \hat{\mathcal{L}}(\psi)\end{aligned}$$

( $L \ \& \ L'$  is regular if  $L$  and  $L'$  are (Fernando, 2004a)). The key to  $\mathbf{U}$  is lazy evaluation; instead of unwinding  $p \mathbf{U} q$  fully to  $\boxed{p}^* \boxed{q}$ , we make do with

$$\boxed{q} + \boxed{p \ q} + \boxed{p \ p \ q} + \dots + \boxed{p}^{n-1} \boxed{q} + \boxed{p}^n \boxed{p \mathbf{U} q} \quad \text{portrays} \quad p \mathbf{U} q$$

from which we get

$$\hat{\mathcal{L}}(\varphi \mathbf{U} \psi) = \sum_{k < n} (\hat{\mathcal{L}}_k(\varphi) \& \square^k \hat{\mathcal{L}}(\psi)) + (\hat{\mathcal{L}}_n(\varphi) \& \square^n \boxed{\varphi \mathbf{U} \psi})$$

where just as with  $\mathcal{L}_k$ ,  $\hat{\mathcal{L}}_0(\varphi) = \square^*$  and  $\hat{\mathcal{L}}_{k+1}(\varphi) = \hat{\mathcal{L}}(\varphi) \& \square \hat{\mathcal{L}}_k(\varphi)$ . Similarly,

$$\sum_{k < n} \boxed{q}^k \boxed{p, q} + \boxed{q}^n \boxed{p \mathbf{R} q} \quad \text{portrays} \quad p \mathbf{R} q$$

and so

$$\hat{\mathcal{L}}(\varphi \mathbf{R} \psi) = \sum_{k < n} (\hat{\mathcal{L}}_k(\psi) \& \square^k (\hat{\mathcal{L}}(\varphi) \& \hat{\mathcal{L}}(\psi))) + (\hat{\mathcal{L}}_n(\psi) \& \square^n \boxed{\varphi \mathbf{R} \psi}).$$

So much for the definition of  $\hat{\mathcal{L}}$ . To formulate an analogue to Theorem 1, let us replace  $B(\varphi)$  by  $C(\varphi)$ , where

$$\begin{aligned} C(p) &= \{p\} & C(\varphi \bullet \psi) &= C(\varphi) \cup C(\psi) \text{ for } \bullet \in \{\wedge, \vee\} \\ C(\mathbf{X}\varphi) &= C(\varphi) & C(\varphi) &= \emptyset \text{ for } \varphi \in \{\top, \perp\} \end{aligned}$$

and  $C(\varphi \bullet \psi) = C(\varphi) \cup C(\psi) \cup \{\varphi \bullet \psi\}$  for  $\bullet \in \{\mathbf{U}, \mathbf{R}\}$ . Let  $C_0(\varphi) = C(\varphi) \cap P$ .

**Theorem 2.** *For all  $\varphi \in \Phi$  and  $n > 0$ , there is a language  $L \subseteq (2^{C_0(\varphi)})^n (2^{C(\varphi)})^*$  that portrays  $\varphi$  and is regular.*

Neither the language  $\hat{\mathcal{L}}(\varphi)$  behind Theorem 2 nor the language  $\mathcal{L}(\varphi)$  behind Theorem 1 necessarily offers the most obvious portrayal of  $\varphi$ . For example,  $\square$  portrays every tautology (whether or not that tautology contains  $\mathbf{R}$  or  $\mathbf{U}$ ),  $\emptyset$  portrays every contradiction, and the regular language

$$\hat{L} = \boxed{r} + \boxed{p}^* (\boxed{p, q} \boxed{r} + \boxed{p} \boxed{r, q} + \boxed{p} \boxed{r} \square^* \boxed{q})$$

portrays the formula  $\hat{\varphi} = (p \wedge (\top \mathbf{U} q)) \mathbf{U} r$ , which (as we noted above)  $\mathcal{L}$  maps to a non-regular language.<sup>3</sup> Each of  $\hat{L}$ ,  $\mathcal{L}(\hat{\varphi})$  and  $\hat{\mathcal{L}}(\hat{\varphi})$  have very different temporal extents, and temporal extent is crucial in event semantics. In particular, we had better not confuse  $\boxed{\varphi}^+$  with  $\boxed{\varphi}$  as event-types, even though both portray  $\varphi$ . For the record, let us state an obvious but important fact about portrayal, agreeing (as usual) that  $\varphi, \psi \in \Phi$  are *logically equivalent* if the same timeline-valuation pairs satisfy them.

<sup>3</sup>Observe that  $\hat{L}\square^* = \mathcal{L}(r \vee (p \mathbf{U} (p \wedge \hat{\psi})))$  for  $\hat{\psi} = (q \wedge \mathbf{X}r) \vee \mathbf{X}(r \wedge q) \vee \mathbf{X}(r \wedge \mathbf{X}(\top \mathbf{U} q))$ . Excess  $\square$ 's at the end of a string in a language  $L$  can be stripped off by intersecting  $L$  with  $\square^+ (2^\Phi)^* (2^\Phi - \{\square\})$ .

**Proposition 3.** *If  $\varphi, \psi \in \Phi$  are logically equivalent, then they are portrayed by the same languages. Conversely, if there is a language that portrays both  $\varphi$  and  $\psi$ , then  $\varphi$  and  $\psi$  are logically equivalent.*

Clearly, event-types as languages are much finer grained notions than formulas in  $\Phi$  under logical equivalence.<sup>4</sup> But if this is all we want to say, then surely Theorems 1 and 2 are overkill, are they not?

Part of the interest in Theorems 1 and 2 lies in the incremental construction of a timeline-valuation pair  $x, \mathfrak{l}$  satisfying a formula  $\varphi$ , which can (for the purpose of  $\models$ ) be reduced to the infinite sequence

$$\mathfrak{l}(x(0)), \mathfrak{l}(x(1)), \mathfrak{l}(x(2)), \dots$$

Readers familiar with tableaux for LTL (e.g. Clarke et al. (1999)) will have no doubt noticed that progressively larger initial segments of such sequences are given by strings in  $\mathcal{L}(\varphi)$  and  $\hat{\mathcal{L}}(\varphi)$  that grow as  $n$  approaches  $\infty$ .<sup>5</sup> (To keep the notation simple, we have decorated neither  $\mathcal{L}$  nor  $\hat{\mathcal{L}}$  with  $n$ ; we pay a price for that now.) The only difference between  $\mathcal{L}(\varphi)$  and  $\hat{\mathcal{L}}(\varphi)$  is the care we exercised in the latter to produce increments that some finite automata can accept. Furthermore, we have refrained from mentioning automata on infinite strings (again, see, for instance, Clarke et al. (1999)) because we want a string that is interpretable as an event, whose time E may precede a *speech time* S (for the past tense, following Reichenbach). If we are to put S and E on the same timeline (with length  $\mathbb{N}$ ), then the string with time E had better be finite so that S can come after E.

In view of the irreducibility of the release operator R over finite strings (accounting for the failure in Theorem 1 of the inclusion  $B(\varphi) \subseteq P$ ), the question arises: does event semantics have any use for R? But why should it *not*? Although an event may be bounded, its effects need not. Consider (12), repeated below.

(12) Pat stopped the car before it hit the tree.

An effect of Pat stopping the car is that car be stationary. We can assume the car remains stationary unless some force puts it in motion, which is, in turn, a precondition for the car hitting the tree. For this reason, we may conclude from (12) that in the absence of any intervening forces, the car did not hit the tree. To formalize such reasoning, let us suppose that for certain formulas  $\varphi$ , we could build a formula  $F\varphi$  saying intuitively that a force is applied on  $\varphi$ . Then the formula

<sup>4</sup>Borrowing terminology from Schubert (2000), event-types collect *characterized* situations, whereas logical equivalence is given by a wider *supports* relation.

<sup>5</sup>The entailments at stake here (definable from  $\&$ ) are related in Fernando and Nairn (2005) to restrictions and replacements in Beesley and Karttunen (2003).

(F $\varphi$ )R $\varphi$  says:

(†)  $\varphi$  holds and will continue to do so until a force is applied on it.

The notion of inertia implicit in (†) is developed in Fernando (2004b) to analyze (12) and develop ideas about aspect from Reichenbach and Vendler. But instead of explicitly mentioning R, rules of inertial flow are set out, with F $\varphi$  read as “freeze  $\varphi$ .”

### 3.3 Branching beyond CTL and regular languages

A natural reading of (12) supports (13), repeated below.

(13) The car did not hit the tree, but it may well have.

A first attempt at interpreting the multiple possibilities in (13) is to relativize satisfaction  $\models$  to a set  $X$  of timelines alongside  $x, l$ , from which to reset  $x$  by an existential operator  $\hat{E}$

$$X, x \models_l \hat{E}\varphi \quad \text{iff} \quad (\exists x' \in X) X, x' \models_l \varphi .$$

Note that whether or not  $X, x \models_l \hat{E}\varphi$  holds is independent of  $x$ .  $\hat{E}$  can be read as *epistemic might* if  $X$  is understood as the conversational common ground, subject to update (e.g. Veltman (1996)). This construal brings the portrayal of  $\varphi \vee \psi$  by  $\boxed{\varphi} + \boxed{\psi}$  in line with an interpretation of disjunction as a list of epistemic possibilities (Zimmermann, 2000). Restricting quantification to timelines  $x'$  with the same initial state  $x'(0)$  leads to the existential operator  $E$

$$X, x \models_l E\varphi \quad \text{iff} \quad (\exists x' \in X) x'(0) = x(0) \text{ and } X, x' \models_l \varphi$$

and to *state formulas* in CTL\* (Emerson (1992), §4.2) with  $x$  truncated to  $x(0)$ .  $E$  approximates what Condoravdi (2002) calls *metaphysical might* insofar as the equation  $x(0) = x'(0)$  captures alternatives under *historical necessity* (e.g. Thomason (1984)) on a set  $X$  consisting, for some binary (“successor”) relation  $R$  on the set  $S$  of states, of timelines  $x$  such that for all  $i \geq 0$ ,  $x(i) R x(i+1)$ .

Can we use  $\hat{E}$  or  $E$  to analyze the *may* in (13)? Treating it epistemically as  $\hat{E}$  will not do, if (as is eminently plausible) the first clause of (13) eliminates timelines where the car hits the tree from the common ground. As for  $E$ , it fails to provide a notion of counterfactual event or realis distance to mark ‘car did not hit tree’ as factual, and ‘car hit tree’ as an unrealized possibility (according to (13)). That such a notion is missing from CTL\* is what is called the “disconnection from the present” (Nelken and Francez, 1996) of system specifications.

But already, CTL\* presents a problem for our identification of event-types with languages, rather than the automata accepting them. Evaluating  $E\varphi$  according to  $\models$  requires more structure than is provided by

a set of strings.<sup>6</sup> Accordingly, we add branches to strings, generating the set  $\Sigma^b$  of (non-empty) *b-strings*  $s$  over an alphabet  $\Sigma$

$$s ::= \alpha \mid ss' \mid b(s, s')$$

from symbols  $\alpha \in \Sigma$  by binary operations of concatenation and branching  $b$ .<sup>7</sup> We will work with alphabets  $\Sigma \subseteq 2^\Phi$  consisting of sets of formulas, and will shortly formalize the intuition that  $b(s, s')$  says  $s'$  may follow  $s$ , whereas  $ss'$  says  $s'$  follows  $s$  (without qualification). Hence, if  $s_1$  clashes with  $s_2$ , then  $b(s, s_1)s_2$  describes  $s_1$  as a counterfactual continuation of  $s$ , which continues instead along  $s_2$ . For example,

$$b(\boxed{\text{moving-car}}, \boxed{\text{moving-car, contact}} \boxed{\text{contact}}) \boxed{\text{moving-car}} \boxed{S}$$

is a b-string depicting the sentence *The car stopped before it hit the tree* (with speech time  $S$ ), assuming (for simplicity)

$$\boxed{\text{moving-car}} \boxed{\text{moving-car}} \boxed{S} \text{ depicts } \textit{the-car-stopped}$$

and

$$\boxed{\text{moving-car, contact}} \boxed{\text{contact}} \text{ depicts } \textit{the-car-hit-the-tree}.$$

For a formal treatment of depiction in terms of portrayal, we step up to sets of b-strings (as with ordinary non-branching strings). We turn a finite automaton (over  $\Sigma$ ) with set  $Q$  of states, set  $\rightarrow \subseteq Q \times \Sigma \times Q$  of transitions, initial state  $q_0 \in Q$  and set  $F \subseteq Q$  of final states into a *finite b-automaton* (over  $\Sigma$ ) by adding to the finite automaton a binary relation  $\xrightarrow{b} \subseteq Q \times Q$  on  $Q$  (intuitively marking realisations as well as temporal distance). We define a ternary relation  $\Rightarrow \subseteq Q \times \Sigma^b \times Q$  by

$$\begin{aligned} q &\xRightarrow{\alpha} q' && \text{iff } q \xrightarrow{\alpha} q' \\ q &\xRightarrow{ss'} q' && \text{iff } (\exists q'' \xrightarrow{s} q) q'' \xRightarrow{s'} q' \\ q &\xRightarrow{b(s, s')} q' && \text{iff } q \xrightarrow{s} q' \text{ and } (\exists q_1 \xleftarrow{b} q') (\exists q_2 \in F) q_1 \xRightarrow{s'} q_2 \end{aligned}$$

and say the finite b-automaton *accepts* a b-string  $s$  if for some  $q \in F$ ,  $q_0 \xRightarrow{s} q$ . An equivalent presentation of b-strings is provided by the notion

<sup>6</sup>For instance, the difference between the regular expressions  $\boxed{r}(\boxed{p} + \boxed{q})$  and  $\boxed{r} \boxed{p} + \boxed{r} \boxed{q}$  surfaces when  $\models$ -evaluating the formula  $\text{EX}(r \wedge \text{EX}p \wedge \text{EX}q)$  against the respective structures suggested by the regular expressions. Let  $S = \{0, 1, 2, 3, 4\}$ ,  $I(1) = I(2) = \{r\}$ ,  $I(3) = \{p\}$ ,  $I(4) = \{q\}$  and compare  $R = \{(0, 1), (1, 3), (1, 4)\}$  with  $R' = \{(0, 1), (1, 3), (0, 2), (2, 4)\}$ .

<sup>7</sup>We may impose the equations

$$\begin{aligned} (ss')s'' &= s(s's'') & b(b(s, s'), s') &= b(s, s') \\ b(ss', s'') &= sb(s', s'') & b(b(s, s'), s'') &= b(b(s, s''), s') \end{aligned}$$

which the automata-theoretic notions below (viz  $\xRightarrow{s}$ ) respect.

of a *b-form*  $\sigma$  (over  $\Sigma$ ), defined simultaneously with *b-possibilities*  $A$  by

$$\begin{aligned}\sigma & ::= (\alpha, A) \mid \sigma\sigma' \\ A & ::= \emptyset \mid \sigma + A\end{aligned}$$

for  $\alpha \in \Sigma$  with constant  $\emptyset$ . Each b-string  $\mathbf{s}$  has a b-form  $f(\mathbf{s}) = (\alpha_1, A_1) \cdots (\alpha_n, A_n)$  given by

$$\begin{aligned}f(\alpha) & = (\alpha, \emptyset) \\ f(\mathbf{s}\mathbf{s}') & = f(\mathbf{s})f(\mathbf{s}') \\ f(b(\mathbf{s}, \mathbf{s}')) & = (\alpha_1, A_1) \cdots (\alpha_{n-1}, A_{n-1})(\alpha_n, f(\mathbf{s}') + A_n).\end{aligned}$$

We can then view a finite b-automaton as a 2-sorted top-down tree automaton (e.g. Comon et al. (2002)), and acceptability of  $\mathbf{s}$  as derivability of  $q_0\mathbf{s} \rightarrow \mathbf{s}$  from the rules

$$\begin{aligned}q(\alpha, A) & \rightarrow (\alpha, q'A) && \text{for } q \xrightarrow{\alpha} q' \in F \\ q((\alpha, A)\sigma) & \rightarrow (\alpha, q'A)q'\sigma && \text{for } q \xrightarrow{\alpha} q' \\ q(\emptyset) & \rightarrow \emptyset \\ q(\sigma + A) & \rightarrow q'\sigma + qA && \text{for } q \xrightarrow{b} q' .\end{aligned}$$

Next, let us link  $\xrightarrow{b}$  to an interpretation of formulas  $\text{may}(\varphi)$  relative to the transitions  $\rightarrow \subseteq Q \times (\Sigma \cup \{b\}) \times Q$  of a finite b-automaton  $(Q, \rightarrow, q_0, F)$ , forming

- (a) timelines  $x : \mathbb{N} \rightarrow (Q \times \Sigma)$  such that for all  $n \geq 0$ ,  $q(n) \xrightarrow{\alpha(n)} q(n+1)$  where  $x(n) = (q(n), \alpha(n))$  [assuming wlog  $\text{domain}(\bigcup_{\alpha} \xrightarrow{\alpha}) = Q$ ]
- (b) the valuation  $\mathfrak{l} : (Q \times \Sigma) \rightarrow \Sigma$  mapping  $(q, \alpha)$  to  $\alpha$ , and
- (c) a binary relation  $\leftarrow$  on timelines  $x, x'$  from (a) given by

$$x' \leftarrow x \quad \text{iff} \quad q \xrightarrow{b} q' \quad \text{where } x(0) = (q, \alpha) \text{ and } x'(0) = (q', \alpha').$$

If we agree that

$$\leftarrow, x \models_{\mathfrak{l}} \text{may}(\varphi) \quad \text{iff} \quad (\exists x' \leftarrow x) \leftarrow, x' \models_{\mathfrak{l}} \varphi$$

then  $b(\square, \boxed{p})$  portrays  $\text{may}(p)$ ,<sup>8</sup> assuming the obvious extension of the notion of portrayal in the previous section to b-strings. (For a smooth generalization of the total timeline-valuation pairs mentioned in section 2, we must restrict the range of  $\mathfrak{l}$  to subsets  $\alpha$  of  $P$  such that  $(\forall p \in P) p \in \alpha$  iff  $\bar{p} \notin \alpha$ .)

---

<sup>8</sup>How does  $\text{may}$  apply to the counterfactual in (13) above? The modal branches forward in time, leaving the work of moving back (into the past) to the perfect (expressed by *have -en*), as in Condoravdi (2002). We have, for simplicity, confined ourselves in this paper to future operators. An obvious way to introduce past operators would double the domain of a timeline from  $\mathbb{N}$  to  $\mathbb{N} \cup \{-i \mid i \in \mathbb{N}\}$ .



What strings can we recover from a b-string? We can strip off the branches in a b-string  $s \in \Sigma^b$  to form the string  $s_{-b} \in \Sigma^+$ , setting  $\alpha_{-b} = \alpha$ ,  $(ss')_{-b} = (s_{-b})(s'_{-b})$ , and  $b(s, s')_{-b} = s_{-b}$ . We may also collect all possibilities in the language  $\pi(s) \subseteq \Sigma^+$  including  $s_{-b}$  alongside the branching possibilities  $\hat{\pi}(s)$  of  $s$

$$\begin{array}{ll} \pi(\alpha) = \{\alpha\} & \hat{\pi}(\alpha) = \emptyset \\ \pi(ss') = (s_{-b})\pi(s') + \hat{\pi}(s) & \hat{\pi}(ss') = (s_{-b})\hat{\pi}(s') + \hat{\pi}(s) \\ \pi(b(s, s')) = (s_{-b})\pi(s') + \pi(s) & \hat{\pi}(b(s, s')) = (s_{-b})\pi(s') + \hat{\pi}(s). \end{array}$$

Let us call a set of b-strings a *b-language*, and agree that a b-language is *regular* if it is the set of b-strings accepted by a finite b-automaton.

**Proposition 4.** *If  $L$  is a regular b-language, then the languages  $L_{-b} = \{s_{-b} \mid s \in L\}$  and  $L_\pi = \bigcup\{\pi(s) \mid s \in L\}$  are both regular.*

**Proof.** Given a finite b-automaton for  $L$ , we need only drop the relation  $\xrightarrow{b}$  to get a finite automaton for  $L_{-b}$ , and turn  $s \xrightarrow{b} s'$  to  $s \xrightarrow{\epsilon} s'$  for  $L_\pi$ .  $\dashv$

The elimination of realis distance  $\xrightarrow{b}$  in Proposition 4 takes us back to the “disconnection from the present” in CTL\*.

## References

- Asher, N. and A. Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.
- Beaver, D. and C. Condoravdi. 2003. A uniform analysis of *before* and *after*. In *Proc. Semantics and Linguistic Theory XIII*. Cornell Linguistics Circle Publications.
- Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Publications, Stanford.
- Bennett, B. and A. Galton. 2004. A unifying semantics for time and events. *Artificial Intelligence* 153:13–48.
- Blackburn, P., C. Gardent, and M. de Rijke. 1996. On rich ontologies for tense and aspect. In J. Seligman and D. Westerståhl, eds., *Logic, Language and Computation*, vol. 1, pages 77–92. CSLI Lecture Notes Number 58, Stanford.
- Clarke, E.M., O. Grumberg, and D.A. Peled. 1999. *Model Checking*. MIT Press, Cambridge, MA.
- Comon, H., M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2002. Tree automata techniques and applications. Available on: <http://www.grappa.univ-lille3.fr/tata>.

- Condoravdi, Cleo. 2002. Temporal interpretation of modals: Modals for the present and for the past. In D. Beaver, S. Kaufmann, B. Clark, and L. Casillas, eds., *The Construction of Meaning*, pages 59–88. CSLI, Stanford.
- Dowty, David R. 1979. *Word Meaning and Montague Grammar*. Reidel, Dordrecht.
- Emerson, E. Allen. 1992. Temporal and modal logic. In J. v. Leeuwen, ed., *Handbook of Theoretical Computer Science*, vol. B: Formal Methods and Semantics, pages 995–1072. MIT Press.
- Fernando, Tim. 2004a. A finite-state approach to events in natural language semantics. *Journal of Logic and Computation* 14(1):79–92.
- Fernando, Tim. 2004b. Inertia in temporal modification. In *Proc. Semantics and Linguistic Theory XIV*, pages 56–73. Cornell Linguistics Circle Publications.
- Fernando, T. and R. Nairn. 2005. Entailments in finite-state temporality. In *Proc. 6th International Workshop on Computational Semantics*, pages 128–138. Tilburg University.
- Heinämäki, O. 1972. Before. In *Papers from the 8th Regional Meeting of the Chicago Linguistic Society*, pages 139–151. University of Chicago.
- Kamp, H. and U. Reyle. 1993. *From Discourse to Logic*. Kluwer Academic Publishers, Dordrecht.
- Nelken, Rani and Nissim Francez. 1996. Automatic translation of natural language system specifications. In *CAV '96: Proceedings of the 8th International Conference on Computer Aided Verification*, pages 360–371. Springer-Verlag.
- Parsons, Terence. 1990. *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press, Cambridge, MA.
- Schubert, Lenhart. 2000. The situations we talk about. In J. Minker, ed., *Logic-Based Artificial Intelligence*, pages 407–439. Kluwer Academic Publishers, Dordrecht.
- Thomason, Richmond. 1984. Combinations of tense and modality. In D. Gabbay and F. Guenther, eds., *Handbook of Philosophical Logic*, pages 135–165. Reidel.
- Veltman, Frank. 1996. Defaults in update semantics. *J. Philosophical Logic* 25:221–261.
- Zimmermann, Thomas Ede. 2000. Free choice disjunction and epistemic possibility. *Natural Language Semantics* 8:255–290.

---

# On the formal semantics of begin and end of states in a model theory for temporal DRT

PETRA DÜNGES

## Abstract

In this paper we show that the intended meaning of begin and end of states is not embodied in the model theory of temporal DRT as given in *From Discourse to Logic*. As a consequence the non-continuous reading of the present perfect of statives is not expressed quite correctly by Kamp and Reyle. We introduce first order axioms for begin and end of states and events. Further to capture the intended meaning of begin and end of states two second order axioms are needed which say that the end of a state cannot overlap a similar state and that the begin of a state cannot overlap a similar state. This treatment of begin and end of states can be used not only for DRT but for other eventuality-based theories of temporal semantics as well.

**Keywords** FORMAL SEMANTICS, DRT, TEMPORAL SEMANTICS, MODEL THEORY, STATES, BEGIN, END

In DRT discourses are translated into discourse representation structures which are translated into formulas of first order logic. The model theory uses a Davidsonian eventuality-based semantics. The temporal analysis of sentences in DRT is based on a theory of aspect and a two-dimensional theory of tense.

The theory of aspect uses a system of verb classes: we have statives, accomplishments, achievements and activities. In this theory it is determined whether the eventuality described by a given sentence is an event or a state.

*FG-MoL 2005.*  
James Rogers (ed.).  
Copyright © 2009, CSLI Publications.

The motivation for the distinction between state- and event-describing sentences comes from the way temporal localization adverbs like **on Sunday** function. In an event-describing sentence like the following (with an accomplishment in the simple past tense)

Mary wrote the letter on Sunday

the localization time of the described eventuality is *included* in the time referred to by **on Sunday**. In contrast, in a state-describing sentence like the following (with an accomplishment in the past progressive)

Mary was writing the letter on Sunday

we only know that the localization time of the state in question *overlaps* with the time referred to by **on Sunday**. Note that Mary may have already been writing the letter on Saturday. The time referred to by a temporal localization adverb we call the **adverb time**.

After the described eventuality has been determined, it is localized in time via a two-dimension theory of tense. A central role is played by the **Temporal Perspective Point**, TPpt. Two relations are important in this theory: the relation in which TPpt stands with the *utterance time*  $n$  of the sentence in question, and the relation in which the *adverb time*  $i$  stands with TPpt. A suitable temporal perspective point has to be chosen out of the context of the preceding discourse. For a single sentence discourse,  $n$  is the only available candidate. In sentences with a temporal localization adverb **Adverb** we have the condition **Adverb**( $i$ ). If there is no temporal localization adverb present, no condition is put on  $i$ .

Now we concentrate on model theory, see Kamp and Reyle (1993, p. 667 ff).<sup>1</sup>

**Definition 1** A structure  $\mathcal{EV} = (EV, E, S, <, \circ)$  such that  $EV, E$  and  $S$  are sets,  $EV = E \cup S$ ,  $E \cap S = \emptyset$  and  $EV \neq \emptyset$  is called an **eventuality structure**.

$EV$  is called the set of **eventualities**,  $E$  the set of **events**,  $S$  the set of **states**. Eventualities are denoted by **ev**, events by **e** and states by **s**.

---

<sup>1</sup>One of the referees suggested that the work of Sylviane Schwer might be of relevance to this paper. Schwer (2004) shows an interesting connection between the eventuality structures of Kamp and S-languages. As she is working in a different framework, it will take substantial time to determine the exact relevance of her work to the present question of begin and end of states. Schwer does not treat states as arguments of predicates, so it is not to be expected that she has a solution to the non-continuous reading of the present perfect in the framework of DRT, however. I postpone investigations into Schwer (2004) for a later occasion.

The **precedence** relation,  $<$ , and the **overlap** relation,  $\circ$ , are binary relations on  $EV$  such that for all  $ev_i \in EV$ :

$$ev_1 < ev_2 \rightarrow \neg ev_2 < ev_1 \quad (P1)$$

$$(ev_1 < ev_2 \wedge ev_2 < ev_3) \rightarrow ev_1 < ev_3 \quad (P2)$$

$$ev_1 \circ ev_1 \quad (P3)$$

$$ev_1 \circ ev_2 \rightarrow ev_2 \circ ev_1 \quad (P4)$$

$$ev_1 < ev_2 \rightarrow \neg ev_2 \circ ev_1 \quad (P5)$$

$$(ev_1 < ev_2 \wedge ev_2 \circ ev_3 \wedge ev_3 < ev_4) \rightarrow ev_1 < ev_4 \quad (P6)$$

$$ev_1 < ev_2 \vee ev_1 \circ ev_2 \vee ev_2 < ev_1 \quad (P7)$$

Eventualities are localized in time at intervals. Intervals are convex sets of instants. A **punctual interval**  $i$  is of the form  $i = \{t\}$  where  $t$  is an instant;  $n$  and **TPpt** are punctual intervals. The precedence relation  $<$  on the set of instants is asymmetric, transitive and linear. The interval structure associated with the instant structure  $\mathcal{T}$  is denoted by  $\mathcal{INT}(\mathcal{T})$ . Precedence,  $<$ , and overlap,  $\circ$ , between intervals  $i_1, i_2$  are defined as follows:  $i_1 < i_2$  if  $t_1 < t_2$  for all  $t_1 \in i_1, t_2 \in i_2$ , and  $i_1 \circ i_2$  if there is a  $t \in i_1 \cap i_2$ . The localization function  $LOC : \mathcal{EV} \rightarrow \mathcal{INT}(\mathcal{T})$  from eventualities to intervals is a homomorphism with respect to precedence and overlap.<sup>2</sup>

In Dünges (1998) it is shown that the following holds:

**Proposition 1**

$$ev_1 \circ ev_2 \leftrightarrow LOC(ev_1) \circ LOC(ev_2) \quad (i)$$

$$ev_1 < ev_2 \leftrightarrow LOC(ev_1) < LOC(ev_2) \quad (ii)$$

A sentence with a stative in the simple present is state-describing and has the temporal property  $TPpt = n, i = TPpt$ . So sentence (4.1) gets formula (4.2). See figure 1.

$$\text{Mary lives in Amsterdam} \quad (4.1)$$

$$\exists i, s, x (i = n \wedge LOC(s) \circ i \wedge \text{Mary}(x) \wedge \text{live-in-Amsterdam}(s, x)) \quad (4.2)$$

Begin and end of states and events play a role, too. A linguistic motivation for the introduction of the end of a state is the treatment Kamp and Reyle give to the present perfect of statives in sentences without temporal localization adverbs. Such a sentence is state-describing and

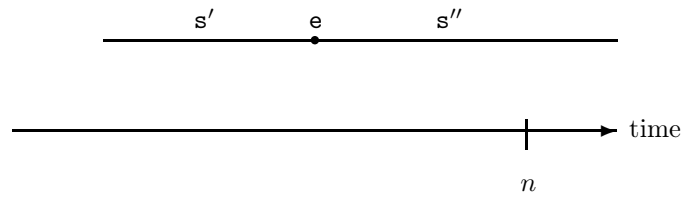
---

<sup>2</sup>There is more to say about LOC, for the present article LOC being a homomorphism suffices, however. For further conditions on LOC see Kamp and Reyle (1993), Dünges (1998) and Dünges (2001).

FIGURE 1: A model for a sentence with a stative in the simple present



FIGURE 2: A model for a sentence with a stative in the present perfect



has the temporal property  $\text{TPpt} = n, i = \text{TPpt}$ . Here the described eventuality is the result state  $s''$  of a state  $s'$  that is referred to by the stative and  $s''$  is triggered by the *termination* of the underlying state  $s'$ . Thus the following sentence

$$\text{Mary has lived in Amsterdam} \quad (4.3)$$

is describing a state that comes about through “*termination of the state of living in Amsterdam*”, see Kamp and Reyle (1993, p. 567). We call this interpretation of the present perfect of statives the **non-continuative reading**. In Kamp and Reyle (1993, p. 580) we find essentially the following formula for (4.3), where  $\supset$  stands for the abut relation. See figure 2.

$$\begin{aligned} \exists i, s', s'', e, x (i = n \wedge \text{LOC}(s'') \circ i \wedge e = \text{end}(s') \wedge e \supset s'' \\ \wedge \text{Mary}(x) \wedge \text{live-in-Amsterdam}(s', x)) \end{aligned} \quad (4.4)$$

By the way, formula (4.4) and formula (4.2) can both be true in the same model at  $n$ . This is as it should be, as intuitively the truth of *Mary has lived in Amsterdam* does not speak against the truth of *Mary lives in Amsterdam*, if both sentences are uttered at the same time: Mary may have come back to Amsterdam once she left it.

Kamp and Reyle mention, that  $e \supset s''$  means that  $s''$  “*starts the very moment  $e$  ends*”, but do not give a formal definition of the abut relation

or a complete formal characterization of begin and end of eventualities in Kamp and Reyle (1993).

Let us start with some first order axioms for begin and end of eventualities. Kamp and Reyle introduce the functions  $\text{begin} : \text{EV} \rightarrow \text{E}$  and  $\text{end} : \text{EV} \rightarrow \text{E}$  in their definition of a *model*, see Kamp and Reyle (1993, p. 677 ff.). The only conditions they put on these functions are the following

$$\text{begin}(\text{begin}(\text{ev})) = \text{begin}(\text{ev}) \quad (\text{BB})$$

$$\text{end}(\text{end}(\text{ev})) = \text{end}(\text{ev}) \quad (\text{EE})$$

More than that is needed to capture the intended meaning of begin and end of eventualities, however. One intuition that must be reflected by our theory is that begin and end of an event belong to that event, but begin and end of a state do not belong to that state. Why is this so? Consider an event  $\mathbf{e}$  described by *Mary wrote the letter*. Assume that  $\text{LOC}(\mathbf{e}) = [t_1, t_2]$ . Then intuitively there is a state  $\mathbf{s}$  described by *Mary was writing the letter* and  $\text{LOC}(\mathbf{s}) = ]t_1, t_2[$ . Meaning postulates for the progressive can take care of this. Intuitively we want  $\text{LOC}(\text{begin}(\mathbf{e})) \subseteq \text{LOC}(\mathbf{e})$  but  $\text{LOC}(\text{begin}(\mathbf{s})) < \text{LOC}(\mathbf{s})$ . In addition, no temporal gaps should be allowed between the begin of a state and the state on the one hand and the state and the end of the state on the other hand.

In order to capture the no-gap intuition we need the **adjacency**-relation,  $\sqsupset$ . We define the adjacency relation between eventualities via the adjacency relation between intervals as follows:

$$i_1 \sqsupset i_2 \text{ iff } i_1 < i_2 \wedge i_1 \cup i_2 \in \text{INT}(\mathcal{T}) \quad (\sqsupset 1)$$

$$\text{ev}_1 \sqsupset \text{ev}_2 \text{ iff } \text{LOC}(\text{ev}_1) \sqsupset \text{LOC}(\text{ev}_2) \quad (\sqsupset 2)$$

To begin with, we introduce a partition of the set of events,  $\text{E}$ , into a set of **punctual events**,  $\text{PE}$ , and a set of **non-punctual events**,  $\text{NPE}$ . Punctual events are localized at punctual intervals, non-punctual events at non-punctual intervals. Non-punctual events consist of three parts: begin, middle and end. The **constitutes function**  $+$  :  $\text{PE} \times \text{S} \times \text{PE} \rightarrow \text{NPE}$  is a partial function, which puts together these three parts of a non-punctual event. Begin and end of eventualities are punctual events, the middle of an event is a state.

We require the fulfillment of the following axioms:

$$\forall \mathbf{e} \in \text{PE} (\text{begin}(\mathbf{e}) = \mathbf{e} = \text{end}(\mathbf{e})) \quad (\text{LEV1})$$

$$\forall \mathbf{e} \in \text{NPE} (\text{begin}(\mathbf{e}) \sqsupset \text{middle}(\mathbf{e})) \quad (\text{LEV2})$$

$$\forall \mathbf{e} \in \text{NPE} (\text{middle}(\mathbf{e}) \sqsupset \text{end}(\mathbf{e})) \quad (\text{LEV3})$$

$$\forall \mathbf{e} \in \text{NPE}(\mathbf{e} = +(\text{begin}(\mathbf{e}), \text{middle}(\mathbf{e}), \text{end}(\mathbf{e}))) \quad (\text{LEV4})$$

$$\forall \mathbf{e} \in \text{NPE}(\text{LOC}(\mathbf{e}) = \text{LOC}(\text{begin}(\mathbf{e})) \cup \text{LOC}(\text{middle}(\mathbf{e})) \cup \text{LOC}(\text{end}(\mathbf{e}))) \quad (\text{LEV5})$$

$$\forall \mathbf{s} \in \text{S}(\text{begin}(\mathbf{s}) \sqsubseteq \mathbf{s}) \quad (\text{LEV6})$$

$$\forall \mathbf{s} \in \text{S}(\mathbf{s} \sqsubseteq \text{end}(\mathbf{s})) \quad (\text{LEV7})$$

Conditions (BB) and (EE) of Kamp and Reyle are met by our functions `begin` and `end` as well, because the `begin` and the `end` of an eventuality are punctual events and we have axiom (LEV1).

Now we can define the **abut**-relation  $\supset$  as follows:

$$\text{ev}_1 \supset \text{ev}_2 \text{ iff } \text{LOC}(\text{end}(\text{ev}_1)) = \text{LOC}(\text{begin}(\text{ev}_2)) \quad (\supset)$$

Note that there is a subtle difference between the adjacency and the abut relation. The adjacency relation between eventualities does not allow for a time gap between them. In contrast, if  $\mathbf{s} \supset \mathbf{s}'$  holds, then there is a tiny time gap between the two states, given by  $\text{LOC}(\text{begin}(\mathbf{s}'))$ .

In order to get inferences like the following:

$$\text{Mary wrote the letter} \models \text{Mary was writing the letter}$$

we may introduce meaning postulates:

$$\text{write}(e, x, y) \rightarrow (\text{PROG}(\text{write}))(\text{middle}(e), x, y) \quad (\text{mp:write})$$

Let us come back to our example **Mary has lived in Amsterdam**. Consider a model  $\mathfrak{M}$  and an assignment  $\beta$  such that  $(\beta(i), \beta(s'), \beta(s''), \beta(e), \beta(x))$  is a witness for the truth of (4.4) in  $(\mathfrak{M}, \beta)$ . Let  $\mathbf{s}' = \beta(s')$ ,  $\mathbf{s}'' = \beta(s'')$  and  $\mathbf{e} = \beta(e)$ . We have  $\mathbf{e} = \text{end}(\mathbf{s}')$ . We get  $\text{end}(\text{end}(\mathbf{s}')) = \text{end}(\mathbf{s}')$ , by axiom (EE). Thus  $\mathbf{e} \supset \mathbf{s}''$  means that  $\text{LOC}(\text{end}(\mathbf{s}')) = \text{LOC}(\text{begin}(\mathbf{s}''))$ .

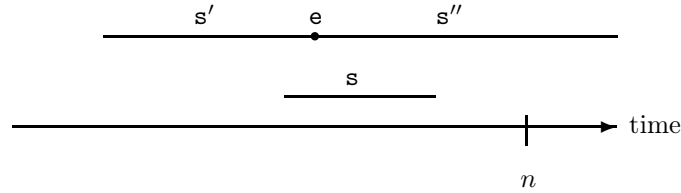
At first glance everything seems to be in order. But note that the result state that is described by **Mary has lived in Amsterdam** is triggered by the *termination* of some stay of Mary's in Amsterdam. And the termination condition is not mirrored in all models for the corresponding formulas.

Where is the problem? Consider again our witness for the truth of (4.4) in  $(\mathfrak{M}, \beta)$ . Assume that there is a state  $\mathbf{s}$  such that  $\mathbf{s} = \beta(s)$  and  $(\mathfrak{M}, \beta) \models \text{live-in-Amsterdam}(s, x)$ . Further, assume that  $(\mathfrak{M}, \beta) \models \text{end}(s') \circ s$ . See figure 3. Note that  $(\mathfrak{M}, \beta) \models \text{live-in-Amsterdam}(s', x)$ . Nevertheless, we cannot say, that Mary's life in Amsterdam is terminated at  $\text{LOC}(\text{end}(s'))$ .

The trouble is that  $(\mathfrak{M}, \beta) \models \text{live-in-Amsterdam}(s, x)$  and  $\mathbf{s} \circ \text{end}(s')$ . So **Mary lives in Amsterdam** is true in that model if uttered



FIGURE 3: An intuitively undesirable model for a sentence with a stative in the present perfect



at  $\text{LOC}(\text{end}(s'))$ . Thus the intended interpretation of the termination condition is not reflected in our model.

How can we exclude unwanted models like this one? This question points us to a deeper problem. Namely, it shows that we have still not said all there is to say about the end (and begin) of states.

So far in our axioms about begin and end we have only been concerned with eventualities themselves, but not with the facts they describe. It is time now that we bring them into play. This can be done by considering states with description.

**Definition 2** *Let  $\mathfrak{M}$  be a model and  $\beta$  an assignment in  $\mathfrak{M}$ . Let  ${}^sX^n$  be a variable for a  $n$ -place state predicate and  $x_1, \dots, x_n$  variables for individuals. Then  ${}^sX^n(-, x_1, \dots, x_n)$  is called a **description**.*

*Let  $s$  be a state in  $\mathfrak{M}$ . Let  $s = \beta(s)$  and  $(\mathfrak{M}, \beta) \models {}^sX^n(s, x_1, \dots, x_n)$ . Then  $s$  is called a **state with description**  ${}^sX^n(-, x_1, \dots, x_n)$  **in  $\mathfrak{M}$  relative to  $\beta$** . Let  $s$  and  $s'$  be states with the same description in  $\mathfrak{M}$  relative to  $\beta$ . Then  $s$  and  $s'$  are called **similar states**.*

In the undesirable model for *Mary has lived in Amsterdam* that was depicted in figure 3,  $s$  and  $s'$  are similar states.

We require that the end of a state with description cannot overlap with a similar state. And we require that the begin of a state with description cannot overlap with a similar state. The following two second order axioms do this job.

$${}^sX^n(s, x_1, \dots, x_n) \rightarrow (\neg \exists s' ({}^sX^n(s', x_1, \dots, x_n) \wedge \text{LOC}(\text{end}(s)) \circ \text{LOC}(s')))) \quad (\text{No} \circ \text{EndS})$$

$${}^sX^n(s, x_1, \dots, x_n) \rightarrow (\neg \exists s' ({}^sX^n(s', x_1, \dots, x_n) \wedge \text{LOC}(\text{begin}(s)) \circ \text{LOC}(s')))) \quad (\text{No} \circ \text{BeginS})$$

This move solves our problem with the non-continuous reading of the present perfect of statives. But it is more than a technical trick for a

problem with one type of sentences. These axioms capture something important about the meaning of begin and end of states, as they help to express the idea that a certain condition is *terminated* (at the end of a state describing that condition) or *started* (at the begin of a state describing that condition).

Alternatively we may consider the following two axioms:

$$\begin{aligned} {}^s X^n(s, x_1, \dots, x_n) \rightarrow (\neg \exists s' ({}^s X^n(s', x_1, \dots, x_n) \wedge s \circ s' \\ \wedge \neg \text{LOC}(s) = \text{LOC}(s'))) \quad (\text{No}\circ\text{S}) \end{aligned}$$

$$\begin{aligned} {}^s X^n(s, x_1, \dots, x_n) \rightarrow (\neg \exists s' ({}^s X^n(s', x_1, \dots, x_n) \wedge s \sqsupset s') \wedge \\ \neg \exists s' ({}^s X^n(s', x_1, \dots, x_n) \wedge s' \sqsupset s)) \quad (\text{No}\sqsupset\text{S}) \end{aligned}$$

**Proposition 2** *The fulfillment of both (No $\circ$ BeginS) and (No $\circ$ EndS) implies the fulfillment of both (No $\circ$ S) and (No $\sqsupset$ S) and vice versa.*

These axioms are quite strong. At first glance it might seem that they are interfering with the so called **principle of homogeneity** or the **subinterval property**, as proponents of interval semantics would have it.

The subinterval property is formulated by Dowty (1986, p. 42) as follows: “A sentence is stative iff it follows from the truth of  $\phi$  at an interval  $I$  that  $\phi$  is true at all subintervals of  $I$ . (e.g. if John was asleep from 1:00 to 2:00 PM. then he was asleep at all subintervals of this interval ...)”

As temporal DRT is not a theory of interval semantics, where the truth of a sentence is evaluated with respect to intervals of time, the subinterval property can not be rendered *literally* into the framework of DRT. But it might be tempting to introduce the concept of substates and to require the fulfillment of the following second order axiom, see figure 4.

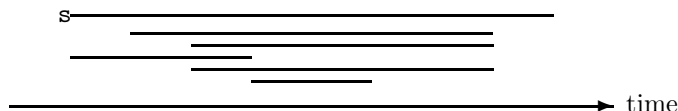
$$\begin{aligned} ({}^s X^n(s, x_1, \dots, x_n) \wedge i \subseteq \text{LOC}(s)) \rightarrow \exists s' ({}^s X^n(s', x_1, \dots, x_n) \\ \wedge i = \text{LOC}(s')) \quad (\text{SUB}) \end{aligned}$$

**Definition 3** *Let  $\mathbf{s}$  and  $\mathbf{s}'$  be similar states such that  $\text{LOC}(\mathbf{s}') \subseteq \text{LOC}(\mathbf{s})$  then  $\mathbf{s}'$  is called a **substate** of  $\mathbf{s}$ . If  $\text{LOC}(\mathbf{s}') \subset \text{LOC}(\mathbf{s})$  then  $\mathbf{s}'$  is called a **proper substate** of  $\mathbf{s}$ .*

But note the following result

**Proposition 3** *Let  $\mathbf{s}$  be a state with description. Let  $i \subset \text{LOC}(\mathbf{s})$  and  $i \neq \text{LOC}(\mathbf{s})$ .*

*Then (SUB) does not hold in presence of axiom (No $\circ$ S).*

FIGURE 4: A state  $s$  with some substates

No harm is done, however, as we do not need axiom (SUB). We will demonstrate this in the following.

So, how do we treat examples that require the principle of homogeneity, without having axiom (SUB) at our disposal? We propose that in the framework of DRT the sentence

$$\text{John was asleep from 1:00 to 2:00} \quad (4.5)$$

should be translated in the following way:

$$\begin{aligned} \exists i, x, s (i < n \wedge i \subseteq \text{LOC}(s) \wedge \text{from1:00-to2:00}(i) \\ \wedge \text{asleep}(s, x) \wedge \text{John}(x)) \end{aligned} \quad (4.6)$$

We require that  $\text{from1:00-to2:00}(i)$  means that  $i = [t_1, t_2]$  where  $t_1, t_2$  are the instants referred to by “at 1:00” and “at 2:00”, respectively. The unusual condition  $i \subseteq \text{LOC}(s)$  is necessary for the temporal localization adverb **from 1:00 to 2:00**. Note that for an adverb like **on Sunday** we use  $\text{LOC}(s) \circ i$  - but for **from 1:00 to 2:00** this condition would yield plainly wrong results.

Now given a witness  $w = (\beta(i), \beta(x), \beta(s))$  for (4.6) and a standard semantics for terms like “at 1:00”, then  $w$  is a witness, too, for the formulas for

$$\text{John was asleep from A to B} \quad (4.7)$$

for all  $A, B$  such that  $1:00 \leq A < B \leq 2:00$ .

Thus in DRT we do not need an axiom like (SUB) to capture the gist of the principle of homogeneity.

Moreover, note that (SUB) introduces spurious substates which seem to have no place in a truly eventuality-based theory of temporal semantics. Thus we can do away with that axiom without having to shed a single tear.

## References

Dowty, David R. 1986. The effects of aspectual class on the temporal structure of discourse: Semantics of pragmatics. *Linguistics and Philosophy* Vol.

9 9:37–61.

Dünges, Petra. 1998. Eventualities in time. CLAUS report 104, Institute for Computational Linguistics, University of the Saarland, P.O. Box 151150, D-66041 Saarbrücken, Germany. 55 pages.

Dünges, Petra. 2001. Eventualities in time – the localization of eventualities in the platonist and the reductionist picture of time. *Grammars* 4(1):69–83.

Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic*. Dordrecht, Boston, London: Kluwer Academic Publishers.

Schwer, Sylviane. 2004. Traitement de la temporalite des discours: une analyse situs. Chronos VI, International colloquium on syntax, semantics and pragmatics of tense, mood and aspect, Geneva, Switzerland, 22–24 Sept. 2004.

---

# How to Define Simulated Annealing for Optimality Theory?

TAMÁS BÍRÓ <sup>†</sup>

## Abstract

*Optimality Theory* (OT) requires an algorithm optimising the *Harmony function* on the set of candidates. *Simulated annealing*, a well-known heuristic technique for combinatorial optimisation, has been argued to be an empirically adequate solution to this problem. In order to generalise simulated annealing for a non-real valued Harmony function, two representations of a violation profile are proposed: by using polynomials and ordinal numbers.

**Keywords** OPTIMALITY THEORY, HEURISTIC COMBINATORIAL OPTIMIZATION, SIMULATED ANNEALING, ORDINAL NUMBERS, POLYNOMIALS

## 5.1 Optimality Theory and optimisation

A grammar in Optimality Theory (Prince and Smolensky (2004), aka Prince and Smolensky (1993)) consists of two modules, *Gen* and *Eval*. The input—the underlying representation *UR*—is mapped by *Gen* onto a *set of candidates*  $Gen(UR)$ , reflecting language typology. For each language, the language-specific *Eval* chooses the element (or elements) appearing as the surface form.

*Eval* is usually perceived as a *pipeline*, in which constraints filter out sub-harmonic candidates. Each constraint assigns violation marks to the candidates in its input, and all candidates with more marks than some other ones are out of the game. Nonetheless, *Eval* can also be seen

---

<sup>†</sup>I wish to acknowledge the support of the University of Groningen’s Program for High-Performance Computing; as well as thank the following people for valuable discussions: Gosse Bouma, Gertjan van Noord, Krisztina and Balázs Szendrői.

as a function assessing the candidates for *Harmony*: the most harmonic one will surface in the language.

A constraint  $C_i$  is a function mapping from the candidate set to the set of non-negative integers. The (universal) constraints are ranked into a (language-specific) *hierarchy*:  $C_N \gg C_{N-1} \gg \dots \gg C_0$ . Eval assigns a vector (a *violation profile*, a *Harmony value*) to each candidate  $w$ :

$$H(w) = \left( C_N(w), C_{N-1}(w), \dots, C_0(w) \right) \in \mathbb{N}_0^{N+1} \quad (5.8)$$

Eval also includes an *optimisation process* that finds the optimal candidate(s), and returns it (them) as the surface representation corresponding to underlying form  $UR$ :

$$SR(UR) = \operatorname{argopt}_{w \in \operatorname{Gen}(UR)} H(w) \quad (5.9)$$

Here, optimisation is with respect to *lexicographic ordering*. *Lexicographic ordering* of vectors is the way words are sorted in a dictionary: first compare the first elements of the vectors, then, if they are the same, compare the second ones, and so on. Formally speaking:

**Definition 5**  $H(w_1)$  is *more optimal* (*more harmonic*) than  $H(w_2)$  ( $H(w_1) \succ H(w_2)$ ), or simply candidate  $w_1$  is *better* than  $w_2$  ( $w_1 \succ w_2$ ), if and only if there exists  $k \in \{N, N-1, \dots, 0\}$  such that

1.  $C_k(w_1) < C_k(w_2)$ ; and
2. for all  $j \in \{N, N-1, \dots, 0\}$ , if  $j > k$  then  $C_j(w_1) = C_j(w_2)$ .

Two violation profiles are *equal* ( $H(w_1) = H(w_2)$ ), two candidates are *equivalent*:  $w_1 \simeq w_2$ ) iff for all  $j \in \{N, N-1, \dots, 0\}$ ,  $C_j(w_1) = C_j(w_2)$ .

We shall call the constraint  $C_k$ , which determines the relative ordering of  $H(w_1)$  and  $H(w_2)$ , the *fatal constraint* (the highest ranked constraint with uncanceled marks).

This definition follows from the filtering approach: being worse on a higher ranked constraint cannot be compensated by a better behaviour on lower ranked constraints. This phenomenon is called the *categorical ranking* of the constraints (*Strict Domination Hypothesis*), and is probably a major reason why OT has become so popular.

The following three properties can be shown from definition 5 directly (Bíró, forthcoming); by them, the soundness of Eq. (5.9) follows:<sup>1</sup>

**Theorem 6** *The set of violation profiles is a well ordered set, namely:*

- **TRANSITIVITY**: *if  $w_1 \succ w_2$  and  $w_2 \succ w_3$ , then  $w_1 \succ w_3$  also holds.*

---

<sup>1</sup>Importantly, the proof of the last statement requires the set of possible violation levels—the range of each constraint—form a well ordered set. This criterion is met in our case, since the violation levels are non-negative integers.

- LAW OF TRICHOTOMY: *for any two candidates  $w_1$  and  $w_2$ , exactly one of the following three statements holds:*
  1.  $H(w_1) \prec H(w_2)$  (that is,  $w_1 \prec w_2$ );
  2.  $H(w_1) \succ H(w_2)$  (that is,  $w_1 \succ w_2$ );
  3.  $H(w_1) = H(w_2)$  (that is,  $w_1 \simeq w_2$ ).
- THE EXISTENCE OF A MOST OPTIMAL SUBSET: *Let  $S$  be a set of candidates. Then,  $S$  has a subset  $S_0 \subseteq S$  such that*
  1. *if  $w_1 \in S_0$  and  $w_2 \in S_0$ , then  $H(w_1) = H(w_2)$ ;*
  2. *if  $w_1 \in S_0$  and  $w_3 \in S \setminus S_0$ , then  $w_1 \succ w_3$ .*

Optimality Theory poses the following computational challenge: what algorithm realises the optimisation required by Eval? Eisner (2000) demonstrates that finding the optimal candidate is OptP-complete. In addition, numerous linguistic models use an infinite candidate set. Several solutions have been proposed, although each of them is built on certain presuppositions, and they require large computational resources. Finite state techniques (e.g. Ellison (1994), Frank and Satta (1998), Karttunen (1998), Gerdemann and van Noord (2000), Bíró (2003)) not only require Gen and the constraints to be finite state, but work only with some further restrictions. *Chart parsing (dynamic programming*, e.g. Tesar and Smolensky (2000), Kuhn (2000)) has assumptions met by most linguistic models, but also requires a relatively large memory. Similar applies to genetic algorithms (Turler, 1994).

A cognitively adequate optimisation algorithm, however, does not have to be exact. Speech is full of errors, and (a part of) the performance errors could be the result of the optimisation process returning erroneous outputs. Yet, a cognitively adequate algorithm should always return *some* response within constant time, since the conversation partners are not computer users used to watch the sandglass.

This train of thought leads to heuristic optimisation techniques, defined by Reeves (1995) as “a technique which seeks good (i.e. near-optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is.” In this paper, we implement Optimality Theory by using the simplest heuristic optimisation technique, *simulated annealing*, and introduce the *Simulated Annealing for Optimality Theory algorithm* (SA-OT).

The SA-OT algorithm will, under normal conditions, find the “correct”, i.e. the grammatical output—the optimal element of the candidate set—with high probability, within constant time, using only a very restricted memory. Human speakers sometimes speed up the computational algorithm, and the price is paid in precision: we propose to see

(some) fast speech phenomena as decreased precision of Eval due to the increased speed. Similarly, by speeding up SA-OT, the chance of finding suboptimal, yet “good (i.e. near-optimal) solutions” increases. The models of fast speech phenomena thus constructed support the cognitive adequateness of SA-OT (Bíró (2004), Bíró (forthcoming)).

## 5.2 Heuristic Optimisation with Simulated Annealing

*Simulated annealing*, also referred to as *Boltzmann Machines* or as *stochastic gradient ascent*, is a wide-spread stochastic technique for combinatorial optimisation (e.g. Reeves (1995)). Only few have applied simulated annealing in linguistics, most of them for parsing (e.g. Selman and Hirst (1985), Howells (1988), Kempen and Vosse (1989), Selman and Hirst (1994)). It may also be found in the pre-history of Optimality Theory (Smolensky, 1986) and in later work on *Harmonic grammar*—including *Maximum Entropy* models of OT (Jäger, 2003)—, though usually related to grammar learning. To our best knowledge, it has never been applied within the standard OT paradigm, especially for finding the optimal candidate.

Simulated Annealing searches for the state of a system minimising the cost function  $E$  (Energy or Evaluation) by performing a random walk in the search space. If the rule were to move always downhill (*gradient descent*), then the system would very easily be stuck in local minima. Therefore, we also allow moving upwards with some chance, which is higher in the beginning of the simulation, and which then diminishes. The control parameter  $T$  determining the uphill moves is called “temperature”, because the idea proposed independently by Kirkpatrick et al. (1983) and by Černý (1985) originates in statistical physics (Metropolis et al., 1953).

The random walk is launched from an initial state  $w_0$ . At each time step, a random neighbour state ( $w'$ ) of the actual state  $w$  is picked. We need, thus, to have a *topology* on the search space that defines the neighbours of a state (the *neighbourhood structure*), as well as the *a priori* probability distribution determining the choice of a neighbour in each step. Subsequently, we compare  $w'$  to  $w$ , and the random walker moves from  $w$  to  $w'$  with probability  $P(w \rightarrow w' | T)$ , where  $T$  is the temperature at that moment of the simulation. (A random number  $r$  is generated between 0 and 1, and if  $r < P(w \rightarrow w' | T)$ , the random walker moves.) If  $E(w)$  is the cost function to minimise, then:

$$P(w \rightarrow w' | T) = \begin{cases} 1 & \text{if } E(w') \leq E(w) \\ e^{-\frac{E(w') - E(w)}{T}} & \text{if } E(w') > E(w) \end{cases} \quad (5.10)$$



Moving downhill is always possible, and moving uphill depends on the difference in  $E$  and on the temperature  $T$ . At the beginning of the simulation,  $T$  is assigned a high value, making any move very likely. The value of  $T$  is then decreased gradually, while even the smallest jump does not become highly improbable. When the temperature has reached its lowest value, the algorithm returns the state—a local minimum—into which the random walker is “frozen”. Obviously, nothing guarantees finding the global minimum, but the slower the *cooling schedule* (the more iterations performed), the higher the probability to find it.

### 5.3 Simulated Annealing for OT: the basic idea

How to combine simulated annealing with Optimality Theory? The search space is the candidate set, as defined by standard OT. Yet, a *neighbourhood structure* (a *topology*) should be added—an unknown concept in OT literature—in order to determine how to pick the next candidate. We propose to consider two candidates as neighbours if they differ only minimally: if a *basic operation* transforms one into the other. What a basic operation is depends on the problem, but should be a naturally fitting choice. It is the neighbourhood structure that determines which candidates are *local* optima, which may be returned as erroneous outputs. Thus, the definition of the topology is crucial to account for speech errors.

If the topology determines the horizontal structure of the landscape in which the random walker roves, the Harmony function adds its vertical structure. Here again, standard Optimality Theory provides only the first part of the story. The transition probability  $P(w \rightarrow w' \mid T) = 1$  if  $w'$  is better than  $w$  (*i.e.*,  $H(w') \succ H(w)$ ). But how to define the transition probability to a worse candidate, in function of the temperature  $T$ ? How to adopt Eq. (5.10)? What is  $H(w') - H(w)$ , let alone its exponent? And what should temperature look like?

Equation (5.10) provides the meaning of temperature:  $T$  defines the range of  $E(w') - E(w)$  above which no uphill jump is practically possible ( $P(w \rightarrow w' \mid T) \approx 0$ , if  $E(w') - E(w) \gg T$ ), and below which uphill moves are allowed ( $P(w \rightarrow w' \mid T) \approx 1$ , if  $E(w') - E(w) \ll T$ ). In turn, we *first* have to define the difference  $H(w') - H(w)$  of two violation profiles, *then* introduce temperature for OT in an analogous way. Last, we can adjust Eq. (5.10) and formulate the SA-OT algorithm.

Two approaches—two representations of the violation profile—are proposed in order to carry out this agenda. Both may have its adherents and its opponents. And yet, both approaches lead to the same algorithm.

## 5.4 Violation profiles as polynomials

As mentioned, a crucial feature of Optimality Theory is *strict domination*: a candidate suboptimal for a higher ranked constraint can never win, even if it satisfies the lower ranked constraints best. Prince and Smolensky (2004) present why the Harmony function  $H(w)$  satisfying strict domination cannot be realised with a real-valued function.

Suppose first that an upper bound  $q > 0$  exists on the number of violation marks a constraint can assign to a candidate. The possible levels of violation are  $0, 1, \dots, q - 1$ . Then, the following real-valued Energy function  $E(w)$  realises the Harmony  $H(w)$  known from (5.8):

$$E(w) = C_N(w) \cdot q^N + C_{N-1}(w) \cdot q^{N-1} + \dots + C_1(w) \cdot q + C_0(w) \quad (5.11)$$

$E(w)$  realising  $H(w)$  means that for all  $w_1$  and  $w_2$ ,  $E(w_1) \leq E(w_2)$  if and only if  $H(w_1) \succeq H(w_2)$ . In other words, optimising the Harmony function is equivalent to minimising the Energy function. Observe that  $E(w)$  with a lower  $q$  does not necessarily realise  $H(w)$ .

However, nothing in general guarantees that such an upper bound exists: Eq. (5.11) with a given  $q$  is only an approximation. Then, let us represent the violation profiles as polynomials of  $q \in \mathbb{R}^+$ :

$$E(w)[q] = C_N(w) \cdot q^N + C_{N-1}(w) \cdot q^{N-1} + \dots + C_1(w) \cdot q + C_0(w) \quad (5.12)$$

and consider the behaviour of  $E(w)[q]$  as  $q$  goes to infinity! Yet,  $E(w)[q]$  also goes to infinity as  $q$  grows boundless:  $\lim_{q \rightarrow \infty} E(w)[q] = +\infty$ .

The trick is to perform an operation first, or to check the behaviour of the energy function first, and only subsequently bring  $q$  to the infinity. By performing *continuous* operations, it makes sense to change the order of the operation and of the limit to infinity.

First, let us compare two violation profiles seen as polynomials. The following definition—comparing the limits—is meaningless:  $w_1 \succ w_2$  iff  $\lim_{q \rightarrow \infty} E(w_1)[q] < \lim_{q \rightarrow \infty} E(w_2)[q]$ . We can, however, consider the limit of the comparison, instead of the comparison of the limits:

**Definition 6**  $E(w_1) \prec E(w_2)$  if and only if

$$\begin{aligned} &\text{either } \lim_{q \rightarrow +\infty} (E(w_2)[q] - E(w_1)[q]) > 0, \\ &\text{or } \lim_{q \rightarrow +\infty} (E(w_2)[q] - E(w_1)[q]) = +\infty. \end{aligned}$$

Furthermore,  $E(w_1) = E(w_2)$  iff  $E(w_1)[q] = E(w_2)[q]$  for all  $q \in \mathbb{R}^+$ .

Energy-polynomials with this definition of  $\prec$  realise the Harmony function:  $E(w_1) \preceq E(w_2)$  if and only if  $H(w_1) \succeq H(w_2)$ . For a proof, see the Appendix and BÍRÓ (forthcoming). Consequently, the *polynomial representation* of the Harmony function is well-founded.

Can we use energy polynomials to formulate simulated annealing for Optimality Theory? As explained, the role of temperature in simulated annealing is to define a magnitude above which counter-optimal transitions are improbable, and below which they are very probable. Thus, temperature must have the same type (dimension, form) as the function to optimise. If, in our case, the energy function takes different polynomials as values, then  $T$  should also be polynomial-like:

$$T[q] = \langle K_T, t \rangle [q] = t \cdot q^{K_T} \quad (5.13)$$

Temperature  $T = \langle K_T, t \rangle$  looks as if it were a violation profile that has incurred  $t$  marks from a constraint—if some constraint has  $K_T$  as index. But temperature can be more general: we only require  $t \in \mathbb{R}^+$ , whereas  $K_T$  may take any real number as value.

The last step is to define the transition probability of moving from candidate  $w$  to a neighbour  $w'$ . If  $w' \succeq w$ , the probability is 1. Otherwise, we repeat the trick: *first* perform the operations proposed by (5.10), and only *afterwards* take the  $q \rightarrow +\infty$  limit:

$$P(w \rightarrow w' \mid T[q]) = \lim_{q \rightarrow +\infty} e^{-\frac{E(w')[q] - E(w)[q]}{T[q]}} \quad (5.14)$$

Observe that if  $C_k$  is the fatal constraint when comparing  $w$  and  $w'$ , then the dominant summand in the expression  $E(w')[q] - E(w)[q]$  is  $[C_k(w') - C_k(w)]q^k$ . Thus, (5.14) and (5.13) yield the following

RULES OF MOVING from  $w$  to  $w'$  at temperature  $T = \langle K_T, t \rangle$ :

- If  $w'$  is better than  $w$ : move!  $P(w \rightarrow w' \mid T) = 1$
- If  $w'$  loses due to fatal constraint  $C_k$ :
  - If  $k > K_T$ : don't move!  $P(w \rightarrow w' \mid T) = 0$
  - If  $k < K_T$ : move!  $P(w \rightarrow w' \mid T) = 1$
  - If  $k = K_T$ : move with probability  $P = e^{-(C_k(w') - C_k(w))/t}$ .

Note that the last expression requires  $t > 0$ , as in thermodynamics. Gradually dropping  $T$  can be done by diminishing  $K_T$  in a loop with an embedded loop that reduces  $t$ . Thus, the height of the allowed counter-optimal jumps also diminish—similarly to usual simulated annealing.

## 5.5 Violation profiles as ordinal numbers

Instead of considering the limit  $q \rightarrow +\infty$  of real-valued weights in polynomials, why not take *infinite weights*? In set theory, the well ordered set  $\{0, 1, 2, \dots, q - 1\}$  defines the integer  $q$ . When the possible levels of violation formed this set, we could use weight  $q$ . In the general case, the possible levels of violation of the constraints form the set  $\{0, 1, 2, \dots\}$ : this well ordered set is called  $\omega$ , the first limit ordinal (Suppes, 1972).

Arithmetic can be defined on ordinal numbers, including comparison, addition and multiplication. These latter operations are associative, but not commutative. Therefore, we can introduce a new representation of the Harmony function  $H(w)$ :

$$E(w) = \omega^N C_N(w) + \dots + \omega C_1(w) + C_0(w) = \sum_{i=N}^0 \omega^i C_i(w) \quad (5.15)$$

Because  $\omega$  is the upper limit of the natural numbers,  $\omega^i n < \omega^{i+1}$  for any finite  $n$ . Thus, the definition of  $E(w)$  in (5.15) with the usual relation  $<$  from ordinal arithmetic also *realises* the Harmony function:  $E(w_1) \leq E(w_2)$  if and only if  $H(w_1) \succeq H(w_2)$ . The very definition of limit ordinals excludes ganging up effects.

We need now the difference of two  $E$  values. Instead of subtraction, one can define an operation  $\Delta(a, b)$  on the violation profile-like ordinal numbers with the form  $\sum_{i=N}^0 \omega^i a_i$ , such that  $a = b + \Delta(a, b)$ :

**Definition 7** If  $a = \sum_{i=N}^0 \omega^i a_i$  and  $b = \sum_{i=N}^0 \omega^i b_i$  and  $a > b$ , let be  $\Delta(a, b) = \sum_{i=N}^0 \omega^i \delta_i$ , where  $\delta_i = \begin{cases} a_i - b_i & \text{if } \forall j. (i < j \leq N): a_j = b_j \\ a_i & \text{otherwise} \end{cases}$

The co-efficient of the highest non-zero term in  $\Delta(E(w'), E(w))$  is the difference of the violation levels of the fatal constraint. The lower summands vanish compared to the highest term, so we can neglect them in a new definition of two violation profile-like ordinal numbers:

**Definition 8** If  $a = \sum_{i=N}^0 \omega^i a_i$  and  $b = \sum_{i=N}^0 \omega^i b_i$ , and  $a > b$ , let be  $\Delta'(a, b) = \sum_{i=N}^0 \omega^i \delta'_i$ , where  $\delta'_i = \begin{cases} a_i - b_i & \text{if } \forall j. (i < j \leq N): a_j = b_j \\ 0 & \text{otherwise} \end{cases}$

Observe that for candidates  $w$  and  $w'$ , if  $C_k$  is the fatal constraint why  $w \succ w'$ , then  $\Delta'(E(w'), E(w)) = \omega^k [C_k(w') - C_k(w)]$ .

Next, we introduce the following conventions, where  $a, b, i$  and  $j$  are positive integers, and  $x, y$  and  $z$  are ordinal numbers (remember that  $\omega$  means “infinity”):

$$e^{-\frac{\omega^i a}{\omega^j b}} := e^{-\omega^{i-j} \frac{a}{b}} := \begin{cases} 1 & \text{if } i < j \\ e^{-\frac{a}{b}} & \text{if } i = j \\ 0 & \text{if } i > j \end{cases} \quad (5.16)$$

Temperature has to have the same form as the difference of two violation profiles  $\Delta'(E(w'), E(w))$ , so we propose

$$T = \langle K_T, t \rangle = \omega^{K_T} t \quad (5.17)$$

```

ALGORITHM: Simulated Annealing for Optimality Theory (SA-OT)
Parameters: w_init, K_max, K_min, K_step, t_max, t_min, t_step
w <-- w_init
  for K = K_max to K_min step K_step
    for t = t_max to t_min step t_step
      choose random w' in neighbourhood(w)
      w <-- w' with probability P( w-->w' | T=<K,t>)
      as defined in the ‘Rules of moving’
    end-for
  end-for
return w

```

FIGURE 1: The algorithm of *Simulated Annealing Optimality Theory*.

Now, all tools are ready to define probability  $P(w \rightarrow w' | T)$ , closely following Eq. (5.10). If  $E(w) \geq E(w')$  then  $P(w \rightarrow w' | T) = 1$ , else

$$P(w \rightarrow w' | T) = e^{-\frac{\Delta'(E(w'), E(w))}{T}} \quad (5.18)$$

Some readers may prefer the way leading to Eq. (5.14), while others the one to (5.18). Yet, the interpretation of both of them yields the same *Rules of moving*, those in section 5.4. Both trains of thought introduce temperature as a pair  $T = \langle K_T, t \rangle$ . Diminishing it requires a double loop: the inner one reduces  $t$ , and the outer one  $K_T$ .

## 5.6 Conclusion: SA-OT

The pseudo-code of the *Optimality Theory Simulated Annealing* algorithm (OT-SA) can be finally presented (figure 1).

Out of the parameters of the algorithm,  $K_{max}$  is usually higher than the index of the highest ranked constraint, in order to introduce an initial phase when the random walker may rove unhindered in the search space. Similarly,  $K_{min}$  defines the length of the final phase of the simulation, giving enough time to “relax”, to reach the closest local optimum. Otherwise, SA-OT would return any candidate, not only local optima, resulting in an uninteresting model. Typically,  $K_{step} = 1$ .

Parameters  $t_{max}$ ,  $t_{min}$  and  $t_{step}$  drive  $t$  in the inner loop, influencing only the exponential appearing in the last case ( $k = K_T$ ) of the *Rules of moving*. As candidates  $w$  and  $w'$  differ only minimally—in a *basic operation*—, their violation profiles are also similar: usually  $|C_k(w') - C_k(w)| \leq 2$ , motivating  $t_{max} = 3$  and  $t_{min} = 0$ . Parameter  $t_{step}$  is the most interesting one, and can vary along more orders of magnitude: by being inversely proportional to the number of iterations performed, it directly controls the speed of the simulation, that is, its precision.

In practice, the algorithm is surprisingly successful in modelling, besides other, fast speech phenomena in Dutch metrical stress assignment (Bíró, 2004). In SA-OT, the frequency of the different forms can be fine-tuned by varying the parameters (especially  $t_{step}$ ). It can also predict different frequencies for the same phenomenon in different inputs (Bíró, forthcoming), if the search space has a different structure: indeed, the *topology* of the search space is an important novel concept in SA-OT.

To sum up, *Simulated Annealing for Optimality Theory* (SA-OT) is a promising algorithm to find the optimal element of the candidate set. In the present paper, we have argued that it is both cognitively plausible and mathematically well-founded, whereas further work has shown that it can account for real phenomena.

### 5.7 Appendix: Energy-polynomials realise $H(w)$

Here, we sketch how to prove that energy-polynomials—with Definition 6 of  $\prec$  in section 5.4—realise the Harmony function:  $E(w_1) \preceq E(w_2)$  if and only if  $H(w_1) \succeq H(w_2)$ . For a more detailed proof, see Bíró (forthcoming). First, we have to demonstrate:

**Theorem 7** LAW OF TRICHOTOMY FOR ENERGY POLYNOMIALS: *for any  $w_1$  and  $w_2 \in GEN(UR)$ , exactly one of the following statements holds: either  $E(w_1) \prec E(w_2)$ , or  $E(w_1) \succ E(w_2)$ , or  $E(w_1) = E(w_2)$ .*

For a proof, note that the polynomial  $P[q] = E(w_1)[q] - E(w_2)[q]$  may have maximally  $N$  roots, the greatest of which be  $q_N$ . Unless  $E(w_1)[q] = E(w_2)[q]$  for all  $q$ 's,  $P[q]$  is either constantly positive or constantly negative for  $q > q_N$ . Subsequently, we need:

**Lemma 8** *If  $H(w_1) \succ H(w_2)$ , then  $E(w_1) \prec E(w_2)$ .*

**Proof** Let  $C_k$  be the fatal constraint due to which  $H(w_1) \succ H(w_2)$ . If  $k = 0$  then  $E(w_2)[q] - E(w_1)[q] = C_0(w_2) - C_0(w_1) > 0$  for all  $q$ . By definition, then,  $E(w_1) \prec E(w_2)$ .

If, however,  $k > 0$ , then let  $c$  be such that  $c > C_i(w_1)$  and  $c > C_i(w_2)$  for all  $i < k$ . Further, let  $q_0 = \max(\frac{2c}{C_k(w_2) - C_k(w_1)}, 2)$ . For all  $q > q_0$ :

$$\begin{aligned} E(w_2)[q] - E(w_1)[q] &= \sum_{i=0}^N [C_i(w_2) - C_i(w_1)]q^i = \\ &= [C_k(w_2) - C_k(w_1)]q^k + \sum_{i=0}^{k-1} [C_i(w_2) - C_i(w_1)]q^i \end{aligned} \quad (5.19)$$

because  $C_k$  is the fatal constraint. As  $q > q_0 \geq \frac{2c}{C_k(w_2) - C_k(w_1)}$ , in the first summand we use  $C_k(w_2) - C_k(w_1) > 2c/q$ . For the second

component, we employ the fact that  $C_i(w_2) - C_i(w_1) > -c$  for all  $i < k$ , as well as the sum of a geometrical series. Consequently,

$$E(w_2)[q] - E(w_1)[q] > \frac{2c}{q}q^k - c\frac{q^k - 1}{q - 1} = c\frac{q^k - 2q^{k-1} + 1}{q - 1} > 0 \quad (5.20)$$

because  $q > q_0 \geq 2$ . In sum, either  $k = 0$  or  $k > 0$ , we have shown that there exists a  $q_0$  such that for all  $q > q_0$ :  $E(w_2)[q] - E(w_1)[q] > 0$ . Because this difference is a polynomial, we obtain one of the two cases required by Definition 6 of  $E(w_1) \prec E(w_2)$ .  $\square$

Finally, the following four statements can be simply demonstrated by using the definitions, the previous lemma and the laws of trichotomy:

**Theorem 9** *Energy-polynomials realise the Harmony function:*

- $E(w_1) = E(w_2)$ , if and only if  $H(w_1) = H(w_2)$ ;
- $E(w_1) \prec E(w_2)$ , if and only if  $H(w_1) \succ H(w_2)$ .

## References

- Biró, Tamás. 2003. Quadratic alignment constraints and Finite State Optimality Theory. In *Proc. FSMNLP within EACL 2003*, pages 119–126. Budapest; also ROA-600<sup>2</sup>.
- Biró, Tamás. 2004. When the hothead speaks: Simulated Annealing Optimality Theory for Dutch fast speech. presented at CLIN 2004, Leiden.
- Biró, Tamás. forthcoming. *Implementations of and Machine Learning in Optimality Theory*. Ph.D. thesis, University of Groningen.
- Eisner, Jason. 2000. Easy and hard constraint ranking in Optimality Theory: Algorithms and complexity. In J. E. et al., ed., *Finite-State Phonology: Proc. SIGPHON-5*, pages 57–67. Luxembourg.
- Ellison, T. Mark. 1994. Phonological derivation in Optimality Theory. In *COLING-94, Kyoto*, pages 1007–1013. also: ROA-75.
- Frank, Robert and Giorgio Satta. 1998. Optimality Theory and the generative complexity of constraint violability. *Comp. Ling.* 24(2):307–315.
- Gerdemann, Dale and Gertjan van Noord. 2000. Approximation and exactness in Finite State Optimality Theory. In *Jason Eisner, Lauri Karttunen, Alain Thériault (eds): SIGPHON 2000, Finite State Phonology*.
- Howells, T. 1988. VITAL: a connectionist parser. In *Proc. 10th Annual Meeting of the Cognitive Science Society*, pages 18–25. Lawrence Erlbaum.

---

<sup>2</sup>ROA stands for Rutgers Optimality Archive at <http://roa.rutgers.edu>

- Jäger, Gerhard. 2003. Maximum entropy models and Stochastic Optimality Theory. m.s., ROA-625.
- Karttunen, Lauri. 1998. The proper treatment of Optimality Theory in computational phonology. In *Proc. FSMNLP*, pages 1–12. Ankara.
- Kempen, Gerard and Theo Vosse. 1989. Incremental syntactic tree formation in human sentence processing: a cognitive architecture based on activation decay and simulated annealing. *Connection Science* 1:273–290.
- Kirkpatrick, S., C. D. Gelatt Jr., and M. P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* 220(4598):671–680.
- Kuhn, Jonas. 2000. Processing Optimality-theoretic syntax by interleaved chart parsing and generation. In *Proc. ACL-38*, pages 360–367. Hongkong.
- Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. Equation of state calculation by fast computing machines. *Journal of Chemical Physics* 21(6):1087–1092.
- Prince, Alan and Paul Smolensky. 2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Malden, MA, etc.: Blackwell.
- Reeves, Colin R., ed. 1995. *Modern Heuristic Techniques for Combinatorial Problems*. London, etc.: McGraw-Hill.
- Selman, Bart and Graeme Hirst. 1985. A rule-based connectionist parsing system. In *Proc. of the Seventh Annual Meeting of the Cognitive Science Society, Irvine*, pages 212–221. Hillsdale, NJ: Lawrence Erlbaum.
- Selman, Bart and Graeme Hirst. 1994. Parsing as an energy minimization problem. In G. Adriaens and U. Hahn, eds., *Parallel Natural Language Processing*, pages 238–254. Norwood, NJ: Ablex Publishing.
- Smolensky, Paul. 1986. Information processing in dynamical systems: Foundations of Harmony Theory. In *Rumelhart et al.: Parallel Distributed Processing*, vol. 1, pages 194–281. Cambridge, MA–London: MIT Press.
- Suppes, Patrick. 1972. *Axiomatic Set Theory*. New York: Dover.
- Tesar, Bruce and Paul Smolensky. 2000. *Learnability in Optimality Theory*. Cambridge, MA - London, England: The MIT Press.
- Turkel, William. 1994. The acquisition of Optimality Theoretic systems. m.s., ROA-11.
- Černý, V. 1985. Thermodynamical approach to the Travelling Salesman Problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications* 45:41–55.



---

# Finite Presentations of Pregroups and the Identity Problem

ALEXA H. MATER AND JAMES D. FIX

## Abstract

We consider finitely generated pregroups, and describe how an appropriately defined rewrite relation over words from a generating alphabet yields a natural partial order for a pregroup structure. We investigate the *identity problem* for pregroups; that is, the algorithmic determination of whether a word rewrites to the identity element. This problem is undecidable in general, however, we give a dynamic programming algorithm and an algorithm of Oerhle (2004) for free pregroups, and extend them to handle more general pregroup structures suggested in Lambek (1999). Finally, we show that the identity problem for a certain class of non-free pregroups is NP-complete.

**Keywords** PREGROUPS, FREE PREGROUPS, WORD PROBLEM, DYNAMIC PROGRAMMING, NP-COMPLETE

Lambek (1999) introduced Compact Bilinear Logic (CBL) to provide a computational method for deciding whether an utterance in a natural language is grammatical. Using CBL, utterances are modeled as elements of mathematical structures called *pregroups*. The structure of a pregroup is based on a partial order over its elements and a set of rules for how pregroup element multiplication relates to the partial order.

## 6.1 Basic definitions

**Definition 9** [Protogroup] A *protogroup*  $\mathcal{P}$  is a quintuple  $(\mathcal{A}, \cdot, \leq, \ell, r)$  consisting of a set of elements, a binary operation, and a binary relation, and two unary operations respectively, which satisfy the following:

*FG-MoL 2005.*  
James Rogers (ed.).  
Copyright © 2009, CSLI Publications.

1.  $\cdot$  is associative.
2.  $\leq$  is a partial order; i.e. it is reflexive, transitive and anti-symmetric.
3. There is a  $1 \in \mathcal{A}$  where  $1 \cdot a = a = a \cdot 1$  for every  $a \in \mathcal{A}$ .
4. For every  $a \in \mathcal{A}$ ,  $a^\ell \cdot a \leq 1$  and  $a \cdot a^r \leq 1$ .
5. For any  $a, b, x, y \in \mathcal{A}$ , if  $a \leq b$ , then  $x \cdot a \cdot y \leq x \cdot b \cdot y$ .

The elements  $a^\ell$  and  $a^r$  are the left and right inverses of  $a$ , respectively. With these, it is natural to assume that

$$1^r = 1 = 1^\ell, a^{\ell r} = a = a^{r \ell}, (a \cdot b)^r = b^r \cdot a^r, (a \cdot b)^\ell = b^\ell \cdot a^\ell$$

for any  $a, b \in \mathcal{A}$ . An  $a \in \mathcal{A}$  may have an infinite sequence of inverses:

$$\dots, a^{\ell \ell}, a^\ell, a, a^r, a^{r r}, \dots$$

We will follow Lambek in denoting  $a$  by  $a^{(0)}$ , the  $n$ -th left inverse of  $a$  by  $a^{(-n)}$  and the  $n$ -th right inverse of  $a$  by  $a^{(n)}$ , so that the above sequence can be written

$$\dots, a^{(-2)}, a^{(-1)}, a^{(0)}, a^{(1)}, a^{(2)} \dots$$

**Definition 10** [Pregroup] A *pregroup*  $\mathcal{P} = (\mathcal{A}, \cdot, \leq, \ell, r)$  is a pro-togroup satisfying the additional property:

6. For every  $a, b \in \mathcal{A}$ , if  $a \leq b$  then  $a^{(2i)} \leq b^{(2i)}$  and  $b^{(2i+1)} \leq a^{(2i+1)}$ .

It will be useful for our purposes to consider an equivalent pregroup property, as noted by Buszkowski (2002):

- 6'. For all  $a \in \mathcal{A}$ ,  $1 \leq a^{(i+1)} a^{(i)}$ .

## 6.2 Applying CBL to Natural Languages

Lambek's idea was to encode a natural language as a pregroup  $\mathcal{P}$  by associating with each word in the language an element of  $\mathcal{P}$ . An utterance is encoded as a product  $b = a_1 a_2 \dots a_n$  of pregroup elements  $a_i$ , one for each word in the utterance. The goal is to have  $b \leq S$ , for some distinguished element  $S$ , whenever the sentence encoded as  $b$  is grammatical. For example, from Lambek (1999) we might assign elements of a pregroup to the sentence "I have been seen" as follows:

$$\begin{array}{cccc} \text{I} & \text{have} & \text{been} & \text{seen} \\ \pi & \pi^r S p^\ell & p o^{\ell \ell} p^\ell & p o^\ell \end{array}$$

where the symbols  $\pi, S, p, o$  are elements of a pregroup. This sequence reduces to  $S$  as demonstrated by the following:

$$\begin{aligned} \pi(\pi^r S p^\ell)(p o^{\ell\ell} p^\ell)(p o^\ell) &= (\pi\pi^r)S(p^\ell p)(o^{\ell\ell}(p^\ell p)o^\ell) \\ &\leq (\pi\pi^r)S(p^\ell p)(o^{\ell\ell}1o^\ell) \\ &= (\pi\pi^r)S(p^\ell p)(o^{\ell\ell}o^\ell) \\ &\leq S \end{aligned}$$

Thus, “I have been seen” is a grammatical utterance with this pregroup assignment.

As a result, we are interested in the algorithmic question of determining whether  $x \leq y$  in a pregroup, where  $x$  and  $y$  are given as a product of pregroup elements. This is the natural *word problem* for pregroups. For *free* pregroups (defined formally below) where cancellations alone yield the pregroup ordering, Oerhle (2004) presented an efficient algorithm for the important case when  $y$  is given as a single element, rather than as a product. We will review a version of this algorithm shortly.

The treatment of pregroups in Lambek (1999) suggests that algorithms for the word problem need to consider additional order relations among elements. For example, Lambek suggests first, second, and third person singular pronouns could be assigned the types  $\pi_1$ ,  $\pi_2$ , and  $\pi_3$ , respectively, and the additional order relations  $\pi_i \leq \pi$  for a type  $\pi$  of all singular pronouns could be assumed part of the pregroup structure.

Considering this further, the pregroup order could include a relation like  $bc \leq a$ . Such additional constraints could give a pregroup a context-free rewrite structure, the reverse of Chomsky Normal Form productions  $A \rightarrow BC$ .

To handle the additional structure that may come with a pregroup, care must be taken in specifying the form in which it is presented as input to an algorithm. To do this, we adapt the notion from combinatorial group theory (see Cohen (1989), Magnus et al. (1966)) of a finite group presentation to pregroups and protogroups. This is a syntactic treatment of pregroups, one where strings of symbols from an alphabet form the pregroup elements, and where the ordering structure comes from rewrite steps, possibly augmented with additional order relations between words.

### 6.3 Pregroup Presentations

For any set  $\mathcal{G}$ , let  $\mathcal{G}' = \{a^{(i)} \mid a \in \mathcal{G}, i \in \mathcal{Z}\}$  be the set of pregroup *letters* over  $\mathcal{G}$ . A pregroup *word*  $W$  over  $\mathcal{G}$  is a sequence of letters, that is,  $W = x_1 x_2 \dots x_n$  where each  $x_k \in \mathcal{G}'$ . We include the empty letter sequence, denoted by  $\varepsilon$ , in the set of pregroup words. Let  $\mathcal{G}^*$

denote the set of all pregroup words over  $\mathcal{G}$ . The elements of  $\mathcal{G}$  are the *generators* of  $\mathcal{G}^*$ .

Define the *left inverse function*  $L : \mathcal{G}^* \rightarrow \mathcal{G}^*$  as follows:

$$L(W) = \begin{cases} \varepsilon & \text{if } W = \varepsilon \\ a^{(i-1)} & \text{if } W = a^{(i)} \text{ for } a \in \mathcal{G} \\ L(x_n) \dots L(x_2)L(x_1) & \text{if } W = x_1 \dots x_n \text{ and each } x_k \in \mathcal{G}'. \end{cases}$$

The *right inverse function*  $R : \mathcal{G}^* \rightarrow \mathcal{G}^*$  is defined similarly.

Any binary relation  $\mathcal{R}$  on  $\mathcal{G}^*$  induces a rewrite relation on  $\mathcal{G}^*$  defined by:

1.  $W \xrightarrow{\mathcal{R}} W$ ,
2.  $W_1 \xrightarrow{\mathcal{R}} W_3$  whenever  $W_1 \xrightarrow{\mathcal{R}} W_2$  and  $W_2 \xrightarrow{\mathcal{R}} W_3$ ,
3.  $W_1 W_2 \xrightarrow{\mathcal{R}} W_1 W' W_2$  whenever  $W \xrightarrow{\mathcal{R}} W'$ ,
4.  $\lambda \xrightarrow{\mathcal{R}} \rho$  for  $(\lambda, \rho) \in \mathcal{R}$ , and
5.  $a^{(i)} a^{(i+1)} \xrightarrow{\mathcal{R}} \varepsilon$  for  $a \in \mathcal{G}$ .

A rewrite step employing rule (5) (in conjunction with rule (3)) is called a *contraction*.

If we identify two words  $W_1$  and  $W_2$ , that is, say that  $W_1 = W_2$ , whenever both  $W_1 \xrightarrow{\mathcal{R}} W_2$  and  $W_2 \xrightarrow{\mathcal{R}} W_1$  hold, we have the following proposition:

**Proposition 10** *The set of all words over  $\mathcal{G}$  forms a protogroup under the binary operation of juxtaposition with identity element  $\varepsilon$ , the unary operations  $L$  and  $R$ , and the binary predicate  $\xrightarrow{\mathcal{R}}$ .*

Thus, it is natural to define the following.

**Definition 11** [Pregroup Presentation] If  $\mathcal{P}$  is a protogroup given by  $\mathcal{G}$  and  $\mathcal{R}$  according to the previous proposition then  $\mathcal{G}$  and  $\mathcal{R}$  form a *pregroup presentation* of  $\mathcal{P}$ .

The relations  $\mathcal{R}$  can encode the additional structure among words. For example, a presentation of Lambek's natural language pregroup in the discussion above would include generators  $\pi, \pi_1, \pi_2, \pi_3$  and the relations  $(\pi_1, \pi), (\pi_2, \pi), (\pi_3, \pi)$ .

**Definition 12** [Pregroup Presentation] Extend  $\xrightarrow{\mathcal{R}}$  to include the property

6.  $\varepsilon \xrightarrow{\mathcal{R}} a^{(i+1)} a^{(i)}$  for  $a \in \mathcal{G}$ .

A rewrite step employing this rule is called an *expansion*. Using this, we have a proposition for pregroups analogous to Proposition 10. We

also obtain an analogous notion of a *pregroup presentation* which we will denote by  $\mathcal{P} = \langle \mathcal{G}, \mathcal{R} \rangle_{\mathcal{P}}$ .

In what follows, we assume that an algorithm's input includes a finite-sized presentation of the structure being considered. We will be less careful about describing presentations syntactically, and return to using  $\leq$  instead of  $\xrightarrow{\mathcal{R}}$ ,  $\ell$  and  $r$  instead of  $L$  and  $R$ , and  $1$  instead of  $\varepsilon$ .

## 6.4 The Identity Problem

From a linguistic perspective, we are primarily interested in determining, whether  $W \leq S$  where  $S$  is some distinguished element in the generating set for  $\mathcal{P}$ . Note that  $W \leq S$  if  $W = W_1TW_2$ ,  $W_1 \leq 1$ ,  $W_2 \leq 1$ , and  $T \leq S$ . Using this fact for the pregroups we consider, leads us to consider the following problem:

### The Pregroup Identity Problem

**Given:**  $\mathcal{P} = \langle \mathcal{G}, \mathcal{R} \rangle_{\mathcal{P}}$ ,  $W \in \mathcal{G}^*$

**Determine:** whether  $W \leq 1$  in  $\mathcal{P}$ .

We say that  $W$  is *nullable* in  $\mathcal{P}$  whenever  $W \leq 1$ .

In general, the identity problem is undecidable for pregroups. This can be proven by showing that any finitely presented group can be encoded as a finitely presented pregroup. Since the identity problem is known to be undecidable for finite presentations of groups, this implies that the identity problem is undecidable for pregroups in general. The identity problem is decidable, however, for certain restricted classes of pregroups. Below we describe algorithms for these classes.

### 6.4.1 An Algorithm for Free Pregroups

We say that a pregroup is *free* if its presentation has  $\mathcal{R} = \emptyset$ . The following is a consequence of a corollary of Lambek (1999), page 21:

**Proposition 11** *Let  $W$  be a nullable word in pregroup  $\mathcal{P} = \langle \mathcal{G}, \emptyset \rangle_{\mathcal{P}}$ . Then  $W \leq 1$  by a series of contractions.*

As an immediate consequence, the identity problem for free protogroups and pregroups are equivalent. In addition, a simple dynamic programming algorithm, similar to the CYK algorithm (Younger, 1967, Kasami, 1965) for parsing strings in a context-free grammar, can be employed to determine whether  $W$  is nullable. Let  $\nu$  be a boolean predicate over words  $\mathcal{G}^*$  that holds exactly when a word is nullable. Let  $W = x_1 \dots x_n$

/	1	2	3	4	5	6	7	8
1	×	1	×	0	×	1	×	1
2	×	×	1	×	1	×	0	×
3	×	×	×	0	×	0	×	0
4	×	×	×	×	1	×	0	×
5	×	×	×	×	×	0	×	0
6	×	×	×	×	×	×	0	×
7	×	×	×	×	×	×	×	1
8	×	×	×	×	×	×	×	×

FIGURE 1: The table  $T$  corresponding to the word  $a^\ell aa^r a^\ell aaaa^r$ .

for  $x_k \in \mathcal{G}'$ . The following recurrence holds:

$$\nu(x_i \dots x_j) = 1 \text{ whenever } \begin{cases} x_i \in \text{lefts}(x_j) \text{ and } j = i + 1 \\ x_i \in \text{lefts}(x_j) \text{ and } \nu(x_{i+1} \dots x_{j-1}) = 1 \\ \nu(x_i \dots x_k) = 1 \text{ and } \nu(x_{k+1} \dots x_j) = 1 \\ \text{for some } k, \end{cases}$$

$$\nu(x_i \dots x_j) = 0 \text{ otherwise.}$$

In the above,  $\text{lefts}(a^{(i)})$  is just the singleton set  $\{a^{(i-1)}\}$ .

Briefly, to determine  $\nu(x_1 \dots x_n)$  the dynamic programming algorithm simply constructs a table  $T$  with entries

$$T[i, j] = \nu(x_i \dots x_j)$$

Entries  $T[i, j]$  where  $j - i = 1$  are determined first, followed by those where  $j - i = 3$ , then  $j - i = 5$ , and so on, ending with the determination of  $T[1, n]$ . The table's entries can be determined in  $O(n^3)$  time assuming that membership in  $\text{lefts}(x_j)$  can be determined in constant time. Figure 1 gives the table for the word  $a^\ell aa^r a^\ell aaaa^r$ .

#### 6.4.2 Allowing ambiguity: Oehrle's algorithm

In linguistic applications of protogroups it is common to assume that a lexical element may have more than one word associated with it. Oehrle (2004) gives a clever extension of the above algorithm, as a graph rewrite algorithm, that efficiently handles word assignment ambiguity. We give a brief review and reformulation of it here.

In Oehrle's algorithm, each lexical element  $e_i$  of a candidate sentence  $e_1 \dots e_m$  is assigned a directed acyclic graph fragment  $F_i$ . If  $e_i$  can be assigned the pregroup word  $W = x_1 \dots x_n$ , then the graph fragment  $F_i$  has a subfragment with  $n$  vertices  $v_1, \dots, v_n$  each labelled with letters  $x_1$  through  $x_n$ . A directed edge connects  $v_j$  with  $v_{j+1}$  in this subfragment.  $F_i$  is just a union of the subfragments of all words  $W$  that can

be assigned to  $e_i$  (these are assumed to form a finite set).

Let  $s(F_i)$  be the set of vertices in  $F_i$  with no predecessors (incoming edges) and let  $t(F_i)$  be those with no successors (outgoing edges). A graph  $G$  is constructed from the candidate sentence by adding edges from all vertices of  $t(F_i)$  to all vertices of  $s(F_{i+1})$ , for all  $i$ . In addition, a distinguished vertex  $s$  is added with edges to  $s(F_1)$  and a distinguished vertex  $t$  is added with edges from  $t(F_m)$ .

The following graph modification algorithm is then applied to  $G$

```

Q := the edges of G
while Q is not empty do
  remove an edge (u, v) from Q
  if label(u) ∈ lefts(label(v)) then
    for each predecessor u' of u, successor v' of v do
      add edge (u', v') to G and Q
    
```

The candidate sentence is nullable if  $s$  and  $t$  are connected by an edge in the resulting graph.

### 6.4.3 Non-free Pregroups with Promotions

As we noted above, allowing arbitrary relations in  $\mathcal{R}$  makes the identity problem undecidable. We consider two restrictions on  $\mathcal{R}$ :

**Generator Promotions** : All relations in  $\mathcal{R}$  are of the form  $a \leq b$  where  $a, b \in \mathcal{G}$ , that is,  $\mathcal{R} \subset \mathcal{G} \times \mathcal{G}$ .<sup>1</sup> In this case, we can extend our rewrite system to include

$$\begin{array}{ccc} a^{(2k)} & \xrightarrow{\mathcal{R}} & b^{(2k)} \\ b^{(2k+1)} & \xrightarrow{\mathcal{R}} & a^{(2k+1)} \end{array}$$

for all  $(a, b) \in \mathcal{R}$  and  $k \in \mathcal{Z}$ .

**Letter Promotions** : All relations in  $\mathcal{R}$  are of the form  $a^{(i)} \leq b^{(j)}$  where  $a, b \in \mathcal{G}$ , that is,  $\mathcal{R} \subset \mathcal{G}' \times \mathcal{G}'$ . In this case, we can extend our rewrite system to include

$$\begin{array}{ccc} a^{(i+2k)} & \xrightarrow{\mathcal{R}} & b^{(j+2k)} \\ b^{(j+2k+1)} & \xrightarrow{\mathcal{R}} & a^{(i+2k+1)} \end{array}$$

for all  $(a^{(i)}, b^{(j)}) \in \mathcal{R}$  and  $k \in \mathcal{Z}$ .

Employing one of these rules in a rewrite step is called a *promotion* (note that Lambek calls Generator Promotions *induced steps*). The following generalization of Corollary 1 of Lambek (1999), page 21 holds for these presentations:

---

<sup>1</sup>Note that such presentations are called free pregroups by Lambek (1999).

**Theorem 12** *Let  $W$  be a nullable word in a pregroup  $\mathcal{P} = \langle \mathcal{G}, \mathcal{R} \rangle$  where  $\mathcal{R} \subset \mathcal{G}' \times \mathcal{G}'$ . Then  $W \leq 1$  by a series of contractions and promotions.*

**Proof** Let  $W \leq UV \leq Ua^{(i+1)}a^{(i)}V \leq 1$  where the last inequality occurs only through contractions and promotions. We have two cases to consider:

**Case 1:**  $U \leq 1$  without expansions,  $a^{(i+1)} \leq b^{(j)}$  by promotions,  $V \leq 1$  without expansions, and  $a^{(i)} \leq b^{(j+1)}$  by promotions. In other words, the expansion introduces two letters that cancel after a series of promotions. In this case,  $UV \leq 1$  without expansions.

**Case 2:**  $U = U_1xU_2$ ,  $x \leq b^{(j)}$  by promotions,  $a^{(i+1)} \leq b^{(j+1)}$  by promotions,  $U_2 \leq 1$  without expansions,  $V = V_1yV_2$ ,  $y \leq c^{(k+1)}$  by promotions,  $a^{(i)} \leq c^{(k)}$  by promotions,  $V_1 \leq 1$  without expansions, and  $U_1V_2 \leq 1$  without expansions. In other words, the first inserted letter cancels with the a letter in the prefix  $U$  and the second inserted letter cancels a letter in the suffix  $V$ . In this case, note that,  $b^{(j)} \leq a^{(i)}$  and  $c^{(k+1)} \leq a^{(i+1)}$  by promotions, and so

$$xy \leq b^{(j)}y \leq a^{(i)}y \leq a^{(i)}c^{(k+1)} \leq c^{(k)}c^{(k+1)} \leq 1$$

by promotions and one contraction. Thus

$$UV = U_1xU_2V_1yV_2 \leq U_1xV_1yV_2 \leq U_1xyV_2 \leq U_1V_2 \leq 1$$

without expansions.

Since, in either case,  $UV \leq 1$ , we can always remove the last expansion to demonstrate the nullability of  $W$ , and the theorem follows.  $\square$

Given a presentation where  $\mathcal{R}$  has only generator promotions, we can compute the transitive closure  $\mathcal{R}^*$  of  $\mathcal{R}$ . Using  $\mathcal{R}^*$  we can employ the algorithms for free pregroups given above using

$$\text{lefts}(a^{(i)}) = \{b^{(i)} \mid (a, b) \in \mathcal{R}^*, b \in \mathcal{G}\}$$

for even  $i$  and

$$\text{lefts}(a^{(i)}) = \{b^{(i)} \mid (b, a) \in \mathcal{R}^*, b \in \mathcal{G}\}$$

for odd  $i$ .

A presentation with letter promotions, however, makes the identity problem NP-hard. Consider the following problem:

**The Pregroup Letter Promotion Problem**

**Given:** pregroup  $\mathcal{P} = \langle \mathcal{G}, \mathcal{R} \rangle_p$  where  $\mathcal{R} \subseteq \mathcal{G}' \times \mathcal{G}'$ ,  $x, y \in \mathcal{G}'$

**Determine:** whether  $x \leq y$  by promotions.

Suppose  $x = a^{(i)}$  and  $y = b^{(j)}$  for  $a, b \in \mathcal{G}$ . Note that  $x \leq y$  by promotions exactly when there exists a generator sequence  $a_0, a_1, \dots, a_n \in \mathcal{G}$



with  $a_0 = a$  and  $a_n = b$ , an integer sequences  $i_1, \dots, i_n$  and  $k_1, \dots, k_n$  where either

$$\left( a_{m-1}^{(2k_m)}, a_m^{(i_m+2k_m)} \right) \in \mathcal{R},$$

or

$$\left( a_m^{(i_m+2k_m+1)}, a_{m-1}^{(2k_m+1)} \right) \in \mathcal{R}$$

for each  $m$  with  $0 < m \leq n$ , and

$$j - i = \sum_{m=1}^n i_m$$

as this allows the series of promotions

$$a^{(i)} = a_0^{(i)} \leq a_1^{(i+i_1)} \leq a_2^{(i+i_1+i_2)} \leq \dots \leq a_n^{(i+i_1+i_2+\dots+i_n)} = b^{(j)}$$

that demonstrate that  $x \leq y$ . The summation requirement is the key to our NP-hardness proof.

**Theorem 13** *The Pregroup Letter Promotion Problem is NP-complete*

**Proof** The following problem is NP-complete (see Garey and Johnson (1979)):

**The Subset Sum Problem**

**Given:** integer subset  $S$ , integer  $s$

**Determine:** the existence of an  $X \subseteq S$  with  $s = \sum_{x \in X} x$ .

An instance of the Subset Sum Problem with set  $S = \{t_1, t_2, \dots, t_n\}$  and target sum  $s$  can be reduced to the Letter Problem instance

$$\begin{aligned} \mathcal{G} &= \{a_0, a_1, \dots, a_n\} \\ \mathcal{R} &= \{(a_{i-1}, a_i) \mid 0 \leq i \leq n\} \cup \{(a_{i-1}, a_i^{(2t_i)}) \mid 0 \leq i \leq n\} \\ x &= a_0 \\ y &= a_n^{(2s)} \end{aligned}$$

It should be clear that  $x \leq y$  by promotions exactly when there exists a subset of  $S$  that sums to  $s$ . □

**6.5 Further Questions**

Though allowing general letter promotions make the word problem for pregroups NP-hard, there may be algorithms that work well in practice. For example, it is likely that any application of pregroups would make use of letter promotions of the form  $a^{(i)} \leq b^{(j)}$ , where  $|j - i| \leq M$ , where  $M$  is sufficiently small. In addition, it seems interesting to investigate what pregroup structures result from other natural classes of presentation relations  $\mathcal{R}$  beyond those that allow just generator and

letter promotions, and from an algorithmic perspective, to determine the complexity of the word problems for these classes.

## References

- Buszkowski, W. 2002. Pregroups: Models and grammars. In H. de Swart, ed., *Relational Methods in Computer Science*, vol. 2561 of *Lecture Notes in Computer Science*, pages 35–49. Berlin: Springer-Verlag.
- Buszkowski, W. 2003. Sequent systems for compact bilinear logic. *Mathematical Logic Quarterly* 49(5):467–474.
- Cohen, D. E. 1989. *Combinatorial Group Theory: a topological approach*, vol. 14 of *London Mathematical Society Student Texts*. Cambridge: Cambridge University Press.
- Garey, M. R. and D.S. Johnson. 1979. *Computers and Intractability*. New York, NY: W.H. Freeman.
- Kasami, T. 1965. An efficient recognition and syntax-analysis algorithm for context-free languages. *Scientific Report AFCRL-65-758* .
- Lambek, J. 1995. Bilinear logic in algebra and linguistics. In J.-Y. Girard, Y. Lafont, and L. Regnier, eds., *Advances in Linear Logic*, London Mathematical Society Lecture Note Series, pages 43–60. Cambridge, UK: Cambridge University Press.
- Lambek, J. 1999. Type grammar revisited. In A. Lecomte, F. Lamarche, and G. Perrier, eds., *Logical Aspects of Computational Linguistics*, vol. 1582 of *Lecture Notes in Computer Science*, pages 1–27. Berlin: Springer-Verlag.
- Magnus, W., A. Karrass, and D. Solitar. 1966. *Combinatorial Group Theory: Presentations of Groups in Terms of Generators and Relations*, vol. XIII of *Pure and Applied Mathematics: A Series of Texts and Monographs*. New York: Interscience Publishers.
- Oerhle, R. 2004. A parsing algorithm for pregroup grammars. In *Categorical Grammars: An Efficient Tool for Natural Language Processing*.
- Younger, D. H. 1967. Recognition and parsing of context-free languages in time  $O(n^3)$ . *Information and Control* 10(2):609–617.

# On rigid NL Lambek grammars inference from generalized functor-argument data

DENIS BÉCHET AND ANNIE FORET

## Abstract

This paper is concerned with the inference of categorial grammars, a context-free grammar formalism in the field of computational linguistics. A recent result has shown that whereas they are not learnable from strings in the model of Gold, rigid and  $k$ -valued non-associative Lambek grammars are still learnable from generalized functor-argument structured sentences. We focus here on the algorithmic part of this result and provide an algorithm that can be seen as an extension of Buszkowski, Penn and Kanazawa's contributions for classical categorial grammars.

**Keywords** GRAMMATICAL INFERENCE, CATEGORIAL GRAMMARS, NON-ASSOCIATIVE LAMBEK CALCULUS, LEARNING FROM POSITIVE EXAMPLES, MODEL OF GOLD

## 7.1 Introduction and background

This paper is concerned with the inference of categorial grammars, a context-free grammar formalism in the field of computational linguistics. We consider learning from positive data in the model of Gold. When the data has no structure, a recent result has shown that rigid and  $k$ -valued non-associative Lambek (*NL*) grammars admit no learning algorithm (in the model of Gold). By contrast, these classes become theoretically learnable from generalized functor-argument structured sentences; the paper (Bechet and Foret (2003)) establishes the existence of such learning algorithms, without explicitly defining one.

*FG-MoL 2005.*  
James Rogers (ed.).  
Copyright © 2009, CSLI Publications.

We focus here on the algorithmic part for *NL* rigid grammars; we explicit an algorithm (for the rigid case) that can be seen as an extension to *NL* of Buszkowski, Penn and Kanazawa’s contributions Buszkowski and Penn (1990), Kanazawa (1998a) for classical (*AB*) categorial grammars. The algorithm introduced here outputs a grammar with variables and constraints, an alternative representation for *NL* rigid grammars, that is defined later in this article; this kind of representation can be seen as the necessary information used by parses to check sentence recognizability Aarts and Trautwein (1995).

**Learning.** The model of Gold (1967) used here consists in defining an algorithm on a finite set of structured sentences that converges to obtain a grammar in the class that generates the examples.

In a grammar system  $\langle \mathbb{G}, \mathbb{S}, \mathbb{L} \rangle$  (that is  $\mathbb{G}$  is a “hypothesis space”,  $\mathbb{S}$  is a “sample space”,  $\mathbb{L}$  is a function from  $\mathbb{G}$  to subsets of  $\mathbb{S}$ ) a function  $\phi$  is said to learn  $\mathbb{G}$  in Gold’s model iff for any  $G \in \mathbb{G}$  and for any enumeration  $\langle e_i \rangle_{i \in \mathbb{N}}$  of  $\mathbb{L}(G)$  there exists  $n_0 \in \mathbb{N}$  and a grammar  $G' \in \mathbb{G}$  such that  $\mathbb{L}(G') = \mathbb{L}(G)$  and  $\forall n \geq n_0, \phi(\langle e_0, \dots, e_n \rangle) = G'$ .

**Categorial Grammars.** The reader not familiar with Lambek Calculus and its non-associative version will find nice presentation in (Lambek (1961), Buszkowski (1997), Moortgat (1997), de Groote and Lamarche (2002)). We use in the paper non-associative Lambek calculus (written *NL*) without empty sequence and without product.

The *types*  $Tp$  are generated from a set of *primitive types*  $Pr$  by two binary connectives “/” (over) and “\” (under):  $Tp ::= Pr \mid Tp \backslash Tp \mid Tp / Tp$ .

A *categorial grammar* is a structure  $G = (\Sigma, I, S)$  where:  $\Sigma$  is a finite alphabet (the words in the sentences);  $I : \Sigma \mapsto \mathcal{P}^f(T)$  is a function that maps a set of types to each element of  $\Sigma$  (the possible categories of each word);  $S \in Pr$  is the *main type* associated to correct sentences. A *k-valued categorial grammar* is a categorial grammar where, for every word  $a \in \Sigma$ ,  $I(a)$  has at most  $k$  elements. A *rigid categorial grammar* is a 1-valued categorial grammar.

A sentence belongs to the *language of*  $G$ , provided its words can be assigned types that derive  $S$  according to the rules of the type calculus.

The case of *AB* categorial grammars is defined by two rules :

$$/_e : A / B, B \Rightarrow A \quad \text{and} \quad \backslash_e : B, B \backslash A \Rightarrow A \quad (AB \text{ rules})$$

In the case of *NL*, the derivation relation  $\vdash_{NL}$  is defined by (left part belongs to  $\mathcal{T}_{Tp}$  the set of binary trees over  $Tp$ ) :

$$\begin{array}{c}
\frac{}{A \vdash A} \mathbf{Ax} \\
\Gamma \vdash A \quad \Delta[A] \vdash B \\
\hline
\Delta[\Gamma] \vdash B
\end{array}
\mathbf{Cut}
\begin{array}{c}
\frac{(\Gamma, B) \vdash A}{\Gamma \vdash A/B} /R \\
\Gamma \vdash A \quad \Delta[B] \vdash C \\
\hline
\Delta[(B/A, \Gamma)] \vdash C
\end{array}
/L
\begin{array}{c}
\frac{(A, \Gamma) \vdash B}{\Gamma \vdash A \setminus B} \setminus R \\
\Gamma \vdash A \quad \Delta[B] \vdash C \\
\hline
\Delta[(\Gamma, A \setminus B)] \vdash C
\end{array}
\setminus L$$

**Learning categorial grammars.** Lexicalized grammars of natural languages such as categorial grammars are adapted to learning perspectives: since the rules are known, the task amounts to determine the type assignment; this usually involves a unification phase.

In fact, the learnable or unlearnable problem for a class of grammars depends both of the information that the input structures carry and the model that defines the language associated to a given grammar. The input information can be just a string, the list of words of the input sentence. It can be a tree that describes the sub-components with or without the indication of the head of each sub-component. More complex input information give natural deduction structure or semantics information.

For  $k$ -valued categorial grammars:  $AB$  grammars are learnable from strings (Kanazawa (1998b)), associative Lambek grammars are learnable from natural deduction structures (elimination and introduction rules) (Bonato and Retoré (2001)) but not from strings (Foret and Le Nir (2002)), NL grammars are not learnable from well-bracketed strings but are from generalized functor argument structures (Bechet and Foret (2003)).

**Organization of the paper.** Section 2 explains the notion of functor-argument and its generalized version in the case of  $NL$  (as recently introduced in (Bechet and Foret (2003))). Section 3 defines the algorithm (named  $RGC$ ) on these structures. Section 4 defines grammars with variables and constraints underlying the  $RGC$  algorithm. Section 5 gives the main properties of the  $RGC$  algorithm. Section 6 concludes.

## 7.2 $FA$ structures in $NL$

### 7.2.1 $FA$ structures

Let  $\Sigma$  be an alphabet, a  $FA$  structure over  $\Sigma$  is a binary tree where each leaf is labelled by an element of  $\Sigma$  and each internal node is labelled by the name of the binary rule.

When needed we shall distinguish two kinds of  $FA$  structures, over the grammar alphabet (the default one) and over the types.

### 7.2.2 GAB Deduction

A *generalized AB deduction*, or GAB deduction, over  $Tp$  is a binary tree using the following conditional rules ( $C \vdash_{NL} B$  must be valid in  $NL$ ):

$$\boxed{\begin{array}{c|c} \frac{A/B \quad C}{A} /_{e^+} & \frac{C \quad B \setminus A}{A} \setminus_{e^+} \\ \hline & \text{condition (both rules)} \\ & C \vdash_{NL} B \text{ valid in } NL \end{array}}$$

In fact, a deduction must be justified, for each node, by a proof of the corresponding sequent in  $NL$ . Thus, a rule has three premises: the two sub-deductions and a  $NL$  derivation (later called a *constraint*).

Notation. For  $\Gamma \in \mathcal{T}_{Tp}$  and  $A \in Tp$ , we write  $\Gamma \vdash_{GAB} A$  when there is a GAB deduction  $\mathcal{P}$  of  $A$  corresponding to the binary tree  $\Gamma$ .

There is a strong correspondence between GAB deductions and  $NL$  derivations. In particular Theorem 14 shows that the respective string languages and binary tree languages are the same.

**Theorem 14 (Bechet and Foret (2003))** *If  $A$  is atomic,  $\Gamma \vdash_{GAB} A$  iff  $\Gamma \vdash_{NL} A$*

### 7.2.3 From GAB deductions to FA structures in NL

A GAB deduction  $\mathcal{P}$  can be seen as an annotated FA structure where the leaves are types and the nodes need a logical justification.

We write  $FA_{Tp}(\mathcal{P})$  for the FA structure over types that corresponds to  $\mathcal{P}$  (internal types and  $NL$  derivations are forgotten). We then define FA structures over  $\Sigma$  instead of over  $Tp$  (these two notions are close).

**Definition**  $G = (\Sigma, I, S)$  generates a FA structure  $F$  over  $\Sigma$  (in the GAB derivation model) iff there exists  $c_1, \dots, c_n \in \Sigma$ ,  $A_1, \dots, A_n \in Tp$  and a GAB derivation  $\mathcal{P}$  such that:

$$\begin{cases} G : c_i \mapsto A_i \ (1 \leq i \leq n) \\ FA_{Tp}(\mathcal{P}) = F[c_1 \rightarrow A_1, \dots, c_n \rightarrow A_n] \end{cases}$$

where  $F[c_1 \rightarrow A_1, \dots, c_n \rightarrow A_n]$  is obtained from  $F$  by replacing respectively the left to right occurrences of  $c_1, \dots, c_n$  by  $A_1, \dots, A_n$ .

The *language of FA structures* corresponding to  $G$ , written  $\mathcal{FL}_{GAB}(G)$ , is the set of FA structures over  $\Sigma$  generated by  $G$ .

**Example 1** Let  $G_1: \{John \mapsto N ; Mary \mapsto N ; likes \mapsto N \setminus (S/N)\}$  we get:  $/_{e^+} \setminus_{e^+} (John, likes), Mary \in \mathcal{FL}_{GAB}(G_1)$

## 7.3 Algorithm on functor-argument data

### 7.3.1 Background -RG algorithm

We first recall Buszkowski's Algorithm called RG as in Kanazawa (1998b), it is defined for  $AB$  grammars, based on  $/_e$  and  $\setminus_e$ .

The RG algorithm takes a set  $D$  of functor-argument structures as positive examples and returns a rigid grammar  $RG(D)$  compatible with the input if there is one (compatible means that  $D$  is in the set of functor-argument structures generated by the grammar).

**Sketch of RG-algorithm, computing  $RG(D)$ :**

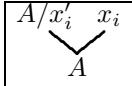
1. assign  $S$  to the root of each structure
2. assign distinct variables to argument nodes
3. compute the other types on functor nodes according to  $/_e$  and  $\backslash_e$
4. collect the types assigned to each symbol, this provides  $GF(D)$
5. unify (classical unification) the types assigned to the same symbol in  $GF(D)$ , and compute the most general unifier  $\sigma_{mgu}$  of this family of types.
6. The algorithm fails if unification fails, otherwise the result is the application of  $\sigma_{mgu}$  to the types of  $GF(D)$  :  $RG(D) = \sigma_{mgu}(GF(D))$ .

### 7.3.2 An algorithm for NL : a proposal

In the context of NL-grammars, we show how rules  $/_{e+}$  and  $\backslash_{e+}$  will play the role of classical forward and backward elimination  $/_e$  and  $\backslash_e$ .

We now give an algorithm that given a set of positive examples  $D$  computes  $RGC(D)$  composed of a general form of grammar together with derivation constraints on its type variables. This formalism is developed in next section. The main differences between  $RG$  and  $RGC$  appear in steps (3) and (6).

**Algorithm for  $RGC(D)$  - rigid grammar with constraints**

1. assign  $S$  to root ;
2. assign distinct variables  $x_i$  to argument nodes ;
3. compute the other types on functor nodes and the derivation constraints according to  $/_{e+}$  and  $\backslash_{e+}$  rules as follows : for each functor node in a structure corresponding to an argument  $x_i$  and a conclusion  $A_i$  assign one of the types (according to the rule)  $A_i/x'_i$  or  $x'_i\backslash A_i$ , where the  $x'_i$  variables are all distinct and add the constraint  $x_i \vdash x'_i$  ; 
4. collect the types assigned to each symbol, this provides  $GF^+(D)$ ; and collect the derivation constraints, this defines  $GC^+(D)$  ; let  $GFC(D) = \langle GF^+(D), GC^+(D) \rangle$
5. unify (classical unification) the types assigned to the same symbol in  $GF^+(D)$ , and compute the most general unifier  $\sigma_{mgu}$  of this family of types.

6. The algorithm fails if unification fails, otherwise the result is the application of  $\sigma_{mgu}$  to the types of  $GF^+(D)$  and to the set of constraints, this defines  $RGC(D) = \langle RG^+(D), RC^+(D) \rangle$  consisting in the rigid grammar  $RG^+(D) = \sigma_{mgu}(GF^+(D))$  and the set of derivation constraints  $RC^+(D) = \sigma_{mgu}(GC^+(D))$ ; the result is later written  $RGC(D) = \sigma_{mgu}(GFC(D))$ .

**Example 2** We consider the following functor-argument structures :

$$\begin{array}{l} \downarrow_{e^+} /_{e^+} (a, man), swims \\ \downarrow_{e^+} /_{e^+} (a, fish), \downarrow_{e^+} (swims, fast) \end{array}$$

The algorithm computes the following grammar with constraints :

	general form	unification	rigid grammar, constraints
<i>a</i>	$X_1 / X'_2, X_3 / X'_4$ $X_2 \vdash X'_2, X_4 \vdash X'_4$	$X_1 = X_3, X'_2 = X'_4$	$X_1 / X'_2$ $X_2 \vdash X'_2, X_4 \vdash X'_2$
<i>fast</i>	$X'_5 \setminus (X'_3 \setminus S)$ $X_3 \vdash X'_3, X_5 \vdash X'_5$		$X'_5 \setminus (X'_3 \setminus S)$ $X_1 \vdash X'_3, (X'_1 \setminus S) \vdash X'_5$
<i>fish</i>	$X_4$		$X_4$
<i>man</i>	$X_2$		$X_2$
<i>swims</i>	$X'_1 \setminus S, X_5$ $X_1 \vdash X'_1$	$X_5 = X'_1 \setminus S$	$X'_1 \setminus S$ $X_1 \vdash X'_1$

## 7.4 Grammars with variables and constraints

Let  $Var$  denote a set of variables, we write  $Tp_{Var}$  the set of types that are constructed from the primitive types  $Pr \cup Var$ .

**Definition.** A grammar with variables and constraints on  $Var$  is a structure  $\langle G, \mathcal{C} \rangle$  where  $G$  is a categorial grammar whose types are in  $Tp_{Var}$  and  $\mathcal{C}$  is a set of sequents  $t_i \vdash t'_i$  with types in  $Tp_{Var}$ .

We then define a structure language  $\mathbb{FL}_{GAB}(\langle G, \mathcal{C} \rangle)$  based on derivations defined similarly to  $GAB$ , by replacing the conditions  $C \vdash B$  valid in  $NL$  with conditions of the form  $C \vdash B \in \mathcal{C}$  (constraints):

**Definition.**  $\mathbb{FL}_{GAB}(\langle G, \mathcal{C} \rangle)$  is the set of functor-argument structures obtained from deductions using the following conditional rules :

$\frac{A/B \quad C}{A} /_{e^+ C}$	$\frac{C \quad B \setminus A}{A} \setminus_{e^+ C}$	condition (both rules) $C \vdash B \in \mathcal{C}$
-----------------------------------	---	--

**Definition.** For any NL-grammar  $G$ ,  $Constraints(G)$  is the set of NL-valid sequents  $t_i \vdash t'_i$  where  $t_i$  is any head subtype<sup>1</sup> of a type assigned by  $G$ , and  $t'_i$  is any argument subtype<sup>2</sup> of any type assigned by  $G$ .

**Property:** the constraints of any rule  $/_{e^+}$  or  $\setminus_{e^+}$  in a  $GAB$  deduction for  $G$  belong to  $Constraints(G)$  (proof by induction on a deduction).

<sup>1</sup> $A$  is head subtype of  $A$ , a head subtype of  $A$  is head subtype of  $A/B$  and  $B \setminus A$

<sup>2</sup> $B$  is an argument subtype of  $A/B$  and of  $B \setminus A$ , any argument subtype of  $A$  is an argument subtype of  $A/B$  and of  $B \setminus A$



**A partial order<sup>3</sup> on grammars with variables and constraints.**

Let us write  $\langle G, C \rangle \subseteq \langle G', C' \rangle$  IFF both  $G \subseteq G'^4$  and  $C \subseteq C'$ .

For a substitution  $\sigma$ , we write  $\sigma \langle G, C \rangle = \langle \sigma(G), \sigma(C) \rangle$  and define:

$$\begin{aligned} \langle G, C \rangle \sqsubseteq \langle G', C' \rangle &\text{ IFF } \exists \sigma : \sigma \langle G, C \rangle \subseteq \langle G', C' \rangle \\ G \sqsubseteq G' &\text{ IFF } \exists \sigma : \sigma(G) \subseteq G' \end{aligned}$$

Next properties follow easily.

**Property:**

$$\langle G, C \rangle \sqsubseteq \langle G', C' \rangle \text{ implies } \mathbb{F}\mathbb{L}_{GAB}(\langle G, C \rangle) \subseteq \mathbb{F}\mathbb{L}_{GAB}(\langle G', C' \rangle)$$

**Property:**  $\mathbb{F}\mathbb{L}_{GAB}(\langle G, \text{Constraints}(G) \rangle) = \mathcal{F}\mathcal{L}_{GAB}(G)$ , for  $G$  on  $Tp$ .

**7.5 Properties of algorithm  $RGC$ .**

Next property is analogue to properties by Buszkowski and Penn for  $RG$ ; the first part concerns the general forms (defined in step (4) of  $RGC$ ), the second concerns the rigid form computed in the final step.

**Property 15** *Let  $D$  be a finite set of structures and  $G$  a rigid grammar:*

*IF  $D \subseteq \mathcal{F}\mathcal{L}_{GAB}(G)$  THEN*

- (1)  $\exists \sigma : \sigma(GFC(D)) \subseteq \langle G, \text{Constraints}(G) \rangle$
- (2)  $RGC(D)$  exists and  $\exists \tau : \tau(RGC(D)) \subseteq \langle G, \text{Constraints}(G) \rangle$
- (3)  $D \subseteq \mathbb{F}\mathbb{L}_{GAB}(GFC(D)) \subseteq \mathbb{F}\mathbb{L}_{GAB}(RGC(D)) \subseteq \mathcal{F}\mathcal{L}_{GAB}(G)$

**Proof of property 15 (1)** : each  $F_i \in D$  corresponds to (at least one)  $GAB$ -derivation in  $G$ , we choose one such  $\mathcal{P}_i$  for each  $F_i$ . Each  $F_i \in D$  also corresponds to an annotated structure computed at step (3) of the  $RGC$  algorithm, that we write  $\mathcal{Q}_i$ . We first focus on these structures for a given  $i$  and define gradually a substitution  $\sigma$ . Each node of  $F_i$  labelled by rule  $/_{e+}$  ( $\setminus_{e+}$  is to be treated similarly) is such that :

- when considered in  $\mathcal{Q}_i$ , it is labelled by a type that we write  $t_j$  with successors labelled by  $t_j/x'_j$  and  $x_j$  (from left to right) for some  $j$ ;
- when considered in  $\mathcal{P}_i$ , it is labelled by a type  $A_j$  with 2 successors labelled by some types  $A_j/B_j$  and  $\Gamma_j$  (from left to right) such that  $\Gamma_j$  is a formula and  $\Gamma_j \vdash B_j$  is valid in  $NL$ .

We then take  $\sigma(x_j) = \Gamma_j$  and  $\sigma(x'_j) = B_j$ . We get more generally :

$$\left\{ \begin{array}{l} \text{(i) if } t \text{ labels a node in } \mathcal{Q}_i \\ \quad \text{then } \sigma(t) \text{ labels the corresponding node in } \mathcal{P}_i \\ \text{(ii) } \sigma(x_j) \vdash \sigma(x'_j) \text{ is valid in NL and belongs to } \text{Constraints}(G) \end{array} \right.$$

where (ii) is clear by construction, and (i) is proved by induction on the length of  $t$  : as a basis, we consider the root node  $S$  that is invariant by  $\sigma$ , and the variable cases with the definition of  $\sigma(x_j)$ , we apply

<sup>3</sup>with equality up to renaming

<sup>4</sup>by  $G \subseteq G'$ , we mean the assignments in  $G$  are also in  $G'$

otherwise the hypothesis for  $t_j$  to  $\sigma(t_j/x'_j) = \sigma(t_j)/\sigma(x'_j)$  or  $\sigma(x'_j \setminus t_j) = \sigma(x'_j) \setminus \sigma(t_j)$ . Properties (i) (ii) give (1) ■

**Proof of property 15 (2):** we take  $\sigma$  as in (1), this shows that unification succeeds for the family of types of  $GF^+(D)$  (it admits a unifier since  $\sigma(GF^+(D)) \subseteq G$ , where  $G$  is rigid); we consider a most general unifier  $\sigma_{mgu}$ , such that  $RGC(D) = \sigma_{mgu}(GFC(D))$ , which provides the existence of  $\tau$  such that  $\sigma = \tau \circ \sigma_{mgu}$ ; rewriting 15(1) gives 15(2) ■

**Proof of property 15 (3):** from  $D \subseteq FL_{GAB}(GFC(D))$  (clear by construction)  $RGC(D) = \sigma_{mgu}(GFC(D))$  and from property 2(2) ■

**Lemma 16 (Incrementality)** *IF  $D \subseteq D' \subseteq \mathcal{FL}_{GAB}(G)$  where  $G$  is rigid*

*THEN  $\exists \eta : \eta(RGC(D)) \subseteq RGC(D')$  (written  $RGC(D) \sqsubseteq RGC(D')$ ).*

**Proof :** suppose  $G$  is rigid with  $D \subseteq D' \subseteq \mathcal{FL}_{GAB}(G)$ ; we already know from Proposition 15(2) that  $RGC(D)$  and  $RGC(D')$  are defined. Let us show  $RGC(D) \sqsubseteq RGC(D')$ ;

let  $\sigma_{mgu}$  and  $\sigma'_{mgu}$  denote the respective most general unifier computed by step 5 of the RGC algorithm for  $D$  and  $D'$ ;

since  $D \subseteq D'$ ,  $\sigma'_{mgu}$  is also a unifier for  $D$  therefore

(i)  $\exists \eta : \sigma'_{mgu} = \eta \circ \sigma_{mgu}$ ;

on the other hand, from  $D$  to  $D'$ , we only add new generalized FA-structures, thus new variables in step 3 of the algorithm, new types and new constraints in step 4, thus:

(ii)  $D \subseteq D'$  implies  $GFC(D) \subseteq GFC(D')$  (modulo variable renaming)

also from the algorithm:

(iii)  $RGC(D) = \sigma_{mgu}(GFC(D))$  and  $RGC(D') = \sigma'_{mgu}(GFC(D'))$

all of which we get  $RGC(D') = \eta \circ \sigma_{mgu}(GFC(D'))$ ;

hence (inclusion modulo variable renaming):

$$\underbrace{\eta \circ \sigma_{mgu}(GFC(D))}_{= \eta(RGC(D))} \subseteq \underbrace{\eta \circ \sigma_{mgu}(GFC(D'))}_{= RGC(D')} \quad \blacksquare$$

**Theorem 17 (Convergence)** *Let  $G$  be a rigid grammar, and  $\langle F_i \rangle_{i \in \mathbb{N}}$  denote any enumeration of  $\mathcal{FL}_{GAB}(G)$ , RGC converges on  $\langle F_i \rangle_{i \in \mathbb{N}}$  to a grammar with variables and constraints having the same language.*

$\exists p_0 \forall p > p_0 : RGC(D_p) = RGC(D_{p_0}), \mathbb{FL}_{GAB}(RGC(D_p)) = \mathcal{FL}_{GAB}(G)$   
where  $D_p = \{F_1, \dots, F_p\}$

**Proof :** given  $G$  rigid  $\{G', \exists \sigma : \sigma(G') \subseteq G\}$  is finite (up to renaming) and from the above lemma:  $RG^+(D_p) \sqsubseteq RG^+(D_{p+1}) \sqsubseteq \dots \sqsubseteq G$ , (using property 15); therefore  $\exists p'_0, \forall p > p'_0 : RG^+(D_{p+1}) = RG^+(D_p)$  (up to renaming). The constraints part of  $\langle G, C \rangle$  computed by the algorithm is such that  $C$  only involves subformulas of the types in grammar  $G$  (this holds for the general form and its substitution instances as well).

Since after  $p_0$ , the grammar part is stationary, and since its set of subformulas is finite, the constraint part must also converge after  $p'_0$  :  $\exists p_0 > p'_0 : \forall p > p_0 : RC^+(D_{p+1}) = RC^+(D_p)$ .

The language equality is then a corollary using property 15(3) ■

## 7.6 Conclusion and remarks

We have proposed a learning algorithm that is unification-based, polynomial according to the size of the input data (unification can be performed in linear time using a suitable data structure), and that can be performed incrementally with new structured sentences.

We recall that a sentence generated by any *NL* grammar can be recognized in polynomial time Aarts and Trautwein (1995). The output of the *RGC* algorithm is a grammar with variables and constraints that also represents the memoized information used to obtain a polynomial parsing algorithm from a grammar without constraint.

Another perspective is the extension of the result and the algorithm exposed here to other variants of categorial grammars; this should apply to NL allowing empty sequents ; another candidate is the associative version, where the forward and backward rules could be still generalized allowing a sequence of types instead of a single type as argument.

## References

- Aarts, E. and K. Trautwein. 1995. Non-associative Lambek categorial grammar in polynomial time. *Mathematical Logic Quarterly* 41:476–484.
- Bechet, Denis and Annie Foret. 2003. k-valued non-associative Lambek grammars are learnable from function-argument structures. In *Proceedings of WoLLIC'2003, volume 85, Electronic Notes in Theoretical Computer Science*.
- Bonato, Roberto and Christian Retoré. 2001. Learning rigid lambek grammars and minimalist grammars from structured sentences. *Third workshop on Learning Language in Logic, Strasbourg* .
- Buszkowski, W. 1997. Mathematical linguistics and proof theory. In van Benthem and ter Meulen (1997), chap. 12, pages 683–736.
- Buszkowski, Wojciech and Gerald Penn. 1990. Categorial grammars determined from linguistic data by unification. *Studia Logica* 49:431–454.
- de Groote, Philippe and François Lamarche. 2002. Classical non-associative lambek calculus. *Studia Logica* 71.1 (2).
- Foret, Annie and Yannick Le Nir. 2002. Lambek rigid grammars are not learnable from strings. In *COLING'2002, 19th International Conference on Computational Linguistics*. Taipei, Taiwan.

- Gold, E.M. 1967. Language identification in the limit. *Information and control* 10:447–474.
- Kanazawa, Makoto. 1998a. *Learnable Classes of Categorical Grammars*. Studies in Logic, Language and Information. Stanford, California: Center for the Study of Language and Information (CSLI) and The European association for Logic, Language and Information (FOLLI).
- Kanazawa, Makoto. 1998b. *Learnable classes of categorical grammars*. Studies in Logic, Language and Information. FoLLI & CSLI. distributed by Cambridge University Press.
- Lambek, Joachim. 1961. On the calculus of syntactic types. In R. Jakobson, ed., *Structure of language and its mathematical aspects*, pages 166–178. American Mathematical Society.
- Moortgat, Michael. 1997. Categorical type logic. In van Benthem and ter Meulen (1997), chap. 2, pages 93–177.
- van Benthem, J. and A. ter Meulen, eds. 1997. *Handbook of Logic and Language*. Amsterdam: North-Holland Elsevier.

---

# Two Type 0-Variants of Minimalist Grammars

GREGORY M. KOBELE AND JENS MICHAELIS

## Abstract

Minimalist grammars (Stabler 1997) capture some essential ideas about the basic operations of sentence construction in the Chomskyan syntactic tradition. Their affinity with the unformalized theories of working linguists makes it easier to implement and thereby to better understand the operations appealed to in neatly accounting for some of the regularities perceived in language. Here we characterize the expressive power of two, apparently quite different, variations on the basic minimalist grammar framework, gotten by:

1. adding a mechanism of ‘feature percolation’ (Kobele, forthcoming), or
2. instead of adding a central constraint on movement (the ‘specifier island condition’, Stabler 1999), using it to replace another one (the ‘shortest move condition’, Stabler 1997, 1999) (Gärtner and Michaelis 2005).

We demonstrate that both variants have equal, unbounded, computing power by showing how each can simulate straightforwardly a 2-counter automaton.

**Keywords** MINIMALIST GRAMMARS, 2-COUNTER AUTOMATA, LOCALITY CONDITIONS, FEATURE PERCOLATION

## 8.1 Introduction

Recently, two variants of the *minimalist grammar* (*MG*) formalism introduced in Stabler 1997 have been discussed w.r.t. the issue of generative capacity (see Gärtner and Michaelis 2005 and Kobele, forthcoming).

Seen from a linguistic perspective, the motivation for studying the two variants arose from two rather different starting points: Kobele (forthcoming), attempting to provide a formalization of mechanisms used to account for pied-piping (Ross 1967), considers MGs endowed

with feature percolation from specifiers to (attracting) heads. Gärtner and Michaelis (2005), as part of a larger project to better understand the effects of constraint interaction in minimalist syntax, study the behaviour of the *specifier island constraint (SPIC)* in isolation from Stabler’s original *shortest move constraint (SMC)*.

What both variants have in common formally is that, in contrast to the original MG-type, they allow the generation of non-mildly context-sensitive languages: Kobele (forthcoming) shows how an arbitrary (*infinite*) *abacus* (Lambek 1961) can be simulated by an  $MG^{+perc}$ , an MG enriched with the corresponding mechanism of feature percolation. As a corollary he shows how any arbitrary recursively enumerable subset of the natural numbers can be derived as the string language of an  $MG^{+perc}$ . Thus, by means of an “MG-external” encoding (i.e. a computable, bijective function  $f_\Sigma : \mathbb{N} \rightarrow \Sigma^*$  for any finite set  $\Sigma$ ),<sup>1</sup>  $MG^{+perc}_S$  can be seen as deriving the class of all type 0-languages over arbitrary finite alphabets. However, the question of how to define an  $MG^{+perc}$  which directly derives an arbitrary type 0-language is left open.

Gärtner and Michaelis (2005) show that there is an  $MG^{-SMC,+SPIC}$ , an MG respecting the SPIC but not the SMC, which derives a non-semilinear string language, and they conjecture that each type 0-language is derivable by some  $MG^{-SMC,+SPIC}$ .

In this paper we prove the full Turing power of  $MG^{-SMC,+SPIC}_S$  as well as  $MG^{+perc}_S$ , showing that for each 2-counter automaton there is an  $MG^{-SMC,+SPIC}$  as well as an  $MG^{+perc}$  which both generate exactly the language accepted by the automaton. In fact, our construction of a corresponding  $MG^{+perc}$  is fully general, holding for all variants of the feature percolation mechanism proposed in Kobele, forthcoming.

## 8.2 2-Counter Automata

**Definition 13** A *2-counter automaton (2-CA)* is given by a 7-tuple  $M = \langle Q, \Sigma, \{\mathbf{1}, \mathbf{2}\}, \delta, \{\#_1, \#_2\}, q_0, Q_f \rangle$ , where  $Q$  and  $\Sigma$  are the finite sets of *states* and *input symbols*, respectively. For  $i \in \{1, 2\}$ ,  $\mathbf{i}$  is the *i-th counter symbol*, and  $\#_i$  is the *i-th end of stack symbol*.  $q_0 \in Q$  is the *initial state*,  $Q_f \subseteq Q$  is the set of *final states*,  $\delta$  is the *transition function* from  $Q \times \Sigma_\epsilon \times \{\mathbf{1}, \#_1\} \times \{\mathbf{2}, \#_2\}$  to  $\mathcal{P}_{\text{fin}}(Q \times \mathbf{1}^* \times \mathbf{2}^*)$ .<sup>2</sup> An *instantaneous configuration* is a 4-tuple from  $Q \times \mathbb{N} \times \mathbb{N} \times \Sigma^*$ . For  $a \in \Sigma_\epsilon$  we write  $\langle q, n_1, n_2, aw \rangle_M \stackrel{\leftarrow}{\sim} \langle q', n'_1, n'_2, w \rangle$  just in case  $\langle q', \gamma_1, \gamma_2 \rangle \in \delta(\langle q, a, g_1, g_2 \rangle)$ ,

<sup>1</sup>Throughout, we take  $\mathbb{N}$  to denote the set of natural numbers, including 0. For every set  $X$ ,  $X^*$  is the Kleene closure of  $X$ , including  $\epsilon$ , the empty string.

<sup>2</sup>For each set  $X$ ,  $X_\epsilon$  denotes the set  $X \cup \{\epsilon\}$ .  $\mathcal{P}_{\text{fin}}(X)$  is the set of all finite subsets of  $X$ . For a singleton set  $\{x\}$ , we often write  $x$ , if this does not cause misunderstandings. Thus,  $x^*$  denotes  $\{x\}^*$  under this convention.

where  $g_i = \#_i$  iff  $n_i = 0$ , and  $n'_i = |\gamma_i| + (n_i \dot{-} 1)$ .<sup>3</sup>  $\vdash_M^*$  is the reflexive-transitive closure of  $\vdash_M$ . The *language determined by*  $M$ ,  $L(M)$ , is the set  $\{w \mid \langle q_0, 0, 0, w \rangle \vdash_M^* \langle q, 0, 0, \epsilon \rangle \text{ for some } q \in Q_f\}$ .

It is known (cf. Hopcroft and Ullman 1979) that the class of languages determined by 2-CAs is exactly the class of type 0-languages.

### 8.3 Minimalist Grammars and Variations

It will be useful to explicitly mark the “outer complement line” and the corresponding specifiers of a minimalist expression. To do this we extend the notation from Stabler and Keenan 2003, introduced there in order to reduce the presentation of minimalist expressions to “the bare essentials.” Throughout we let  $\Sigma$  and  $Syn$  be disjoint sets, a finite set of *non-syntactic features*, the (*terminal*) *alphabet*, and a finite set of *syntactic features*, respectively, in accordance with (F1) and (F2). We take  $Feat$  to be the set  $\Sigma \cup Syn$ . Furthermore, we let  $::$ ,  $:$ , **comp**, **spec** and  $-$  be pairwise distinct new symbols, where, in particular,  $::$  and  $:$  serve to denote *lexical* and *derived* types, respectively.

(F1)  $Syn$  is partitioned into five sets:<sup>4</sup>

<i>Base</i>		a set of ( <i>basic</i> ) <i>categories</i>
<i>Select</i>	$= \{=x \mid x \in Base\}$	a set of <i>selectors</i>
<i>Licensees</i>	$= \{-x \mid x \in Base\}$	a set of <i>licensees</i>
<i>Licensors</i>	$= \{+x \mid x \in Base\}$	a set of <i>licensors</i>
<i>P-Licensors</i>	$= \{+\hat{x} \mid x \in Base\}$	a set of <i>p-licensors</i>

(F2)  $Base$  includes at least the category  $c$ .

**Definition 14** An element of  $\Sigma^* \times \{::, :\} \times Syn^* \times \{\mathbf{comp}, \mathbf{spec}, -\}$  is a *chain* (over  $Feat$ ).  $Chains$  denotes the set of all chains over  $Feat$ .

An element of  $Chains^* \setminus \{\epsilon\}$  is an *expression* (over  $Feat$ ).  $Exp$  denotes the set of all expressions over  $Feat$ .

For given  $\phi \in Syn^*$ ,  $\cdot_\alpha \in \{::, :\}$ , and  $z \in \{\mathbf{comp}, \mathbf{spec}, -\}$ , the chain  $\langle s, \cdot_\alpha, \phi, z \rangle$  is usually denoted as  $\langle s \cdot_\alpha \phi, z \rangle$ , and is said to *display* (*open*) *feature*  $f$  if  $\phi = f\chi$  for some  $f \in Syn$  and  $\chi \in Syn^*$ .

A classical MG in the sense of Stabler 1997 employs the functions *merge* and *move*, creating minimalist expressions from a finite lexicon  $Lex$ . The corresponding definitions are explicitly given w.r.t. trees, as in Stabler 1999 too. There a revised MG-type is introduced, obeying, besides the *shortest move condition* (*SMC*), a particular implementation of the *specifier island condition* (*SPIC*): to be movable, a constituent must

<sup>3</sup>For each set  $M$  and each  $w \in M^*$ ,  $|w|$  denotes the length of  $w$ . For all  $x, y \in \mathbb{N}$ ,  $x \dot{-} y$  is defined by  $x \dot{-} y = x - y$  if  $x > y$ , and by  $x \dot{-} y = 0$  otherwise.

<sup>4</sup>Elements from  $Syn$  will usually be typeset in **typewriter font**.

belong to the transitive complement closure of a given tree, or be a specifier of such a constituent.<sup>5</sup> The SPIC crucially implies that moving or merging a constituent  $\alpha$  into a specifier position blocks checking (by later movement) of any licensee feature displayed by some proper subconstituent of  $\alpha$ . Thus in order to avoid “crashing” derivations, only the lowest embedded complement within the complement closure displaying some licensee can move, and then only if it contains no specifier with an unchecked feature.<sup>6</sup>

The *minimalist grammar (MG)* types to be introduced below differ essentially in the definitions of their structure building functions. Accordingly, we first introduce those. We let  $s, t \in \Sigma^*$ ,  $\cdot_\alpha, \cdot_\beta \in \{\::, \cdot\}$ ,  $\mathbf{f} \in \text{Base}$ ,  $\phi, \chi \in \text{Syn}^*$ ,  $z \in \{\text{comp}, \text{spec}, -\}$ , and  $\alpha_1, \dots, \alpha_k, \beta_1, \dots, \beta_l \in \text{Chains}$  for some  $k, l \in \mathbb{N}$  such that  $\cdot_\alpha = \::$  implies  $k = 0$ , and we let  $i \in \mathbb{N}$  with  $i \leq k$ . Also, relevant in  $(\text{mo-fp}_\otimes)$ , we let  $\psi \in (\text{Syn} \setminus \text{Licensees})^*$ , and  $\phi', \chi' \in \text{Licensees}^*$ , and assume  $\otimes$  to be some linear, non-deleting function from  $\text{Syn}^* \times \text{Syn}^*$  to  $\text{Syn}^*$ , i.e.,  $\otimes$  neither deletes material nor inserts material not in its arguments.

(**me**) *merge* maps partially from  $\text{Exp} \times \text{Exp}$  to  $\text{Exp}$  such that the pair  $\langle \widehat{\alpha}, \widehat{\beta} \rangle$  built from the expressions  $\widehat{\alpha} = \langle \langle s \cdot_\alpha = \mathbf{f}\phi, - \rangle, \alpha_1, \dots, \alpha_k \rangle$  and  $\widehat{\beta} = \langle \langle t \cdot_\beta \mathbf{f}\chi, - \rangle, \beta_1, \dots, \beta_l \rangle$ , belongs to  $\text{Dom}(\text{merge})$ ,<sup>7</sup> and the value  $\text{merge}(\widehat{\alpha}, \widehat{\beta})$  is defined as

- |   |  |
|---|--|
| (me.1) $\langle \langle st : \phi, - \rangle, \beta_1, \dots, \beta_l \rangle$  | if $\cdot_\alpha = \::$ and $\chi = \epsilon$    |
| (me.2) $\langle \langle s : \phi, - \rangle, \langle t \cdot \chi, \text{comp} \rangle, \beta_1, \dots, \beta_l \rangle$                            | if $\cdot_\alpha = \::$ and $\chi \neq \epsilon$ |
| (me.3) $\langle \langle ts : \phi, - \rangle, \beta_1, \dots, \beta_l, \alpha_1, \dots, \alpha_k \rangle$   | if $\cdot_\alpha = \cdot$ and $\chi = \epsilon$  |
| (me.4) $\langle \langle s : \phi, - \rangle, \langle t \cdot \chi, \text{spec} \rangle, \beta_1, \dots, \beta_l, \alpha_1, \dots, \alpha_k \rangle$ | otherwise <sup>8</sup>                           |

(**me**<sup>+SPIC</sup>) The partial mapping  $\text{merge}^{+\text{SPIC}}$  from  $\text{Exp} \times \text{Exp}$  to  $\text{Exp}$  is defined such that the pair  $\langle \widehat{\alpha}, \widehat{\beta} \rangle$  built from the expressions  $\widehat{\alpha} = \langle \langle s \cdot_\alpha = \mathbf{f}\phi, - \rangle, \alpha_1, \dots, \alpha_k \rangle$  and  $\widehat{\beta} = \langle \langle t \cdot_\beta \mathbf{f}\chi, - \rangle, \beta_1, \dots, \beta_l \rangle$ , belongs to  $\text{Dom}(\text{merge})$  iff the *specifier island condition on merge* as expressed in (SPIC.me) is satisfied, in which case  $\text{merge}^{+\text{SPIC}}(\widehat{\alpha}, \widehat{\beta}) = \text{merge}(\widehat{\alpha}, \widehat{\beta})$ . The specifier island condition on

<sup>5</sup>It can be shown that, in terms of derivable string languages, this revised type defines a proper subclass of the original type (Michaelis 2005). That is to say, adding the SPIC to the SMC, in fact, reduces the weak generative capacity of the formalism.

<sup>6</sup>Note that, in (me.2), respectively (me.4) and (mo<sup>-SMC</sup>) below, a constituent displaying a further unchecked feature after *merge* or *move* has been applied, gets marked as complement or specifier, respectively.

<sup>7</sup>For a partial function  $f$  from a set  $A$  into a set  $B$ ,  $\text{Dom}(f)$  is the domain of  $f$ , i.e., the set of all  $x \in A$  for which  $f(x)$  is defined.

<sup>8</sup>Recall that by assumption,  $\cdot_\alpha = \::$  implies  $k = 0$ .



merger in effect enforces a constraint against proper left branch extraction, disallowing movement from inside a specifier (a left branch), by prohibiting the merger of specifiers which contain proper subconstituents potentially moving in a later derivation step:

If  $\cdot_\alpha =:$  then  $l = 0$ . (SPIC.me)

(**mo**<sup>-SMC</sup>)  $move^{-SMC}$  is a partial mapping from  $Exp$  to  $\mathcal{P}_{fin}(Exp)$  such that  $\hat{\alpha} = \langle \langle s \cdot_\alpha +\mathbf{f}\phi, - \rangle, \alpha_1, \dots, \alpha_{i-1}, \langle t \cdot_\beta -\mathbf{f}\chi, z \rangle, \alpha_{i+1}, \dots, \alpha_k \rangle$  belongs to  $Dom(move)$ , and the value  $move^{-SMC}(\hat{\alpha})$  includes

(mo<sup>-SMC</sup>.1)  $\langle \langle ts : \phi, - \rangle, \alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_k \rangle$  if  $\chi = \epsilon$

(mo<sup>-SMC</sup>.2)  $\langle \langle s : \phi, - \rangle, \langle t : \chi, \mathbf{spec} \rangle, \alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_k \rangle$  if  $\chi \neq \epsilon$

(**mo**<sup>-SMC,+SPIC</sup>)  $move^{-SMC,+SPIC}$  maps partially from  $Exp$  to  $\mathcal{P}_{fin}(Exp)$  with  $\hat{\alpha} = \langle \langle s \cdot_\alpha +\mathbf{f}\phi, - \rangle, \alpha_1, \dots, \alpha_{i-1}, \langle t \cdot_\beta -\mathbf{f}\chi, z \rangle, \alpha_{i+1}, \dots, \alpha_k \rangle$  belonging to  $Dom(move^{-SMC,+SPIC})$  iff the *specifier island condition on movement* as expressed in (SPIC.mo) is satisfied, and then  $move^{-SMC,+SPIC}(\hat{\alpha}) = move^{-SMC}(\hat{\alpha})$ . In conjunction with (SPIC.me), which ensures that the only way  $z$  can be **comp** is if it was introduced by (me.2), the specifier island condition on movement requires that all chains internal to the subtree whose root is the chain in question have themselves moved out before it is permitted to move.

If  $z = \mathbf{comp}$  then  $i = k$ . (SPIC.mo)

(**mo**)  $move$  is a partial mapping from  $Exp$  to  $\mathcal{P}_{fin}(Exp)$  such that  $\hat{\alpha} = \langle \langle s \cdot_\alpha +\mathbf{f}\phi, - \rangle, \alpha_1, \dots, \alpha_{i-1}, \langle t \cdot_\beta -\mathbf{f}\chi, z \rangle, \alpha_{i+1}, \dots, \alpha_k \rangle$  belongs to  $Dom(move)$  iff the *shortest move constraint* as expressed in (SMC) holds, in which case  $move(\hat{\alpha}) = move^{-SMC}(\hat{\alpha})$ . The shortest move constraint disallows ‘competition’ for the same position—where there is competition, there is a loser, and thus something that will move farther than it had to.

None of the chains  $\alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_k$  displays **-f**.

(SMC)

(**mo**-**fp**<sub>⊗</sub>)  $move\text{-}fp_{\otimes}$  is a partial mapping from  $Exp$  to  $\mathcal{P}_{fin}(Exp)$  with  $\hat{\alpha}_{\otimes} = \langle \langle s : +\mathbf{f}\psi\phi', - \rangle, \alpha_1, \dots, \alpha_{i-1}, \langle t : -\mathbf{f}\chi', z \rangle, \alpha_{i+1}, \dots, \alpha_k \rangle$  belonging to  $Dom(move\text{-}fp_{\otimes}^{-SMC})$  iff (SMC) is satisfied, and with  $move\text{-}fp_{\otimes}(\hat{\alpha})$  including the expression

(—)  $\langle \langle ts : \psi(\phi' \otimes \chi'), - \rangle, \alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_k \rangle$ .<sup>9</sup>

(**mo**<sup>+perc</sup>)  $move_{\otimes}^{+perc}$  is the partial mapping from  $Exp$  to  $\mathcal{P}_{\text{fin}}(Exp)$  which results from the union of  $move$  and  $move\text{-fp}_{\otimes}$ .

In the following,  $Lex$  denotes a *lexicon (over Feat)*, i.e.,  $Lex$  is a finite set of simple expressions of lexical type, more concretely, a finite subset of  $\Sigma \times \{::\} \times Syn^* \times \{-\}$ .

**Definition 15 (Gärtner and Michaelis 2005)** An *MG without SMC, but obeying the SPIC* ( $MG^{-SMC,+SPIC}$ ) is a tuple  $\langle \Sigma, Syn, \{::, ::\}, Lex, \Omega, c \rangle$  with  $\Omega$  being the set  $\{merge^{+SPIC}, move^{-SMC,+SPIC}\}$ .

**Definition 16 (Kobele, forthcoming)** For a linear, non-deleting mapping  $\otimes$  from  $Syn^* \times Syn^*$  to  $Syn^*$ , a 6-tuple  $\langle \Sigma, Syn, \{::, ::\}, Lex, \Omega, c \rangle$  is called an *MG with percolation from specifiers to heads* ( $MG^{+perc}$ ) if  $\Omega = \{merge, move_{\otimes}^{+perc}\}$ .

For  $G = \langle \Sigma, Syn, \{::, ::\}, Lex, \Omega, c \rangle$ , an  $MG^{-SMC,+SPIC}$ , or  $MG^{+perc}$ , the *closure of G*,  $CL(G)$ , is the set  $\bigcup_{k \in \mathbb{N}} CL^k(G)$ , where  $CL^0(G) = Lex$ , and for  $k \in \mathbb{N}$ ,  $CL^{k+1}(G) \subseteq Exp$  is recursively defined as

$$CL^k(G) \cup \{merge'(v, \phi) \mid \langle v, \phi \rangle \in Dom(merge') \cap CL^k(G) \times CL^k(G)\} \\ \cup \bigcup_{v \in Dom(move') \cap CL^k(G)} move'(v)$$

with  $merge' = merge^{+SPIC}$  if  $G$  is an  $MG^{-SMC,+SPIC}$ , and  $merge' = merge$  otherwise, and with  $move' \in \Omega \setminus \{merge'\}$ . The *(string) language derivable by G*,  $L(G)$ , is the set  $\{s \in \Sigma^* \mid \langle s \cdot c, - \rangle \in CL(G) \text{ for } \cdot \in \{::, ::\}\}$ .

## 8.4 Simulating a 2-Counter Automata

Let  $M = \langle Q, \Sigma, \{\mathbf{1}, \mathbf{2}\}, \delta, \{\#\mathbf{1}, \#\mathbf{2}\}, q_0, Q_f \rangle$  be a 2-CA. In constructing an  $MG^{-SMC,+SPIC}$ ,  $G_1$ , and an  $MG^{+perc}$ ,  $G_2$ , with  $L(G_1) = L(G_2) = L(M)$ , we take  $\#\Sigma$ ,  $\mathbf{1}$  and  $\mathbf{2}$  to be new, pairwise distinct symbols:

$$Base = \{c, \#\Sigma\} \cup \{q \mid q \in Q\} \cup \{\mathbf{1}, \mathbf{2}\} \\ \cup \{0_{qajkr\gamma_1\gamma_2}, 1_{qajkr\gamma_1\gamma_2}^{(h)}, 2_{qajkr\gamma_1\gamma_2}^{(i)} \mid \\ q, r \in Q, a \in \Sigma_{\epsilon}, j \in \{\mathbf{1}, \#\mathbf{1}\}, k \in \{\mathbf{2}, \#\mathbf{2}\}, \gamma_1 \in \mathbf{1}^*, \gamma_2 \in \mathbf{2}^* \\ \text{with } \langle r, \gamma_1, \gamma_2 \rangle \in \delta(\langle q, a, j, k \rangle), 0 \leq h \leq |\gamma_1|, 0 \leq i \leq |\gamma_2|\}$$

For  $q, r \in Q$ ,  $a \in \Sigma_{\epsilon}$ ,  $j \in \{\mathbf{1}, \#\mathbf{1}\}$ ,  $k \in \{\mathbf{2}, \#\mathbf{2}\}$ ,  $\gamma_1 \in \mathbf{1}^*$ ,  $\gamma_2 \in \mathbf{2}^*$  such that  $\langle r, \gamma_1, \gamma_2 \rangle \in \delta(\langle q, a, j, k \rangle)$ , the categories  $1_{qajkr\gamma_1\gamma_2}^{(h)}$  and  $2_{qajkr\gamma_1\gamma_2}^{(i)}$  are likewise denoted by  $1_{qajkr\gamma_1\gamma_2}$  and  $2_{qajkr\gamma_1\gamma_2}$ , respectively.

<sup>9</sup>Note that  $move(\hat{\alpha})$  and  $move\text{-fp}_{\otimes}(\hat{\alpha})$ , in (mo) and (mo-fp) respectively, both are singleton sets because of (SMC). Thus, these functions can easily be identified with one from  $Exp$  to  $Exp$ .

An  $\text{MG}^{-\text{SMC},+\text{SPIC}}$ -expression which represents an instantaneous configuration  $\langle q, n_1, n_2, t \rangle$ , derived from an initial configuration  $\langle q_0, 0, 0, st \rangle$  in a 2-CA will have the following shape:

$$(*) \quad \langle \langle \epsilon : q, - \rangle, \underbrace{\langle \epsilon : -2, \text{comp} \rangle, \dots, \langle \epsilon : -1, \text{comp} \rangle, \dots}_{n_2 \text{ times}}, \underbrace{\langle s : -\#_\Sigma, \text{comp} \rangle}_{n_1 \text{ times}} \rangle$$

**Construction 1**  $G_1 = \langle \Sigma, \text{Syn}, \{::, \cdot\}, \text{Lex}_1, \Omega, c \rangle$  is the  $\text{MG}^{-\text{SMC},+\text{SPIC}}$  with  $L(G_1) = L(M)$  such that  $\text{Lex}_1$  contains exactly the items:

$$\begin{aligned} \phi_0 &= \langle \epsilon :: q_0 -\#_\Sigma, - \rangle \\ \text{For all } q, r \in Q, a \in \Sigma_\epsilon, j \in \{\mathbf{1}, \#\mathbf{1}\}, k \in \{\mathbf{2}, \#\mathbf{2}\}, \gamma_1 \in \mathbf{1}^*, \gamma_2 \in \mathbf{2}^* \text{ such that } \langle r, \gamma_1, \gamma_2 \rangle \in \delta(\langle q, a, j, k \rangle) \\ \chi_{qajkr\gamma_1\gamma_2} &= \langle a :: =q +\#_\Sigma 0_{qajkr\gamma_1\gamma_2} -\#_\Sigma, - \rangle \\ \text{If } j = \#\mathbf{1} \text{ then} \\ \alpha_{qajkr\gamma_1\gamma_2}^0 &= \langle \epsilon :: =0_{qajkr\gamma_1\gamma_2} 1_{qajkr\gamma_1\gamma_2}^{(0)}, - \rangle \\ \text{else} \\ \alpha_{qajkr\gamma_1\gamma_2}^{0'} &= \langle \epsilon :: =0_{qajkr\gamma_1\gamma_2} +1 1_{qajkr\gamma_1\gamma_2}^{(0)}, - \rangle \\ \text{For } 0 \leq h < |\gamma_1| \\ \alpha_{qajkr\gamma_1\gamma_2}^{h+1} &= \langle \epsilon :: =1_{qajkr\gamma_1\gamma_2}^{(h)} 1_{qajkr\gamma_1\gamma_2}^{(h+1)} -1, - \rangle \\ \alpha_{qajkr\gamma_1\gamma_2}^{\sim} &= \langle \epsilon :: =1_{qajkr\gamma_1\gamma_2} +1 1_{qajkr\gamma_1\gamma_2} -1, - \rangle \\ \text{If } k = \#\mathbf{2} \text{ then} \\ \beta_{qajkr\gamma_1\gamma_2}^0 &= \langle \epsilon :: =1_{qajkr\gamma_1\gamma_2} 2_{qajkr\gamma_1\gamma_2}^{(0)}, - \rangle \\ \text{else} \\ \beta_{qajkr\gamma_1\gamma_2}^{0'} &= \langle \epsilon :: =1_{qajkr\gamma_1\gamma_2} +2 2_{qajkr\gamma_1\gamma_2}^{(0)}, - \rangle \\ \text{For } 0 \leq i < |\gamma_2| \\ \beta_{qajkr\gamma_1\gamma_2}^{i+1} &= \langle \epsilon :: =2_{qajkr\gamma_1\gamma_2}^{(i)} 2_{qajkr\gamma_1\gamma_2}^{(i+1)} -2, - \rangle \\ \beta_{qajkr\gamma_1\gamma_2}^{\sim} &= \langle \epsilon :: =2_{qajkr\gamma_1\gamma_2} +2 2_{qajkr\gamma_1\gamma_2} -2, - \rangle \\ \psi_{qajkr\gamma_1\gamma_2} &= \langle \epsilon :: =2_{qajkr\gamma_1\gamma_2} r, - \rangle \\ \text{For each } q \in Q_f, \\ \phi_q &= \langle \epsilon :: =q +\#_\Sigma c, - \rangle \end{aligned}$$

Each derivation necessarily starts with  $\phi_0$  being selected, either by  $\phi_{q_0}$  in case  $q_0 \in Q_f$ , or by  $\chi_{q_0ajkr\gamma_1\gamma_2}$  for some  $r \in Q$ ,  $a \in \Sigma_\epsilon$ ,  $j \in \{\mathbf{1}, \#\mathbf{1}\}$ ,  $k \in \{\mathbf{2}, \#\mathbf{2}\}$ ,  $\gamma_1 \in \mathbf{1}^*$  and  $\gamma_2 \in \mathbf{2}^*$ . Generally, when an expression is selected by a lexical item of the form  $\chi_{qajkr\gamma_1\gamma_2}$  for some  $q, r \in Q$ ,  $a \in \Sigma_\epsilon$ ,  $j \in \{\mathbf{1}, \#\mathbf{1}\}$ ,  $k \in \{\mathbf{2}, \#\mathbf{2}\}$ ,  $\gamma_1 \in \mathbf{1}^*$  and  $\gamma_2 \in \mathbf{2}^*$ ,  $G_1$  begins simulating the CA's application of  $\delta$  to  $\langle q, a, j, k \rangle$  with outcome  $\langle r, \gamma_1, \gamma_2 \rangle$  w.r.t. some matching instantaneous configuration: the currently recognized  $a$  from the input tape is introduced, and afterwards the already recognized prefix (being the alphabetic string of the last chain within the currently derived expression displaying  $-\#_\Sigma$ , cf.  $(*)$ ) is moved to the left of  $a$ .

This creates an expression whose first chain displays  $0_{qajkr\gamma_1\gamma_2}$ . Depending on  $j$ , this expression is selected either by  $\alpha_{qajkr\gamma_1\gamma_2}^0$ , which does not consume an instance of  $\mathbf{1}$  from the 1st CA-counter, or  $\alpha_{qajkr\gamma_1\gamma_2}^{0'}$ , which does by applying  $move^{-SMC,+SPIC}$  in the next step. The resulting expression is selected successively by  $\alpha_{qajkr\gamma_1\gamma_2}^{h+1}$  for  $0 \leq h < |\gamma_1|$ , each time introducing a new complement displaying  $-1$ , and finally creating an expression whose first chain displays  $1_{qajkr\gamma_1\gamma_2}$ . Thus, the correct number of new instances of  $\mathbf{1}$  on the 1st CA-counter are introduced. Now, all the lower complements in the complement closure, which display  $-1$  (and which were created in an earlier cycle simulating another application of  $\delta$ ) are cycled through to the top of the expression, first merging with  $\alpha_{qajkr\gamma_1\gamma_2}^{\sim}$ , then applying  $move^{-SMC,+SPIC}$ . This ends in a configuration in which all chains displaying  $-1$  are consecutive components within the expression derived, immediately following the first chain which still displays  $1_{qajkr\gamma_1\gamma_2}$ . Analogously, proceeding from here by means of the items  $\beta_{qajkr\gamma_1\gamma_2}^{0'}$ ,  $\beta_{qajkr\gamma_1\gamma_2}^i$  for  $0 \leq i \leq |\gamma_2|$ , and  $\beta_{qajkr\gamma_1\gamma_2}^{\sim}$ , the operation of the 2nd CA-counter is simulated. Merging with  $\psi_{qajkr\gamma_1\gamma_2}$ , this procedure results in an expression, the first chain of which displays category  $\mathbf{r}$  (which corresponds to the CA being in state  $r$ ), first followed by the “true” number of consecutive chains displaying  $-2$ , then by the “true” number of consecutive chains displaying  $-1$ , and finally by a chain which displays  $-\#\Sigma$ , and contains as alphabetic material the prefix of the CA-input string recognized so far by the 2-CA simulated. Relying on this, and recalling that, in particular, (SPIC.mo) holds, it is rather easy to verify that a derivation ends in a single-chain expression of the form  $\langle s \cdot \mathbf{c}, - \rangle$  for some  $s \in \Sigma^*$  and  $\cdot \in \{::, :\}$  if, and only if  $s \in L(M)$ .

Constructing the  $MG^{+perc} G_2$  (and arguing that it does its job) works much the same as for the  $MG^{-SMC,+SPIC} G_1$ , and turns out to be even somewhat more straightforward, since both the “ $\alpha$ -” and “ $\beta$ -part” can be reduced to just two alternative chains. This is due to the fact that we can simulate the correct behavior of the  $i$ -th CA-counter,  $i = 1, 2$ , in terms of feature percolation: all  $\mathbf{i}$ -instances currently belonging to the  $i$ -th counter appear as one string of  $-\mathbf{i}$ -instances within one chain representing the counter. Because of this, each CA transition is simulated by just four applications of *merge*, and up to three applications of  $move_{\otimes}^{+perc}$ —one application of  $move_{\otimes}^{+perc}$  is to properly order the previously parsed word with the currently scanned symbol, and each of the other two are to attract the chain with the appropriate licensee features for percolation (and thus happen only when such chains exist, cf. next paragraph including  $(*_2)$  and footnote 10).

An  $\text{MG}^{+\text{perc}}$ -expression representing an instantaneous configuration  $\langle q, n_1, n_2, t \rangle$ , derived from an initial configuration  $\langle q_0, 0, 0, st \rangle$  in a 2-CA looks like the following, where  $(-i)^n$ ,  $i = 1, 2$ , is the string consisting of  $n$  instances of  $-i$ :<sup>10</sup>

$$(*) \quad \langle \langle \epsilon : \mathbf{q}, - \rangle, \langle \epsilon : (-2)^{n_2}, \text{comp} \rangle, \langle \epsilon : (-1)^{n_1}, \text{comp} \rangle, \langle s : -\#\Sigma, \text{comp} \rangle \rangle$$

**Construction 2** Let  $G_2 = \langle \Sigma, \text{Syn}, \{::, \cdot\}, \text{Lex}_2, \Omega, c \rangle$  be the  $\text{MG}^{+\text{perc}}$  with  $L(G_2) = L(M)$  such that  $\text{Lex}_2$  contains exactly the items below:

$$\phi_0 = \langle \epsilon :: \mathbf{q}_0 -\#\Sigma, - \rangle$$

For all  $q, r \in Q$ ,  $a \in \Sigma_\epsilon$ ,  $j \in \{\mathbf{1}, \#\mathbf{1}\}$ ,  $k \in \{\mathbf{2}, \#\mathbf{2}\}$ ,  $\gamma_1 \in \mathbf{1}^*$ ,  $\gamma_2 \in \mathbf{2}^*$

such that  $\langle r, \gamma_1, \gamma_2 \rangle \in \delta(\langle q, a, j, k \rangle)$

$$\chi_{qajkr\gamma_1\gamma_2} = \langle \mathbf{a} :: =\mathbf{q} +\#\Sigma \ 0_{qajkr\gamma_1\gamma_2} \ -\#\Sigma, - \rangle$$

If  $j = \#\mathbf{1}$  then

$$\alpha_{qajkr\gamma_1\gamma_2} = \langle \epsilon :: =0_{qajkr\gamma_1\gamma_2} \ 1_{qajkr\gamma_1\gamma_2} \ (-1)^{|\gamma_1|}, - \rangle$$

else

$$\alpha'_{qajkr\gamma_1\gamma_2} = \langle \epsilon :: =0_{qajkr\gamma_1\gamma_2} \ +\widehat{1} \ 1_{qajkr\gamma_1\gamma_2} \ (-1)^{|\gamma_1|}, - \rangle$$

If  $k = \#\mathbf{2}$  then

$$\beta_{qajkr\gamma_1\gamma_2} = \langle \epsilon :: =1_{qajkr\gamma_1\gamma_2} \ 2_{qajkr\gamma_1\gamma_2} \ (-2)^{|\gamma_2|}, - \rangle$$

else

$$\beta'_{qajkr\gamma_1\gamma_2} = \langle \epsilon :: =1_{qajkr\gamma_1\gamma_2} \ +\widehat{2} \ 2_{qajkr\gamma_1\gamma_2} \ (-2)^{|\gamma_2|}, - \rangle$$

$$\psi_{qajkr\gamma_1\gamma_2} = \langle \epsilon :: =2_{qajkr\gamma_1\gamma_2} \ \mathbf{r}, - \rangle$$

For each  $q \in Q_f$ ,

$$\phi_q = \langle \epsilon :: =\mathbf{q} +\#\Sigma \ c, - \rangle$$

Again, a derivation begins either with with some  $\chi_{q_0ajkr\gamma_1\gamma_2}$  selecting  $\phi_0$ , or with  $\phi_{q_0}$  doing so in case  $q_0 \in Q_f$  (implying that  $\epsilon$  belongs to  $L(M)$ ). As in the case of  $G_1$ , in general, when—within a derivation—an expression is selected by a lexical item  $\phi_q$  for some  $q \in Q_f$ , or a lexical item  $\chi_{qajkr\gamma_1\gamma_2}$  for some  $r \in Q$ ,  $a \in \Sigma_\epsilon$ ,  $j \in \{\mathbf{1}, \#\mathbf{1}\}$ ,  $k \in \{\mathbf{2}, \#\mathbf{2}\}$ ,  $\gamma_1 \in \mathbf{1}^*$  and  $\gamma_2 \in \mathbf{2}^*$ , the selected expression corresponds to an instantaneous configuration derived by the 2-CA from an initial configuration. In both cases  $\text{move}_{\otimes}^{+\text{perc}}$  applies to the resulting expression, checking an instance of  $-\#\Sigma$ . In the former case this ends in a complete expression, if there is no chain left displaying an unchecked licensee feature. Otherwise further derivation steps are blocked. In the latter case  $G_2$  begins simulating the CA's application of  $\delta$  to  $\langle q, a, j, k \rangle$  with outcome  $\langle r, \gamma_1, \gamma_2 \rangle$  w.r.t. some matching instantaneous configuration: after having moved by means of  $\text{move}_{\otimes}^{+\text{perc}}$  the already recognized prefix to the

<sup>10</sup>When  $n_i = 0$ , the respective chain is not present in the expression.

front of the newly scanned instance of  $a$ , the derivation continues by merging an  $\alpha$ -expression—note that if  $\alpha_{qajkr\gamma_1\gamma_2}$  is merged with an expression which itself contains a chain already in possession of  $-1$  features, the SMC will disallow further operations on the resulting expression (in essence, crashing the derivation).<sup>11</sup> This ensures that only those derivations involving  $\alpha_{qajkr\gamma_1\gamma_2}$  succeed, in which it combines with an expression which is completely devoid of  $-1$  features at the point of merger. If  $\alpha'_{qajkr\gamma_1\gamma_2}$  is merged instead, then the chain containing the  $-1$  features will move, and, as per the definition of  $move_{\otimes}^{+perc}$ , percolate its features (modulo the one checked) to the initial chain. In either case, the initial chain comes to host all of the expression's  $-1$  features. We then merge a  $\beta$  expression, where there transpires something similar. The simulation of the 2-CA transition is complete once  $\psi_{qajkr\gamma_1\gamma_2}$  is merged, resulting in an expression which must once again be selected by some  $\phi_r$  or  $\chi_{rajkr\gamma_1\gamma_2s}$ .

Note that, involving feature percolation,  $G_2$  only creates chains in which all licensee instances result from the same licensee, i.e., either  $-1$  or  $-2$ . Thus,  $G_2$  is defined completely independently of the percolation function  $\otimes$  from  $Syn^* \times Syn^*$  to  $Syn^*$  underlying  $move_{\otimes}^{+perc}$ .

## 8.5 Conclusion

We reviewed two apparently unrelated extensions to minimalist grammars, and showed that both of them can derive arbitrary r.e. sets of strings. Moreover, much the same construction sufficed to show this for each variant of MGs presented herein. This highlights that at least a subpart of each of these extensions have similar strong generative capacities. How similar these variants are is a matter left for another time. We note here only that while the 2-CA simulation given for  $MG^{-SMC,+SPIC}$  extends straightforwardly to one of a *queue automaton*, no similarly straightforward extension exists for the  $MG^{+perc}$ , as we were able to nullify our ignorance of the function  $\otimes$  only by (in effect) reducing its domain to strings over a single alphabetic symbol.

## References

Gärtner, Hans-Martin and Jens Michaelis. 2005. A note on the complexity of constraint interaction. *Locality conditions and minimalist grammars*.

<sup>11</sup>This is true so long as  $\gamma_1$  is non-zero. To keep the presentation simple, we impose the condition on  $\delta$  that for  $\langle q', \gamma_1, \gamma_2 \rangle \in \delta(\langle q, a, g_1, g_2 \rangle)$ ,  $\gamma_i = \epsilon$  entails that  $g_i \neq \#_i$ . That this is a normal form can be seen by the fact that we can eliminate a ‘forbidden’ transition  $\langle q', \epsilon, \gamma_2 \rangle \in \delta(\langle q, a, \#_1, g_2 \rangle)$  by the pair of new transitions  $\langle q'', \mathbf{1}, \gamma_2 \rangle \in \delta(\langle q, a, \#_1, g_2 \rangle)$  and  $\langle q', \epsilon, \gamma_2 \rangle \in \delta(\langle q'', \epsilon, \mathbf{1}, g_2 \rangle)$ , where  $q''$  is a new state not in  $Q$ . Analogously, we can eliminate a transition  $\langle q', \gamma_1, \epsilon \rangle \in \delta(\langle q, a, g_1, \#_2 \rangle)$ .

- In P. Blache, E. P. Stabler, J. Busquets, and R. Moot, eds., *Logical Aspects of Computational Linguistics (LACL '05)*, Lecture Notes in Artificial Intelligence Vol. 3492, pages 114–130. Berlin, Heidelberg: Springer.
- Hopcroft, John E. and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley.
- Kobele, Gregory M. forthcoming. Features moving madly. *Research on Language and Computation* Draft version available at <http://www.linguistics.ucla.edu/people/grads/kobele/papers.htm>.
- Lambek, Joachim. 1961. How to program an (infinite) abacus. *Canadian Mathematical Bulletin* 4:295–302.
- Michaelis, Jens. 2005. An additional observation on strict derivational minimalism. In *FG-MoL 2005. The 10th conference on Formal Grammar and The 9th Meeting on Mathematics of Language*, Edinburgh. This volume.
- Ross, John R. 1967. *Constraints on Variables in Syntax*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA. Appeared as *Infinite Syntax*. Ablex, Norwood, NJ, 1986.
- Stabler, Edward P. 1997. Derivational minimalism. In C. Retoré, ed., *Logical Aspects of Computational Linguistics (LACL '96)*, Lecture Notes in Artificial Intelligence Vol. 1328, pages 68–95. Berlin, Heidelberg: Springer.
- Stabler, Edward P. 1999. Remnant movement and complexity. In G. Bouma, G. M. Kruijff, E. Hinrichs, and R. T. Oehrle, eds., *Constraints and Resources in Natural Language Syntax and Semantics*, pages 299–326. Stanford, CA: CSLI Publications.
- Stabler, Edward P. and Edward L. Keenan. 2003. Structural similarity within and among languages. *Theoretical Computer Science* 293:345–363.





---

# Learnability of Some Classes of Optimal Categorical Grammars

JACEK MARCINIEC

## Abstract

In this paper we characterize a learnable class of classical categorical grammars, which lies inbetween (learnable) class of rigid grammars and (not learnable) class of optimal grammars. The learning function we provide is based on *guided optimal unification*, introduced in our paper Marciniec (2004).

**Keywords** CATEGORIAL GRAMMAR, LEARNING, UNIFICATION

## 9.1 Introduction

Kanazawa investigates in Kanazawa (1998) several classes of classical categorical grammars from the point of view of their learnability. All the learning functions discussed there are based on unification algorithms — the standard one and its *optimal* version, introduced by Buszkowski and Penn (1990). They also involve the process of reconstructing the grammar on the basis of some linguistic data, originated in Buszkowski (1987), van Benthem (1987). Only the learning function for the class of rigid grammars is based solely on unification. The rest is a combination of several operations, unification being just one of them. Kanazawa follows two ways of designing unification based learning procedures. The first one approach involves some preliminary operations *before* standard unification is set to work. It can be, for example, additional partitioning of the search space like in the case of *k-valued* grammars.

The second approach incorporates optimal unification accompanied with some selection mechanism *after* calculating all optimal unifiers. Usually the selection mechanism mentioned above is based on mini-

mality of some sort. Here Kanazawa’s *least cardinality* case may serve as an example.

Both standard and optimal unification possess the feature desirable as far as learnability is concerned, namely compactness — the (optimal) unification image of an infinite set of types can be determined on the basis of its finite subset (cf. Marciniak (1997b,a, 2004)). However, the difference between the two is essential — standard unification algorithm outputs unique (if any) solution, whereas the total number of optimal unifiers usually grows with the increase of the input. Therefore, the only possible operation after completion of the former is to accept (or reject) the solution. The latter is more flexible. The application of a post unification choice function is evident. One can also imagine some activity prior to the unification process, though no such a possibility has been elaborated so far.

In Marciniak (2004), we put forward another solution — incorporating selection mechanism *into* unification engine itself. We described a general framework for optimal unification discovery procedure where the limitation of the number of outputs is achieved by controlling the order in which types are unified, ‘decreasing’ this way the nondeterminism of the original algorithm.

In this paper we develop the case when types that are alphabetic variants are to be unified first. Consequently, we introduce the class of semi-rigid grammars, which lies inbetween the class of rigid grammars and the class of optimal grammars (introduced in Kanazawa (1998)). We prove learnability of that class.

## 9.2 Preliminaries

We adopt most of the notation from Buszkowski and Penn (1990).  $\text{FS}(V)$  denotes the set of all *functor-argument* structures on the set of *atoms*  $V$ . If  $A = (A_1, \dots, A_n)_i$  is a structure, then a *substructure*  $A_i$  is the *functor* whereas each *substructure*  $A_j$ , for  $j \neq i$ , is an *argument* of  $A$ . For any set  $T$  of functor-argument structures, by  $\text{SUB}(T)$  we denote the smallest set satisfying the following conditions:  $T \subseteq \text{SUB}(T)$ , if  $(A_1, \dots, A_n)_i \in \text{SUB}(T)$  then also  $A_j \in \text{SUB}(T)$ , for  $j = 1, \dots, n$ . The functor (of the subfunctor) of a structure  $A$  is also its *subfunctor*. The only subfunctor of a structure  $A$  that is an atom will be denoted by  $\uparrow_f(A)$ .

*Types* are all the elements of the set  $\text{Tp} = \text{FS}(\text{Pr})$ , where  $\text{Pr} = \text{Var} \cup \{\text{S}\}$  and  $\text{Var}$  is a countable set of *variables* and  $\text{S}$  is a *designated* primitive type ( $\text{S} \notin \text{Var}$ ). A *substitution* is any homomorphism  $\alpha : \text{Tp} \mapsto \text{Tp}$ , such that  $\alpha(\text{S}) = \text{S}$ . Two types  $t_1$  and  $t_2$  are *alphabetic variants*

$(t_1 \bowtie t_2)$ , if  $t_2 = \alpha_2(t_1)$  and  $t_1 = \alpha_1(t_2)$  for some substitutions  $\alpha_1$  and  $\alpha_2$ . A substitution  $\sigma$  unifies a family of sets of types  $\mathcal{T} = \{T_1, \dots, T_n\}$ , if each  $\sigma[T_j]$  is a singleton. For a substitution  $\alpha$ , by  $\sim_\alpha$  we denote the relation defined as follows:  $t_1 \sim_\alpha t_2$  iff  $\alpha(t_1) = \alpha(t_2)$ . For an equivalence relation  $\sim$  and  $T \subseteq \mathbf{Tp}$ , by  $T/\sim$  we will denote the partition induced on  $T$  by  $\sim$ . For  $\mathcal{T} = \{T_1, \dots, T_n\}$ , we define  $\mathcal{T}/\sim = T_1/\sim \cup \dots \cup T_n/\sim$ .

By a (finite) classical categorial grammar (from now on simply a (finite) grammar) we mean any (finite) relation  $G \subseteq V \times \mathbf{Tp}$ . However, in what follows, it will be convenient to regard a grammar as a family  $\{I_G(v) : v \in V_G\}$ , where  $V_G$  is a *lexicon* of  $G$  and the function  $I_G$  from  $V_G$  to  $2^{\mathbf{Tp}}$  is its *initial type assignment* ( $I_G(v) = \{t \in \mathbf{Tp} : \langle v, t \rangle \in G\}$ ). The *terminal type assignment*  $T_G$  of  $G$  is defined by the following rule:  $t_i \in T_G((A_1, \dots, A_n)_i)$  iff there exist  $t_j$  such that  $t_j \in T_G(A_j)$  for  $j \neq i$ , and  $(t_1, \dots, t_n)_i \in T_G(A_i)$ . Each grammar  $G$  and a type  $t$  determine the *category* of type  $t$ :  $\text{CAT}_G(t) = \{A \in \text{FS}(V_G) : t \in T_G(A)\}$ . By  $\text{FL}(G) = \text{CAT}_G(\mathbf{S})$  we denote the (functorial) language determined by  $G$ .

A language  $L \subseteq \text{FS}(V)$  is said to be *finitely describable* if there exists a finite grammar  $G$  such that  $L \subseteq \text{FL}(G)$ .

$\mathbf{Tp}(G) = \bigcup_{v \in V_G} \text{SUB}(I_G(v))$ . By  $\mathbf{Tp}_a(G)$  we denote the subset of  $\mathbf{Tp}(G)$ , consisting of only argument substructures. For a grammar  $G$  and a substitution  $\alpha$ ,  $\alpha[G]$  will denote the grammar  $\{\alpha[I_G(v)] : v \in V_G\}$ .

A grammar  $G$  is *rigid* if  $\text{card}(I_G(v)) = 1$  for all  $v \in V_G$ . Let  $G$  be any grammar. Denote  $\overline{V} = V \times \mathbb{N}$ . The elements of  $\overline{V}$  (possible ‘copies’ of atoms from  $V$ ) will be denoted by  $v^i$  rather than  $\langle v, i \rangle$ . By *rigid counterpart* of  $G$  we mean any rigid grammar  $\overline{G} \subseteq \overline{V} \times \mathbf{Tp}$ , fulfilling the condition:  $I_G(v) = \bigcup_{i=1}^n I_{\overline{G}}(v^i)$ , where  $n = \text{card}(I_G(v))$ . By  $(\cdot)^\uparrow$  we denote a homomorphism from  $\text{FS}(\overline{V})$  to  $\text{FS}(V)$  fulfilling the condition:  $(v^i)^\uparrow = v$ , for each  $i$ . For any grammar  $G$ , we have  $\text{FL}(G) = (\text{FL}(\overline{G}))^\uparrow$ .

A type  $t \in \mathbf{Tp}(G)$  is *useless* if  $\text{SUB}(\text{FL}(\overline{G})) \cap \text{CAT}_{\overline{G}}(t) = \emptyset$  for any rigid counterpart  $\overline{G}$  of  $G$ .

A grammar  $G$  is said to be *optimal* if it has no useless type and for all  $v \in V_G$ , if  $t_1, t_2 \in I_G(v)$  and  $t_1 \neq t_2$  then  $\{t_1, t_2\}$  is not unifiable.

Throughout this paper *learnability* means *learnability from structures* in the sense of Gold’s identification in the limit (cf. Gold (1967), Kanazawa (1998), Jain et al. (1999), Osherson et al. (1997)).

### 9.3 Unification and Infinity

Below we recapitulate some results from Marciniak (2004):

**Definition 17** A substitution  $\sigma$  partially unifies a family of possibly infinite sets of types  $\mathcal{T} = \{T_1, \dots, T_n\}$ , if  $\text{card}(T_j) < \aleph_0$  for each  $j$ .

**Definition 18** Let  $\mathcal{T} = \{T_1, \dots, T_n\}$  be a family of possibly infinite sets of types. An *optimal unifier* for  $\mathcal{T}$  is a substitution  $\sigma$  fulfilling the following conditions:

- $\mathcal{T} / \sim_\sigma$  is finite,
- $\sigma$  is the most general unifier of  $\mathcal{T} / \sim_\sigma$ ,
- for  $i \in \{1, \dots, n\}$ ,  $a, b \in T_i$ , if  $\sigma(a) \neq \sigma(b)$ , then  $\{\sigma(a), \sigma(b)\}$  is not unifiable.

**Proposition 18** *The number of optimal unifiers of infinite set of types does not have to be finite.*

**Theorem 19 (Marciniec (2004))** *Let  $\mathcal{T} = \{T_1, \dots, T_n\}$  be partially unifiable. For each optimal unifier  $\eta$  for  $\mathcal{T}$  there exist a family  $\mathcal{U} = \{U_1, \dots, U_n\}$  and an optimal unifier  $\sigma$  for  $\mathcal{U}$ , such that for each  $i \in \{1, \dots, n\}$ ,  $U_i \subseteq T_i$ ,  $U_i$  is finite and  $\eta[T_i] = \sigma[U_i]$ .*

**Definition 19** Let  $\approx$  be an equivalence relation on  $\text{Tp}$ . A substitution  $\alpha$  respects  $\approx$  on  $T \subseteq \text{Tp}$ , if the following condition holds:

$$(\forall t_1, t_2 \in T)(t_1 \approx t_2) \rightarrow \alpha(t_1) = \alpha(t_2).$$

A substitution  $\alpha$  respects  $\approx$  on  $\mathcal{T} = \{T_1, \dots, T_n\}$ , if it respects  $\approx$  on each  $T_i$ , for  $i \in \{1, \dots, n\}$ .

**Proposition 20** *For any family  $\mathcal{T} = \{T_1, \dots, T_n\}$ , such that each  $T_i$  is bounded in length, there are only finitely many optimal unifiers respecting the relation  $\approx$ .*

**Proof**  $\mathcal{T} / \approx$  is finite and unifiable and any substitution respecting  $\approx$  unifies  $\mathcal{T} / \approx$ .  $\square$

**Theorem 21 (Marciniec (2004))** *Let  $\mathcal{T} = \{T_1, \dots, T_n\}$  and let each  $T_i$  be finite. The following algorithm (guided optimal unification algorithm) outputs precisely the optimal unifiers for  $\mathcal{T}$ , respecting  $\approx$ :*

- Compute mgu  $\eta$  for  $\mathcal{T} / \approx$
- Compute all optimal unifiers for  $\{\eta(T_1), \dots, \eta(T_n)\}$ .

**Theorem 22 (Marciniec (2004))** *let  $\mathcal{T} = \{T_1, \dots, T_n\}$  be partially unifiable family of nonempty, possibly infinite sets of types. There exists  $\mathcal{U} = \{U_1, \dots, U_n\}$ , such that the set of all optimal unifiers for  $\mathcal{U}$  respecting  $\approx$  is the same as for any  $\mathcal{V} = \{V_1, \dots, V_n\}$ , where each  $V_i$  is finite and  $U_i \subseteq V_i \subseteq T_i$ .*

### 9.4 Semi-rigid Grammars

**Definition 20** Let  $A \in \text{FS}(V)$ . We will describe a construction of a grammar  $\text{GF}(A)$  — *general form* determined by  $A$ . At first we choose any  $\bar{A} \in \text{SUB}(\bar{V})$ , such that  $(\bar{A})^\uparrow = A$  and each atom from  $\bar{V}$  occurs in  $\bar{A}$  at most once (compare the definition of a rigid counterpart). Now, by induction, we define a mapping  $\mapsto$  from  $\text{SUB}(\bar{A})$  to  $\text{Tp}$ :

- $\bar{A} \mapsto S$ ,
- if  $(\bar{A}_1, \dots, \bar{A}_n)_i \mapsto t$  then for each  $j \neq i$  we set  $\bar{A}_j \mapsto x_j$ , where  $x_j$  is a ‘new’ variable, and  $\bar{A}_i \mapsto (x_1, \dots, x_{i-1}, t, x_{i+1}, \dots, x_n)_i$ .

$\text{GF}(A) = \{\langle v, t \rangle \in V \times \text{Tp} : \exists i \in \mathbb{N}(v^i \mapsto t)\}$ . Finally, for  $L \subseteq \text{FS}(V)$ , assuming that  $\text{Tp}(\text{GF}(A)) \cap \text{Tp}(\text{GF}(B)) = \emptyset$  when  $A \neq B$ , we define  $\text{GF}(L) = \bigcup_{A \in L} \text{GF}(A)$ . After Marciniak (1997b), we admit the case of  $L$  being infinite.

**Proposition 23** Let  $G = \alpha[\text{GF}(L)]$  for some substitution  $\alpha$ . For any  $x \in \text{Var}$ , if  $x \in \text{Tp}(\text{GF}(L))$  then  $\alpha(x) \in \text{Tp}_a(G)$ .

**Proposition 24** For any rigidly describable  $L$ ,  $\text{GF}(L)$  is bounded in length.

**Definition 21** Let  $L$  be finitely describable. By  $OG_{\bowtie}(L)$  we will denote the set of all grammars  $G$ , such that  $G = \eta[\text{GF}(L)]$  for some optimal unifier  $\eta$  for  $\text{GF}(L)$ , respecting  $\bowtie$  on  $\text{GF}(L)$ .

From Propositions 20 and 24, we have:

**Corollary 25** For any finitely describable  $L$ , the set  $OG_{\bowtie}(L)$  is finite.

**Proposition 26** Let  $G$  be a grammar. There exists a finite set  $D \subseteq \text{FL}(G)$  such that for each set  $E$ , if  $D \subseteq E \subseteq \text{FL}(G)$  then  $OG_{\bowtie}(E) = OG_{\bowtie}(\text{FL}(G))$ .

**Proposition 27** Let  $L$  be finitely describable. Then

$$L \subseteq \bigcap_{G \in OG_{\bowtie}(L)} \text{FL}(G).$$

**Definition 22** Let  $u, v \in \text{Tp}$ . We define the relation  $\overset{\curvearrowright}{\subseteq} \subseteq \text{SUB}(u) \times \text{SUB}(v)$ :

$$u \overset{\curvearrowright}{\subseteq}_{\{u,v\}} v \quad (9.21)$$

$$\text{if } (u_1, \dots, u_n)_i \overset{\curvearrowright}{\subseteq}_{\{u,v\}} (v_1, \dots, v_n)_i \text{ then } u_i \overset{\curvearrowright}{\subseteq}_{\{u,v\}} v_i \quad (9.22)$$

**Proposition 28** If  $t'_1 \overset{\curvearrowright}{\subseteq}_{\{t_1, t_2\}} t'_2$  then  $\alpha(t'_1) \overset{\curvearrowright}{\subseteq}_{\{\alpha(t_1), \alpha(t_2)\}} \alpha(t'_2)$ , for any substitution  $\alpha$ .

**Definition 23** We define the relation  $\uparrow \sqsubseteq \mathbf{Tp} \times \mathbf{Tp}$ :

$t_1 \uparrow t_2$  iff

$$\left[ \uparrow_f(t_1) \overset{\leftarrow}{\rightsquigarrow}_{\{t_1, t_2\}} \uparrow_f(t_2) \right] \wedge [\uparrow_f(t_1) = \uparrow_f(t_2) \vee \{\uparrow_f(t_1), \uparrow_f(t_2)\} \subseteq \mathbf{Var}].$$

**Example 1** To see that the two following types are related in the sense of  $\uparrow$ :

$$(x_1, ((x_2, x_3)_2, x_7)_1, x_2, (x_3, x_4)_2)_2, \\ ((x_7, x_2)_1, (((x_3, x_2)_1, x_1)_2, (x_1, x_2)_2)_1, x_4, (x_1, x_5, x_4)_1)_2,$$

we simply disregard all the argument subtypes:

$$(\circ, ((\circ, x_3)_2, \circ)_1, \circ, \circ)_2, \\ (\circ, ((\circ, x_1)_2, \circ)_1, \circ, \circ)_2.$$

**Proposition 29** For any substitution  $\alpha$  and a type  $t$ , if  $\uparrow_f(t) = S$  then  $t \uparrow \alpha(t)$ .

**Definition 24** A grammar  $G$  is said to be a *semi-rigid grammar* if it is optimal and the following conditions hold:

for all  $v \in V_G$ ,  $t_1, t_2 \in I_G(v)$  and types  $t'_1, t'_2$ :

$$t_1 \neq t_2 \rightarrow \neg(t_1 \uparrow t_2), \quad (9.23)$$

$$t_1 \neq t_2 \wedge t'_1 \overset{\leftarrow}{\rightsquigarrow}_{\{t_1, t_2\}} t'_2 \rightarrow t'_1 \notin \mathbf{Tp}_a(G) \vee t'_2 \notin \mathbf{Tp}_a(G). \quad (9.24)$$

**Example 2** Below we will denote types traditionally, writing

$$t_1, \dots, t_{i-1} \setminus t_i / t_{i+1}, \dots, t_n$$

instead of  $(t_1, \dots, t_n)_i$ . Let  $G_1$  denote the following grammar over the lexicon  $V = \{a, b\}$ :

$$a \mapsto x, x \setminus S \\ b \mapsto x \setminus x, (x \setminus S) \setminus (x \setminus S)$$

$G_1$  is not semi-rigid because both related types  $x$  and  $x \setminus S$  occur as argument subtypes in the type assignment. Our next example,  $G_2$ :

$$a \mapsto x, x \setminus S \\ b \mapsto x \setminus x, x \setminus (x \setminus S)$$

is semi-rigid (observe that  $\mathbf{FL}(G_1) = \mathbf{FL}(G_2)$ ). Notice also that semi-rigidness does not impose any restrictions concerning the number of types assigned to lexicon elements. For example,  $G_2$  would admit for  $b$  any number of types of the form  $x \setminus (x \setminus (\dots (x \setminus S) \dots))$ .

**Theorem 30** The class of all semi-rigid grammars is learnable.

**Proof** Let  $G$  be a semi-rigid grammar. At first, we will show:

$$G \in OG_{\bowtie}(\mathbf{FL}(G)). \quad (9.25)$$

Let  $\overline{G}$  be any rigid counterpart of  $G$ . Denote  $L = \text{FL}(G)$  and  $\overline{L} = \text{FL}(\overline{G})$ . Since  $\overline{G}$  has no useless type,  $\overline{G} = \eta[\text{GF}(\overline{L})]$  for some mgu of  $\text{GF}(\overline{L})$  (cf. Kanazawa (1998)). We may assume that  $\text{GF}(L) = (\text{GF}(\overline{L}))^\uparrow$ , so  $G = \eta[\text{GF}(L)]$ . Since  $\{I_{\text{GF}(\overline{L})}(v) : v \in V_{\text{GF}(\overline{L})}\} = \{I_{\text{GF}(L)}(v) : v \in V_{\text{GF}(L)}\} / \sim_\eta$  and  $G$  is optimal, the substitution  $\eta$  is an optimal unifier of  $\text{GF}(L)$ . To prove (9.25), we have to show that  $\eta$  respects  $\bowtie$ .

Let  $t_i \bowtie t_j$ , where  $\{t_i, t_j\} \subseteq I_{\text{GF}(L)}(v)$ .

Suppose  $\{\uparrow_f(t_i), \uparrow_f(t_j)\} \not\subseteq \text{Var}$ . By the definition of  $\uparrow_f$ , we have  $\uparrow_f(t_i) = \uparrow_f(t_j) = \text{S}$ . Since  $\bowtie \subseteq \uparrow_f$  and  $\uparrow_f$  is transitive, by Proposition 29, we get  $\eta(t_i) \uparrow_f \eta(t_j)$  and consequently, by (9.23),  $\eta(t_i) = \eta(t_j)$ .

Suppose then  $\{\uparrow_f(t_i), \uparrow_f(t_j)\} \subseteq \text{Var}$ . Denote  $x_i = \uparrow_f(t_i)$  and  $x_j = \uparrow_f(t_j)$ . Since  $t_i \uparrow_f t_j$  from Proposition 28 and the definition of  $\uparrow_f$  it follows:

$$\eta(x_i) \overset{\{\eta(t_i), \eta(t_j)\}}{\longleftrightarrow} \eta(x_j). \quad (9.26)$$

By Proposition 23, both  $\eta(x_i)$  and  $\eta(x_j)$  are argument substructures of  $\text{Tp}(G)$ . By (9.24) and (9.26) we get  $\eta(t_i) = \eta(t_j)$  again.

Let  $\mu$  denote any computable function which, from any finite set of grammars, selects a grammar minimal with respect to the language it determines. Since the problem  $\text{FL}(G_1) \subseteq \text{FL}(G_2)$  is decidable, such a function exists. We define a function  $\varphi$  from  $\text{FS}(V)^*$  to the set of grammars:

$$\varphi(\langle s_0, \dots, s_i \rangle) = \mu(OG_{\bowtie}(\{s_0, \dots, s_i\})).$$

Let  $L = \text{FL}(G)$  for some semi-rigid grammar  $G$  and let  $L = \{s_i : i \in \mathbb{N}\}$ . By Proposition 26, there exists  $n \in \mathbb{N}$  such that for all  $i \geq n$  we have  $OG_{\bowtie}(\{s_0, \dots, s_i\}) = OG_{\bowtie}(\text{FL}(G))$ . Denote  $G' = \varphi(\langle s_0, \dots, s_n \rangle)$ . By Proposition 27, we get  $\text{FL}(G) \subseteq \text{FL}(G')$ . However, (9.25) and the minimality of  $G'$  leads to the conclusion that  $\text{FL}(G') = \text{FL}(G) = L$ . Hence,  $\varphi$  learns the class of all semi-rigid grammars. It is also easy to observe, that  $\varphi$  is responsive, set-driven and consistent on the class.  $\square$

**Proposition 31** *The learning function  $\varphi$ , defined in the proof of Theorem 30, is not prudent.*

**Proof** The grammar:

$$\begin{aligned} a &\mapsto y, y \setminus \text{S} \\ b &\mapsto \text{S} / z \\ c &\mapsto z, y \setminus \text{S} \end{aligned}$$

determines the following language:  $L = \{(a, a)_2, (b, c)_1, (a, c)_2\}$ . It is easy to check, that  $\varphi$  converges on  $L$  to:

$$\begin{aligned} a &\mapsto x, x \setminus S \\ b &\mapsto S / (x \setminus S) \\ c &\mapsto x \setminus S \end{aligned}$$

that is not semi-rigid. □

## References

- Buszkowski, Wojciech. 1987. Discovery procedures for categorial grammars. In Klein and van Benthem (1987).
- Buszkowski, Wojciech and Gerald Penn. 1990. Categorial grammars determined from linguistic data by unification. *Studia Logica* XLIX(4):431–454.
- Gold, E Mark. 1967. Language identification in the limit. *Information and Control* 10:447–474.
- Jain, Sanjay, Daniel Osherson, James S. Royer, and Arun Sharma. 1999. *Systems that Learn: An Introduction to Learning Theory*. Cambridge, Massachusetts: MIT Press, 2nd edn.
- Kanazawa, Makoto. 1998. *Learnable Classes of Categorial Grammars*. Studies in Logic, Language and Information. Stanford, California: CSLI Publications & FoLLI.
- Klein, Evan and Johan van Benthem, eds. 1987. *Categories, Polymorphism and Unification*. Amsterdam: Universiteit van Amsterdam.
- Marciniec, Jacek. 1997a. Connected sets of types and categorial consequence. In C. Retoré, ed., *Logical Aspects of Computational Linguistics*, vol. 1328 of *Lecture Notes in Artificial Intelligence*, pages 292–309. Berlin: Springer.
- Marciniec, Jacek. 1997b. Infinite set unification with application to categorial grammar. *Studia Logica* LVIII(3):339–355.
- Marciniec, Jacek. 2004. Optimal unification of infinite sets of types. *Fundamenta Informaticae* 62(3,4):395–407.
- Osherson, Daniel, Dick de Jongh, Eric Martin, and Scott Weinstein. 1997. Formal learning theory. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*, pages 737–775. Elsevier Science B. V.
- van Benthem, Johan. 1987. Categorial equations. In Klein and van Benthem (1987).



---

## An Additional Observation on Strict Derivational Minimalism

JENS MICHAELIS

### Abstract

We answer a question which, so far, was left an open problem: does—in terms of derivable string languages—the type of a *minimalist grammar* (*MG*) as originally introduced in Stabler 1997 defines a proper superclass of the revised type of an MG and, therefore, the type of a *strict MG* both introduced in Stabler 1999, and known to be weakly equivalent? For both the revised as well as the strict MG-type, the essential difference to the original MG-definition consists in imposing—in addition to the corresponding implementation of the *shortest move condition*—a second condition on the move-operator providing a formulation of the *specifier island condition*, and as such, restricting (further) the domain to which the operator can apply. It has been known already that this additional condition, in fact, ensures that—in terms of derivable string languages—the revised and, therefore, the strict MG-type both constitute a subclass of the original MG-type. We here present a string language proving that the inclusion is proper.

**Keywords** (STRICT) MINIMALIST GRAMMARS, SPECIFIER ISLAND CONDITION, MULTIPLE CONTEXT-FREE GRAMMARS/LINEAR CONTEXT-FREE REWRITING SYSTEMS, (LINEAR) CONTEXT-FREE TREE GRAMMARS

### 10.1 Introduction

The *minimalist grammar* (*MG*) formalism introduced in Stabler 1997 provides an attempt at a rigorous algebraic formalization of the perspectives currently adopted within the linguistic framework of transformational grammar. As has been shown (Michaelis 2001a, 2001b, Harkema 2001), this MG-type determines the same class of derivable string languages as *linear context-free rewriting systems* (*LCFRSs*) (Vijay-Shanker et al. 1987, Weir 1988).

*FG-MoL 2005.*  
James Rogers (ed.).  
Copyright © 2009, CSLI Publications.

Inspired, i.a., by the linguistic work presented in Koopman and Szabolcsi 2000, in Stabler 1999 a revised MG-type has been proposed whose essential departure from the version in Stabler 1997 can be seen as the following: in addition to the *shortest move constraint (SMC)*, a second locality condition, the *specifier island constraint (SPIC)*, is imposed on the move-operator regulating which maximal projection may move *overtly* into the highest specifier position. Deviating from the operator *move* as originally defined in Stabler 1997, a constituent has to belong to the transitive complement closure of a given tree or to be a specifier of such a constituent in order to be movable. An MG of this type, henceforth, is referred to as  $MG^{+SPIC}$ .

Closely in keeping with some further suggestions in Koopman and Szabolcsi 2000, a certain type of a *strict minimalist grammar (SMG)* has been introduced in Stabler 1999 as well: implementing the SPIC with somewhat more “strictness,” leading to *heavy pied-piping* constructions, the SMG-type allows only movement of constituents belonging to the transitive complement closure of a tree. But in contrast to the  $MG^{+SPIC}$ -type, the triggering licensee feature may head the head-label of any constituent within the reflexive-transitive specifier closure of a moving constituent.

$MG^{+SPIC}$ s and SMGs have been shown to be weakly equivalent by Michaelis (2004, 2002) confirming a conjecture explicitly stated in Stabler 1999. The equivalence turned out proving that, in terms of derivable languages,  $MG^{+SPIC}$ s and SMGs not only are subsumed by LCFRSs, but both are equivalent to a particular subclass of the latter, referred to as  $LCFRS_{1,2}$ -type: the righthand side of each rewriting rule of a corresponding LCFRS involves at most two nonterminals, and if two nonterminals appear on the righthand side then only simple strings of terminals are derivable from the first one.<sup>1</sup> It was, however, left unsolved, whether the respective classes of string languages derivable by  $LCFRS_{1,2}$ s and unrestricted LCFRSs—and thus the respective classes of string languages derivable by  $MG^{+SPIC}$ s (or, likewise, SMGs) as defined in Stabler 1999 and MGs as defined in Stabler 1997—are identical.<sup>2</sup>

In this paper we show that the inclusion is in fact proper. We implicitly do so by expressing the reduced structural generative capacity

---

<sup>1</sup>Exactly this condition expresses the strict opacity of specifiers within the  $MG^{+SPIC}$ -version.

<sup>2</sup>Note that, instead of adding the SPIC to the original MG-formalism, using it to simply replace the SMC does not lead to a reduction of the class of derivable string languages. Quite the opposite, the resulting type of MG even allows derivation of every type 0-language (Kobelev and Michaelis 2005).

in terms of a homomorphism mapping trees to strings. Explicitly, we show that, although a particular language (combining “string reversal,” “simple copying” and “intervening balanced bracketing in terms of the context-free Dyck language”) is derivable by an LCFRS, it’s not derivable by an LCFRS<sub>1,2</sub>. The most crucial part of our proof consists in showing that for each LCFRS<sub>1,2</sub>, there is a linear *context-free tree grammar* (cf. Rounds 1970a,b, Fischer 1968, Engelfriet and Schmidt 1977) deriving, modulo a homomorphism, the same string language in *inside-out mode*.

## 10.2 Context-Free Tree Grammars

Giving our definition of a *context-free tree grammar* (CFTG) as it goes back to the work Rounds (1970a,b) and Fischer (1968), we mainly lean on the presentation in Engelfriet and Schmidt 1977.

**Definition 25** A *ranked alphabet*,  $\Sigma$ , is an indexed family  $\langle \Sigma_n \mid n \in \mathbb{N} \rangle$  of pairwise disjoint sets.<sup>3</sup> For  $n \in \mathbb{N}$ , a  $\sigma \in \Sigma_n$  is an *operator of rank  $n$* , whose rank is denoted by  $\text{rank}(\sigma)$ . The *set of trees (over  $\Sigma$ )*,  $T(\Sigma)$ , is built up recursively using the operators in the usual way: if for some  $n \in \mathbb{N}$ , we have  $\sigma \in \Sigma_n$  and  $t_1, \dots, t_n \in T(\Sigma)$  then  $t = \sigma(t_1, \dots, t_n)$  is a tree. The *yield of  $t$* ,  $\text{yield}(t) \in \Sigma_0^*$ , is defined by  $\text{yield}(t) = \sigma$  if  $n = 0$ , and  $\text{yield}(t) = \text{yield}(t_1) \cdots \text{yield}(t_n)$  otherwise. A tree  $t' \in T(\Sigma)$  is a *subtree (of  $t$ )* if  $t' = t$ , or if  $t'$  is a subtree of  $t_i$  for some  $1 \leq i \leq n$ .

Throughout we let  $X = \{x_1, x_2, x_3, \dots\}$  be a countable set of variables, and for  $k \in \mathbb{N}$ , we define  $X_k \subseteq X$  as  $\{x_1, \dots, x_k\}$ . Then for a ranked alphabet  $\Sigma$ , the set of  *$k$ -ary trees (over  $\Sigma$ )*,  $T(\Sigma, X_k)$ , is the set of trees  $T(\Sigma')$  over the ranked alphabet  $\Sigma' = \langle \Sigma'_n \mid n \in \mathbb{N} \rangle$ , where  $\Sigma'_0 = \Sigma_0 \cup X_k$ , and  $\Sigma'_n = \Sigma_n$  for  $n > 0$ . Let  $T(\Sigma, X) = \bigcup_{k \in \mathbb{N}} T(\Sigma, X_k)$ .

**Definition 26** A *context-free tree grammar* (CFTG),  $\Gamma$ , is a 5-tuple  $\langle \Sigma, \mathcal{F}, \mathcal{S}, X, P \rangle$ , where  $\Sigma$  and  $\mathcal{F}$  are finite ranked alphabets of *inoperatives* and *operatives*, respectively.  $\mathcal{S}$  is a distinguished element in  $\mathcal{F}_n$  for some  $n \in \mathbb{N}$ , the *start symbol*.  $P$  is a finite set of productions. Each  $p \in P$  is of the form  $F(x_1, \dots, x_n) \rightarrow t$  for some  $n \in \mathbb{N}$ , where  $F \in \mathcal{F}_n$ ,  $x_1, \dots, x_n \in X$ , and  $t \in T(\Sigma \cup \mathcal{F}, X_n)$ . If, in addition, for each such  $p \in P$ , no  $x_i$  occurs more than once in  $t$  then  $\Gamma$  is *linear*.

For  $t, t' \in T(\Sigma \cup \mathcal{F}, X)$ ,  $t'$  is *directly derivable* from  $t$  ( $t \Rightarrow t'$ ) if for some  $m$  and  $n \in \mathbb{N}$ , there are a  $t_0 \in T(\Sigma \cup \mathcal{F}, X_{n+1})$  containing exactly *one* occurrence of  $x_{n+1}$ , a production  $F(x_1, \dots, x_m) \rightarrow t'' \in P$ , and  $t_1, \dots, t_m \in T(\Sigma \cup \mathcal{F}, X)$  such that  $t = t_0[x_1, \dots, x_n, F(t_1, \dots, t_m)]$

<sup>3</sup>Throughout the paper the following conventions apply:  $\mathbb{N}$  is the set of all non-negative integers. For any set  $M$ ,  $M^*$  denotes the Kleene closure of  $M$ , including  $\epsilon$ , the empty string.  $M_\epsilon$  is the set  $M \cup \{\epsilon\}$ .

and  $t' = t_0[x_1, \dots, x_n, t''[t_1, \dots, t_m]]$ .<sup>4</sup> If  $x_{n+1}$  is not dominated in  $t_0$  by an operative then  $t'$  is derived from  $t$  by an *outside-in (OI)* step ( $t \Rightarrow_{\text{OI}} t'$ ).<sup>5</sup> If  $t_1, t_2, \dots, t_n \in T(\Sigma, X)$  then  $t'$  is derived by an *inside-out (IO)* step ( $t \Rightarrow_{\text{IO}} t'$ ).  $\Rightarrow^*$ ,  $\Rightarrow_{\text{OI}}^*$  and  $\Rightarrow_{\text{IO}}^*$  denote the reflexive-transitive closures of  $\Rightarrow$ ,  $\Rightarrow_{\text{OI}}$  and  $\Rightarrow_{\text{IO}}$ , respectively.

The (tree) languages derivable by  $\Gamma$  in unrestricted, OI- and IO-mode are  $\mathcal{L}(\Gamma) = \{t \in T(\Sigma) \mid \mathcal{S} \Rightarrow^* t\}$ ,  $\mathcal{L}_{\text{OI}}(\Gamma) = \{t \in T(\Sigma) \mid \mathcal{S} \Rightarrow_{\text{OI}}^* t\}$  and  $\mathcal{L}_{\text{IO}}(\Gamma) = \{t \in T(\Sigma) \mid \mathcal{S} \Rightarrow_{\text{IO}}^* t\}$ , respectively. The corresponding string languages derivable by  $\Gamma$  are the sets  $L(\Gamma) = \{\text{yield}(t) \mid t \in \mathcal{L}(\Gamma)\}$ ,  $L_{\text{OI}}(\Gamma) = \{\text{yield}(t) \mid t \in \mathcal{L}_{\text{OI}}(\Gamma)\}$  and  $L_{\text{IO}}(\Gamma) = \{\text{yield}(t) \mid t \in \mathcal{L}_{\text{IO}}(\Gamma)\}$ , each of which being a subset of  $\Sigma_0^*$ .<sup>6</sup>

### 10.3 Linear Context-Free Rewriting Systems

The formalism of a *linear context-free rewriting systems (LCFRSs)* in the sense of Vijay-Shanker et al. 1987 can be seen as presenting a subtype of the formalism a *multiple context-free grammar (MCFG)* in the sense of Seki et al. 1991, where in terms of derivable string languages the generative power of LCFRSs is identical to that of MCFGs.

**Definition 27 (Seki et al. 1991, Vijay-Shanker et al. 1987)** A *multiple context-free grammar (MCFG)*,  $G$ , is a 5-tuple  $\langle N, T, F, R, S \rangle$ , where  $N$  and  $T$  are the finite sets of *nonterminals* and *terminals*, respectively. Each  $A \in N$  is associated with some  $d_G(A) \in \mathbb{N} \setminus \{0\}$ .  $S$  is a distinguished symbol from  $N$ , the *start symbol*, with  $d_G(S) = 1$ .  $F$  and  $R$  are the finite sets of *functions* and (*rewriting*) *rules*, respectively, such that each  $r \in R$  is of the form  $A_0 \rightarrow f(A_1, \dots, A_n)$  for some  $f \in F$  and  $A_0, A_1, \dots, A_n \in N$  for some  $n \in \mathbb{N}$ , where  $f$  is a linear regular function from  $(T^*)^{d_G(A_1)} \times \dots \times (T^*)^{d_G(A_n)}$  into  $(T^*)^{d_G(A_0)}$ , allowing deletion of single components.  $r$  is *nonterminating* in case  $n > 0$ , otherwise  $r$  is *terminating*. If the latter, we have  $f(\emptyset) \in (T^*)^{d_G(A_0)}$ , and we usually denote  $r$  in the form  $A_0 \rightarrow f(\emptyset)$ .

For  $A \in N$  and  $k \in \mathbb{N}$ ,  $L_G^k(A) \subseteq (T^*)^{d_G(A)}$  is given recursively by means of  $\theta \in L_G^0(A)$  for each terminating  $A \rightarrow \theta \in R$ , and for  $k \in \mathbb{N}$ ,  $\theta \in L_G^{k+1}(A)$  if  $\theta \in L_G^k(A)$ , or if there are  $A \rightarrow f(A_1, \dots, A_n) \in R$

<sup>4</sup>For each  $k \in \mathbb{N}$ , and given trees  $\tau \in T(\Sigma \cup \mathcal{F}, X_k)$  and  $\tau_1, \dots, \tau_k \in T(\Sigma \cup \mathcal{F}, X)$ ,  $\tau[\tau_1, \dots, \tau_k]$  is the tree in  $T(\Sigma \cup \mathcal{F}, X)$  resulting from substituting for  $1 \leq i \leq k$ , each occurrence of the (trivial) subtree  $x_i$  of  $\tau$  by an instance of  $\tau_i$ .

<sup>5</sup> $x_{n+1}$  is said to be *dominated in  $t_0$  by an operative  $A \in \mathcal{F}_k$*  for some  $k \in \mathbb{N}$ , if there are  $\tau_1, \dots, \tau_k \in T(\Sigma \cup \mathcal{F}, X)$  such that  $A(\tau_1, \dots, \tau_k)$  is a subtree of  $t_0$  which contains the unique occurrence of  $x_{n+1}$  in  $t_0$ .

<sup>6</sup>Note that  $\mathcal{L}(\Gamma) = \mathcal{L}_{\text{OI}}(\Gamma)$  holds for each CFTG  $\Gamma$ , but the class of context-free tree languages derivable in OI-mode and the one of those derivable in IO-mode are not comparable in full general (Fischer 1968).

and  $\theta_i \in L_G^k(A_i)$  for  $1 \leq i \leq n$  such that  $f(\theta_1, \dots, \theta_n) = \theta$ . The set  $L_G(A) = \bigcup_{k \in \mathbb{N}} L_G^k(A)$  is the *language derivable from A (by G)*.<sup>7</sup>  $L_G(S)$ , also denoted by  $L(G)$ , is the *multiple context-free language (MCFL) (derivable by G)*. We have  $L(G) \subseteq T^*$ , because  $d_G(S) = 1$ . The *rank of G*,  $\text{rank}(G)$ , is the number  $\max\{n \mid A_0 \rightarrow f(A_1, \dots, A_n) \in R\}$ , the *fan-out of G* is the number  $\max\{d_G(A) \mid A \in N\}$ .

If each  $f \in F$  appearing in some  $A_0 \rightarrow f(A_1, \dots, A_n) \in R$ , in addition, has the property that no component of the tuples of tuples of  $(T^*)^{d_G(A_1)} \times \dots \times (T^*)^{d_G(A_n)}$  is erased by mapping under  $f$  into  $(T^*)^{d_G(A_0)}$ , then  $G$  is a (*string based*) *linear context-free rewriting system (LCFRS)*, and  $L(G)$  is a (*string based*) *linear context-free rewriting language (LCFRL)*.

**Example** An LCFRS which has rank 2 and fan-out 3 is the LCFRS  $G_{\text{ex}} = \langle \{S, A, B\}, \{a, b, [, ]\}, \{\text{conc}, \text{id}, e_{[]}, e_a, e_b, f_a, f_b, g, h\}, R, S \rangle$  with  $d(A) = d(B) = 3$ , where  $R$  consists of the following rules:

$$\begin{aligned} S &\rightarrow \text{conc}(B), \\ A &\rightarrow e_a(\emptyset) \mid e_b(\emptyset) \mid f_a(A) \mid f_b(A) \mid h(B, B) \text{ and} \\ B &\rightarrow e_{[]}(\emptyset) \mid \text{id}(A) \mid g(B), \end{aligned}$$

and where the functions are given by:

$$\begin{array}{lll} \text{conc} : & \langle x_0, x_1, x_2 \rangle & \mapsto x_0x_1x_2 \\ \text{id} : & \langle x_0, x_1, x_2 \rangle & \mapsto \langle x_0, x_1, x_2 \rangle \\ e_{[]} : & \emptyset & \mapsto \langle \epsilon, [], \epsilon \rangle \\ e_a : & \emptyset & \mapsto \langle a, a, a \rangle \\ e_b : & \emptyset & \mapsto \langle b, b, b \rangle \\ f_a : & \langle x_0, x_1, x_2 \rangle & \mapsto \langle x_0a, x_1a, ax_2 \rangle \\ f_b : & \langle x_0, x_1, x_2 \rangle & \mapsto \langle x_0b, x_1b, bx_2 \rangle \\ g : & \langle x_0, x_1, x_2 \rangle & \mapsto \langle x_0, [x_1], x_2 \rangle \\ h : & \langle \langle x_0, x_1, x_2 \rangle, \langle y_0, y_1, y_2 \rangle \rangle & \mapsto \langle x_0y_0, y_1x_1, y_2x_2 \rangle \end{array}$$

The language derivable by  $G_{\text{ex}}$  is

$$L(G_{\text{ex}}) = \{w_1 \cdots w_n z_n w_n \cdots z_1 w_1 z_0 w_n^R \cdots w_1^R \mid n \in \mathbb{N} \setminus \{0\}, w_i \in \{a, b\}^+ \text{ for } 1 \leq i \leq n, z_n \cdots z_0 \in D\},$$

where  $D$  is the Dyck language of balanced parentheses, generated by the context free grammar  $G_D = \langle \{S\}, \{[, ]\}, \{S \rightarrow SS \mid [S] \mid \epsilon\}, S \rangle$ , and where for each  $w \in T^*$ ,  $w^R$  denotes the reversal of  $w$ .<sup>8</sup>

<sup>7</sup>Thus, employing the notion of the CFTG-derivation modes introduced above, an MCFG can be considered to derive a tuple of strings in IO-mode.

<sup>8</sup>Thus for each set  $M$  and  $w \in M^*$ ,  $w^R \in M^*$  is defined recursively by  $\epsilon^R = \epsilon$ , and  $(av)^R = v^R a$  for  $a \in M$  and  $v \in M^*$ .

The class of MCFLs and the class of LCFRLs are known to be identical (cf. Seki et al. 1991, Lemma 2.2). Theorem 11 in Rambow and Satta 1999, therefore, shows that for each MCFG  $G$  there is an LCFRS  $G'$  with  $\text{rank}(G') \leq 2$  for which  $L(G) = L(G')$  holds.

**Definition 28** An  $LCFRS_{1,2}$  is an LCFRS  $G$  according to Definition 27 such that  $\text{rank}(G) = 2$ , and  $d_G(A_1) = 1$  for each  $A_0 \rightarrow f(A_1, A_2) \in R$ . In this case  $L(G)$  is an  $LCFRL_{1,2}$ .

**Definition 29** A given  $LCFRS_{1,2}$   $G = \langle N, T, F, R, S \rangle$  is in  $LCFRS_{1,2}$ -normalform ( $LCFRS_{1,2}$ -NF) if each  $f \in F$  is of one of the forms (i)–(iii) for some  $m \in \mathbb{N} \setminus \{0\}$ , or of the form (iv) for some  $a \in T_\epsilon$ .

$$\begin{array}{lll} \text{(i)} & \langle \langle y_1, \langle x_1, x_2, \dots, x_m \rangle \rangle & \mapsto \langle y_1, x_1, x_2, \dots, x_m \rangle \\ \text{(ii)} & \langle \langle y_1, \langle x_1, x_2, \dots, x_m \rangle \rangle & \mapsto \langle y_1 x_1, x_2, \dots, x_m \rangle \\ \text{(iii)} & \langle x_1, x_2, \dots, x_{m+1} \rangle & \mapsto \langle x_{m+1} x_1, x_2, \dots, x_m \rangle \\ \text{(iv)} & \emptyset & \mapsto a \end{array}$$

**Proposition 32** For every  $LCFRS_{1,2}$   $G$ , there exists an  $LCFRS_{1,2}$   $G' = \langle N, T, F, R, S \rangle$  in  $LCFRS_{1,2}$ -NF such that  $L(G) = L(G')$ .

*Proof.* The proposition can essentially be proven applying a “double transformation” to a given  $LCFRS_{1,2}$ : first, using the construction presented in Michaelis 2004, the  $LCFRS_{1,2}$  is transformed into an  $MG^{+SPIC}$  deriving the same string language. Then, using the construction presented in Michaelis 2002, the resulting  $MG^{+SPIC}$  is transformed into an  $LCFRS_{1,2}$  of the corresponding normal form still deriving the same string language, but with (iv') instead of (iv).<sup>9</sup>

$$\text{(iv')} \quad \emptyset \quad \mapsto \quad w, w \in T^*$$

Verifying that (iv') can be strengthened to (iv) is straightforward.<sup>10</sup>  $\square$

## 10.4 Proper Inclusion within LCFRLs

Let  $G = \langle N, T, F, R, S \rangle$  be an  $LCFRS_{1,2}$  in  $LCFRS_{1,2}$ -NF.

**Construction** We now construct a linear CFTG  $\Gamma = \langle \Sigma, \mathcal{F}, \mathcal{S}, X, P \rangle$  with  $\Sigma_0 = T \cup \{\Lambda\}$  for a new symbol  $\Lambda$ , and with  $h[L_{IO}(\Gamma)] = L(G)$ ,<sup>11</sup> where  $h$  is the homomorphism from  $\Sigma_0^*$  to  $T^*$  determined by  $h(a) = a$  for  $a \in T$ , and  $h(\Lambda) = \epsilon$ . In constructing  $\Gamma$ , we assume  $\bullet$  and  $\mathcal{S}$  to be two further, new distinct symbols and let

<sup>9</sup>Here, (i) and (ii) simulate the behavior of the *merge*-operator, (iii) simulates the behavior of the *move*-operator, and (iv') provides “lexical insertion.”

<sup>10</sup>If need be, we simply add new nonterminals, functions and rules of the form  $A \rightarrow a$  and  $B \rightarrow f(C, D)$  to  $G'$ , where  $a \in T_\epsilon$  and  $f$  is in line with (ii), and successively replace all rules of the form (iv') not being in line with (iv).

<sup>11</sup>For any two sets  $M_1$  and  $M_2$ , and any mapping  $g$  from  $M_1$  into  $M_2$ ,  $g[M_1]$  denotes the image of  $M_1$  under  $g$ , i.e., the set  $\{g(m) \mid m \in M_1\} \subseteq M_2$ .

$$\begin{aligned}
 \Sigma_0 &= T \cup \{\Lambda\} \\
 \Sigma_2 &= \{\bullet\} \cup \{A' \mid A \in N \text{ and } d_G(A) = 2\} \\
 \Sigma_n &= \{A' \mid A \in N \text{ and } d_G(A) = n\} \text{ for } n \in \mathbb{N} \setminus \{0, 2\} \\
 \mathcal{F}_0 &= \{\mathcal{S}\} \cup \{\tilde{B} \mid B \in N\} \\
 \mathcal{F}_n &= \{\tilde{A}_D \mid A, D \in N \text{ and } d_G(A) = n\} \text{ for } n \in \mathbb{N} \setminus \{0\}
 \end{aligned}$$

Defining the set of productions in  $\Gamma$ , we first let

$$(s) \mathcal{S} \rightarrow \tilde{S} \in P.$$

For each terminating  $A \rightarrow \theta \in R$  for some  $A \in N$  and  $\theta \in (T^*)^{d_G(A)}$ , we have  $d_G(A) = 1$  and  $\theta \in T_\epsilon$ , because of (iv).

If  $\theta \neq \epsilon$  we let

$$(t.1) \tilde{B} \rightarrow \tilde{A}_B(\theta) \in P \text{ for each } B \in N, \text{ and}$$

$$(t.2) \tilde{A} \rightarrow A'(\theta) \in P.$$

If  $\theta = \epsilon$  we let

$$(t.1) \tilde{B} \rightarrow \tilde{A}_B(\Lambda) \in P \text{ for each } B \in N, \text{ and}$$

$$(t.2) \tilde{A} \rightarrow A'(\Lambda) \in P.$$

For each nonterminating  $A \rightarrow f(B) \in R$  for some  $A, B \in N$  and  $f \in F$ , we have  $f : \langle x_1, \dots, x_{d_G(B)} \rangle \mapsto \langle x_{d_G(B)}x_1, x_2, \dots, x_{d_G(B)-1} \rangle$ , because of (iii).

We let

$$(iii.1) \tilde{B}_D(x_1, \dots, x_{d_G(B)}) \rightarrow \tilde{A}_D(\bullet(x_{d_G(B)}, x_1), x_2, \dots, x_{d_G(B)-1}) \in P$$

for each  $D \in N$ , and

$$(iii.2) \tilde{B}_A(x_1, \dots, x_{d_G(B)}) \rightarrow A'(\bullet(x_{d_G(B)}, x_1), x_2, \dots, x_{d_G(B)-1}) \in P.$$

For each nonterminating  $A \rightarrow f(B, C) \in R$  for some  $A, B, C \in N$  and  $f \in F$ , because of (i) and (ii), we either have (i') or (ii').

$$(i') f : \langle \langle y_{d_G(B)} \rangle, \langle x_1, \dots, x_{d_G(C)} \rangle \rangle \mapsto \langle y_{d_G(B)}, x_1, \dots, x_{d_G(C)} \rangle$$

$$(ii') f : \langle \langle y_{d_G(B)} \rangle, \langle x_1, x_2, \dots, x_{d_G(C)} \rangle \rangle \mapsto \langle y_{d_G(B)}x_1, x_2, \dots, x_{d_G(C)} \rangle$$

If (i'), we let

$$(i.1) \tilde{C}_D(x_1, \dots, x_{d_G(C)}) \rightarrow \tilde{A}_D(\tilde{B}, x_1, x_2, \dots, x_{d_G(C)}) \in P$$

for each  $D \in N$ , and

$$(i.2) \tilde{C}_A(x_1, \dots, x_{d_G(C)}) \rightarrow A'(\tilde{B}, x_1, x_2, \dots, x_{d_G(C)}) \in P.$$

If (ii'), we let

$$(ii.1) \tilde{C}_D(x_1, \dots, x_{d_G(C)}) \rightarrow \tilde{A}_D(\bullet(\tilde{B}, x_1), x_2, \dots, x_{d_G(C)}) \in P$$

for each  $D \in N$ , and

$$(ii.2) \tilde{C}_A(x_1, \dots, x_{d_G(C)}) \rightarrow A'(\bullet(\tilde{B}, x_1), x_2, \dots, x_{d_G(C)}) \in P.$$

We will not provide a strictly formal proof, but briefly emphasize, why  $\Gamma$  does its job as desired: the way in which  $\Gamma$  simulates  $G$  crucially depends on the possibility to rewrite blindly a nonterminal  $B \in N$  by starting with rewriting  $B$  as any  $A \in N$  for which there is a terminating rule in  $R$  available, cf. (t.1). In terms of  $\Gamma$ , this  $A$  gets indexed by  $B$  storing the necessity that  $A$  has to be successively developed inside-out, finally creating a tree rooted in  $B$ . That is to say, in some later derivation step the blind rewriting of  $B$  simulated by  $\Gamma$  must get legitimated by an application of a rule of the sort (i.2), (ii.2) or (iii.2), in order to create a convergent derivation. This sort of simple information inheritance is possible because of the fact that the complex productions in  $G$ , those which are of rank 2, are still “simple enough,” i.e., the contribution of at least one of the nonterminals on the righthand side consists in a simple string of terminals.

**Proposition 33** *Each LCFRL<sub>1,2</sub> is, up to a homomorphism, the string language derivable by some linear CFTG in IO-mode.*  $\square$

An *indexed language (IL)* is the string language derived by an *indexed grammar (IG)* in the sense of Aho 1968.

**Corollary 34** *Each LCFRL<sub>1,2</sub> is an IL.*

*Proof.* By Proposition 33, because for linear CFTGs, IO- and OI-mode derive identical (string) languages (cf. Kepser and Mönnich forthcoming). Furthermore the class of string languages derived by CFTGs in OI-mode is included in the class of ILs (Fischer 1968, Rounds 1970a,b),<sup>12</sup> and the class of ILs is closed under homomorphisms (Aho 1968).  $\square$

**Proposition 35** *The class of LCFRL<sub>1,2</sub>s is properly included in the class of LCFRLs.*

*Proof.* By Corollary 34, because the LCFRL from our example above,  $L(G_{\text{ex}})$ , is known not to be an IL (Staudacher 1993).  $\square$

**Corollary 36** *The class of languages derivable by MG<sup>+SPIC</sup>s is properly included in the class of languages derivable by MGs.*  $\square$

## 10.5 Summary

We have shown that—in terms of derivable string languages—the type of a minimalist grammar (MG) as originally introduced in Stabler 1997 defines a proper superclass of the revised type of an MG and, therefore, the type of a strict MG both introduced in Stabler 1999, and known to be weakly equivalent. In order to achieve the result we have

---

<sup>12</sup>Note that, vice versa, for each IL  $L$ ,  $L \setminus \{\epsilon\}$  is the string language derived by some CFTG in OI-mode (cf. Fischer 1968, Rounds 1970a,b).



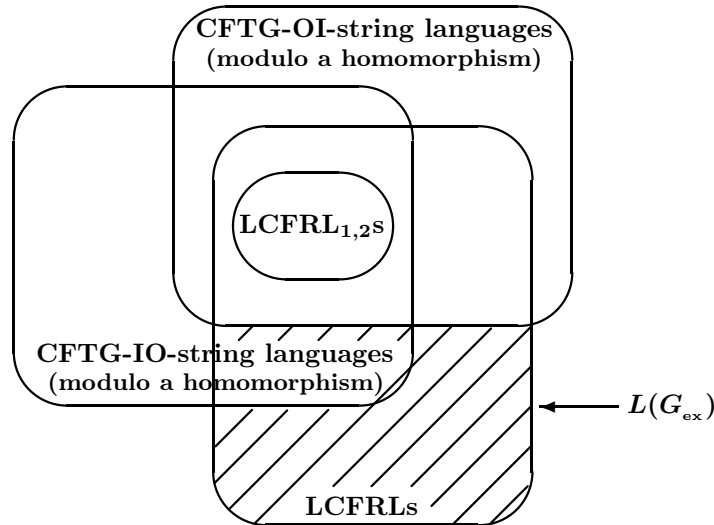


FIGURE 1: Proving the proper inclusion of  $LCFRL_{1,2}s$  within  $LCFRLs$ .

proven that the class of (*string based*) *linear context-free rewriting languages* ( $LCFRLs$ ) properly subsumes a particular subclass of  $LCFRLs$ , referred to as the class of  $LCFRL_{1,2}s$ . This was motivated by the fact that  $LCFRLs$  and  $LCFRL_{1,2}s$  coincide with the two classes of derivable string languages defined by the original and the revised MG-type, respectively.

The most crucial part of our proof consists in showing that every  $LCFRL_{1,2}$  is, modulo a homomorphism, the string language derivable in *inside-out* ( $IO$ ) mode by a linear *context-free tree grammar* ( $CFTG$ ). For linear  $CFTGs$  it holds that the class of languages derivable in  $IO$ -mode is not distinguishable from the class of languages derivable in *outside-in* ( $OI$ ) mode; and the class of string languages generally derivable by  $CFTGs$  in  $OI$ -mode is known to be subsumed by the class of languages derivable by *indexed grammars* ( $IGs$ ). Since the latter class of languages is closed under homomorphisms, the intended result finally followed from presenting an  $LCFRL$  known not to be derivable by an  $IG$ , namely, the language  $L(G_{ex})$  (cf. Figure 1).

## References

Aho, Alfred V. 1968. Indexed grammars—An extension of context-free grammars. *Journal of the Association for Computing Machinery* 15:647–671.

- de Groote, Philippe, Glyn Morrill, and Christian Retoré, eds. 2001. *Logical Aspects of Computational Linguistics (LACL '01)*, Lecture Notes in Artificial Intelligence Vol. 2099. Berlin, Heidelberg: Springer.
- Engelfriet, Joost and Erik M. Schmidt. 1977. IO and OI. I. *Journal of Computer and System Sciences* 15:328–353.
- Fischer, Michael J. 1968. Grammars with macro-like productions. In *Conference Record of 1968 Ninth Annual Symposium on Switching and Automata Theory*, Schenectady, NY, pages 131–142. IEEE.
- Harkema, Henk. 2001. A characterization of minimalist languages. In de Groote et al. (2001), pages 193–211.
- Kepser, Stephan and Uwe Mönnich. forthcoming. Closure properties of linear context-free tree languages with an application to optimality theory. *Theoretical Computer Science*. Preprint available at <http://tcl.sfs.uni-tuebingen.de/~kepser/papers/pubs.html>.
- Kobele, Gregory M. and Jens Michaelis. 2005. Two type-0 variants of minimalist grammars. In *FG-MoL 2005. The 10th conference on Formal Grammar and The 9th Meeting on Mathematics of Language*, Edinburgh. This volume.
- Koopman, Hilda and Anna Szabolcsi. 2000. *Verbal Complexes*. Cambridge, MA: MIT Press.
- Michaelis, Jens. 2001a. Derivational minimalism is mildly context-sensitive. In M. Moortgat, ed., *Logical Aspects of Computational Linguistics (LACL '98)*, Lecture Notes in Artificial Intelligence Vol. 2014, pages 179–198. Berlin, Heidelberg: Springer.
- Michaelis, Jens. 2001b. Transforming linear context-free rewriting systems into minimalist grammars. In de Groote et al. (2001), pages 228–244.
- Michaelis, Jens. 2002. Implications of a revised perspective on minimalist grammars. Draft, Potsdam University. Available at <http://www.ling.uni-potsdam.de/~michael/papers.html>.
- Michaelis, Jens. 2004. Observations on strict derivational minimalism. *Electronic Notes in Theoretical Computer Science* 53:192–209. Proceedings of the joint meeting of the 6th Conference on Formal Grammar and the 7th Meeting on Mathematics of Language (FGMOL '01), Helsinki, 2001.
- Rambow, Owen and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Theoretical Computer Science* 223:87–120.

- Rounds, William C. 1970a. Mappings and grammars on trees. *Mathematical Systems Theory* 4:257–287.
- Rounds, William C. 1970b. Tree-oriented proofs of some theorems on context-free and indexed languages. In *Proceedings of the 2nd Annual ACM Symposium on Theory of Computing*, Northhampton, MA, pages 109–116. ACM.
- Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88:191–229.
- Stabler, Edward P. 1997. Derivational minimalism. In C. Retoré, ed., *Logical Aspects of Computational Linguistics (LACL '96)*, Lecture Notes in Artificial Intelligence Vol. 1328, pages 68–95. Berlin, Heidelberg: Springer.
- Stabler, Edward P. 1999. Remnant movement and complexity. In G. Bouma, G.-J. M. Kruijff, E. Hinrichs, and R. T. Oehrle, eds., *Constraints and Resources in Natural Language Syntax and Semantics*, pages 299–326. Stanford, CA: CSLI Publications.
- Staudacher, Peter. 1993. New frontiers beyond context-freeness: DI-grammars and DI-automata. In *6th Conference of the European Chapter of the Association for Computational Linguistics (EACL '93)*, Utrecht, pages 358–367. ACL.
- Vijay-Shanker, K., David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics (ACL '87)*, Stanford, CA, pages 104–111. ACL.
- Weir, David J. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.



---

# Modular Grammar Design with Typed Parametric Principles

RALPH DEBUSMANN, DENYS DUCHIER AND ANDREAS  
ROSSBERG

## Abstract

This paper introduces a type system for *Extensible Dependency Grammar* (XDG) (Debusmann et al., 2004), a new, modular grammar formalism based on dependency grammar. As XDG is based on graph description, our emphasis is on capturing the notion of *multigraph*, a tuple of arbitrary many graphs sharing the same set of nodes. An XDG grammar consists of the stipulation of an extensible set of *parametric principles*, which yields a modular and compositional approach to grammar design.

**Keywords** GRAMMAR FORMALISMS, DEPENDENCY GRAMMAR, TYPE  
SYSTEMS

## 11.1 Introduction

*Extensible Dependency Grammar* (XDG) (Debusmann et al., 2004) is a general framework for dependency grammar, with multiple levels of linguistic representations called *dimensions*. Its approach, motivated by the dependency grammar paradigm (Tesnière, 1959, Mel'čuk, 1988), is articulated around a description language for multi-dimensional attributed labeled graphs. XDG is a generalization of *Topological Dependency Grammar* (TDG) (Duchier and Debusmann, 2001).

For XDG, a grammar is a constraint that describes the valid linguistic signs as  $n$ -dimensional attributed labeled graphs, i.e.  $n$ -tuples of graphs sharing the same set of attributed nodes, but having different sets of labeled edges. It is central to XDG that all aspects of these signs are

stipulated explicitly by *principles*: the class of models for each dimension, additional properties that they must satisfy, how one dimension must relate to another, and even lexicalization.

Yet, no formal account set in the XDG framework has so far explained what exactly these principles are, nor how they can be brought to bear on specific dimensions. In this paper, we show how an XDG grammar can be formally assembled from modular components called *parametric principles*. This yields a modular and compositional approach to grammar design. Compositional coherence is ensured by a *type system* whose primary novelty is to accommodate the notion of multi-dimensional graphs. Instantiation of parametric principles not only imposes grammatical constraints, but, through the type system, also determines the necessary structure of grammatical signs. In this perspective, a grammar framework is simply a library of parametric principles such as the one offered by the XDG *Development Kit* (XDK) (Debusmann and Duchier, 2004). The XDK is a freely available development environment for XDG grammars including a concurrent constraint parser written in the Mozart/Oz programming language (Mozart Consortium, 2005).

## 11.2 Extensible Dependency Grammar

We briefly illustrate the XDG approach with an example of the German subordinate sentence “(dass) einen Mann Maria zu lieben versucht”.<sup>1</sup> Figure 1 shows an analysis with two dimensions: ID models grammatical function and LP word-order using *topological fields* (Duchier and Debusmann, 2001, Gerdes and Kahane, 2001)

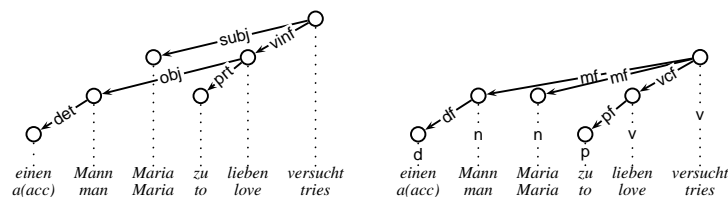


FIGURE 1: Example XDG analysis, ID left, LP right

Both dimensions share the same set of nodes (circles) but have different edges. On ID, *Maria* is subject (**subj**) of control verb *versucht*, and *Mann* object (**obj**) of *lieben*. On LP, *Mann* and *Maria* are both in the *Mittelfeld* (**mf**) of *versucht*, i.e. *Mann* has *climbed* to the finite verb.

<sup>1</sup>(that) Mary tries to love a man – see (Duchier and Debusmann, 2001)

The instance of XDG demonstrated above covers only syntax. However, XDG instances can use arbitrary many dimensions of representation, e.g. including semantics (including predicate-argument structure and scope relationships) (Debusmann et al., 2004), prosody and information structure (Debusmann et al., 2005).

In fact, XDG per se is not a grammar formalism but only a general graph description language for multi-dimensional attributed labeled graphs. In order to be used as a grammar formalism, it first needs to be instantiated using appropriate principles (see below). In this respect, it is similar to *Head-driven Phrase Structure Grammar (HPSG)* (Pollard and Sag, 1994), which per se is only a description language for typed feature structures (Carpenter, 1992), and becomes a grammar formalism only when it is instantiated, using appropriate types for the feature structures and the principles for e.g. head feature percolation and subcategorization.

In particular, XDG is more general than other dependency-based grammar formalisms like *Meaning Text Theory (MTT)* (Mel'čuk, 1988), *Functional Generative Description (FGD)* (Sgall et al., 1986), *Word Grammar (WG)* (Hudson, 1990) and *Free Order Dependency Grammar (FODG)* (Holan et al., 2000) since it can accommodate arbitrary many dimensions of representation and arbitrary principles to stipulate the well-formedness conditions. For instance, the dimensions of XDG need not be trees but can also be directed acyclic graphs (dags), e.g. to handle re-entrancies for the modeling of control constructions or relative clauses. In addition, the dimensions can be totally agnostic to word order. In our example above, the ID dimension did not state any word order requirements, only the LP dimension did.

### 11.3 Type System

**Formalization.** Let  $\mathcal{V}$  be an infinite set of node variables,  $\mathcal{L}$  a finite set of labels, and  $\mathcal{A}$  a set of attributes. An *attributed labeled graph*  $(V, E, A)$  consists of a finite set of nodes  $V \subseteq \mathcal{V}$ , a finite set of labeled directed edges  $E \subseteq V \times V \times \mathcal{L}$  between them, and an assignment  $A : V \rightarrow \mathcal{A}$  of attributes to nodes. An *n-dimensional attributed labeled graph*  $((V, E_1, A_1), \dots, (V, E_n, A_n))$ , or *multigraph*, is a *n*-tuple of labeled attributed graphs  $(V, E_i, A_i)$  over the same set of nodes  $V$ .

To provide a typed account of the XDG framework, we need a satisfactory type for graphs. A first idea is  $(\{\mathcal{V}\}, \{\mathcal{V} \times \mathcal{V} \times \mathcal{L}\}, \mathcal{V} \rightarrow \mathcal{A})$ , where we write  $\{\tau\}$  for *set of*  $\tau$ , but such a type is very imprecise: it fails to express that the nodes used in the edges (2nd arg) are elements of the graph's set of nodes (1st arg). Since additionally element

graphs of a multigraph must be defined over the *same* set of nodes, some form of dependent typing (Aspinall and Hofmann, 2005) appears inescapable. Given that general systems of dependent types tend to make type-checking undecidable, we sketch instead a stratified system, using a specialised notion of kinds (Pierce, 2002), that is sufficient for our purpose.

### 11.3.1 Type structure

We assume given a number of disjoint sets of symbols  $D_i$  called finite domain kinds. Among them, two are distinguished: **Nodes** and **Labels**. Much of our type and kinding systems are quite standard. For reasons of space, we only detail the parts which are original to our proposal, and write  $\tau :: \kappa$  and  $e : \tau$  for the judgments that type  $\tau$  has kind  $\kappa$  and expression  $e$  has type  $\tau$ , omitting kinding and typing contexts.

$\kappa ::=$	$\star$	top	$D_i \sqsubset \star$	$G_v \sqsubset \star$	$M_v \sqsubset \star$
	$D_i$	domain			
	$G_{c_1 \dots c_k}$	graph			
	$M_{c_1 \dots c_k}$	multigraph			
	$\dots$				
			$\frac{\tau :: \kappa \quad \kappa \sqsubset \kappa'}{\tau :: \kappa'}$		

FIGURE 2: Kinds and subkinds

$\tau, \tau_i ::=$	$c_1 \dots c_k$	domain	$[f_1 : \tau_1, \dots, f_n : \tau_n]$	record
	$\{\tau\}$	set	<b>graph</b> $\tau_1 \tau_2 \tau_3$	graph
	$(\tau_1, \dots, \tau_n)$	tuple	$\llbracket f_1 : \tau_1, \dots, f_n : \tau_n \rrbracket$	multigraph
	$\tau_1 \rightarrow \tau_2$	function	<b>grammar</b> $\tau$	grammar
	$!\tau$	singleton	$\dots$	

FIGURE 3: Types

**Finite domain sum types.** They are built from symbols drawn from finite domain kinds: we write  $c_1|\dots|c_k$  for a finite domain sum type in kind  $D_i$  and  $\epsilon_{D_i}$  for its empty sum. Here are their kinding (left) and typing (right) rules:

$$\frac{c_1, \dots, c_k \text{ symbols in } D_i}{c_1|\dots|c_k :: D_i} \quad \frac{c_1|\dots|c_k :: D_i}{c_i : c_1|\dots|c_k}$$

Kinds  $G_{c_1|\dots|c_k}$  and  $M_{c_1|\dots|c_k}$  indexed by finite domain sums make it possible to have types that depend on specific subsets of **Nodes** or **Labels**



without requiring more general term-dependent types.

**Singleton types.** If  $c_1 | \dots | c_k :: \text{Nodes}$  is the type of nodes of a graph, then  $\{c_1 | \dots | c_k\}$  is the type of a set of these nodes. This is not sufficiently precise to type the set of nodes of the graph because it doesn't express that the latter is a maximal set. To achieve this aim, we introduce a novel, specialised variation of *singleton types* (Aspinall, 1995, Stone and Harper, 2005). Unlike standard singleton types, our form does not refer to values, hence we avoid the need for dependent types. We write  $!(c_1 | \dots | c_k)$  for the type inhabited by the single value  $\{c_1, \dots, c_k\}$ . Here are the relevant kinding, typing, and subtyping rules:

$$\frac{\tau :: D_i}{!\tau :: \star} \quad \frac{!(c_1 | \dots | c_k) :: \star}{\{c_1, \dots, c_k\} : !(c_1 | \dots | c_k)} \quad !\tau \sqsubseteq \{\tau\}$$

**Graph types.** A graph is defined from finite domain types  $\nu$  and  $\ell$  for its nodes and labels, and a type  $a$  for its attributes. We write  $G_\nu$  for the kind of a graph over node type  $\nu$ . Kinding and typing rules are:

$$\frac{\nu :: \text{Nodes} \quad \ell :: \text{Labels} \quad a :: \star}{\text{graph } \nu \ell a :: G_\nu} \quad \frac{V : !\nu \quad E : \{(\nu, \nu, \ell)\} \quad A : \nu \rightarrow a}{\text{Graph } V E A : \text{graph } \nu \ell a}$$

The typing rule requires the node set to be assigned a singleton type, capturing the precise set of nodes in the type  $\nu$ . Typing thus precludes the set of edges mentioning invalid nodes. Note also that the syntax of graph kinds in Figure 3 requires  $\nu$  to be a concrete domain type.

**Multigraph types.** A multigraph is a record of graphs over the same finite domain  $\nu$  of nodes. We write  $M_\nu$  for its kind. Here are the kinding and typing rules:

$$\frac{g_1 :: G_\nu \quad \dots \quad g_n :: G_\nu}{\llbracket f_1 : g_1, \dots, f_n : g_n \rrbracket :: M_\nu}$$

$$\frac{G_1 : g_1 :: G_\nu \quad \dots \quad G_n : g_n :: G_\nu}{\llbracket f_1 = G_1, \dots, f_n = G_n \rrbracket : \llbracket f_1 : g_1, \dots, f_n : g_n \rrbracket :: M_\nu}$$

**Grammars.** An XDG *grammar* is a set of predicates over the same multigraph type:

$$\frac{\tau :: M_\nu}{\text{grammar } \tau :: \star} \quad \frac{\tau :: M_\nu \quad S : \{\tau \rightarrow \text{prop}\}}{\text{Grammar } S : \text{grammar } \tau}$$

Note that, in order to match our intuitions about grammars, the grammar type should be polymorphic in the finite domain type for nodes, otherwise the number of nodes is fixed. This can be achieved by extending the kinding system to admit *kind schemes*  $G_\delta$  and  $M_\delta$  where  $\delta$  is a domain variable. An XDG *framework* is a set, also called a library, of principle templates.

## 11.4 Typed Templates

Attributes are usually given in the form of attribute/value matrices and principles parametrized by values which can be found at specific feature paths. For example, on the syntactic dimension  $(V, E_{\text{ID}}, A_{\text{ID}})$  the agreement tuple assigned to each word must be one of those licensed by its lexical entry:<sup>2</sup>

$$\text{Agr } \llbracket \text{id} = \text{Graph } V \ E_{\text{ID}} \ A_{\text{ID}} \rrbracket = \quad \forall v \in V : A_{\text{ID}}(v).\text{agr} \in A_{\text{ID}}(v).\text{lex}.\text{agrs}$$

We can generalize this into a reusable principle by abstracting over feature paths using access functions:

$$\begin{aligned} \text{Elem } D \ F_1 \ F_2 \ M &= \quad \mathbf{let} \ \text{Graph } V \ E \ A = D(M) \\ &\quad \mathbf{in} \ \forall v \in V : F_1(A(v)) \in F_2(A(v)) \end{aligned}$$

but this is not very legible and, for notational convenience, we explore here an alternative that we call *templates*:

$$\text{Elem} \langle d, p_1, p_2 \rangle \llbracket d = \text{Graph } V \ E \ A \rrbracket = \quad \forall v \in V : A(v).p_1 \in A(v).p_2$$

where  $d, p_1, p_2$  are *feature path variables*. A feature path is a (possibly empty) sequence of features. We write  $\pi$  for a path,  $\epsilon$  for the empty path, and  $\pi_1\pi_2$  for the concatenation of  $\pi_1$  and  $\pi_2$ . It is possible to generalize the language by allowing feature paths or feature path variables in types and patterns (and by extension in record ‘dot’ access) where previously only features were allowed. The intuition of such an extension lies in the congruence  $[\epsilon : \tau] \equiv \tau$ ,  $[\pi_1\pi_2 : \tau] \equiv [\pi_1 : [\pi_2 : \tau]]$ , and in the interpretation of a dot access  $\cdot\pi$  as a postfix function of type  $[\pi : \tau] \rightarrow \tau$ .

Note that, if we write  $\mathbf{graph} \ \nu \ \ell \ a$  for the type of Elem’s argument graph, it is our intention that type inference should require  $a$  to match the pattern  $[p_1 : \tau, p_2 : \{\tau\}, \dots]$ . This can be achieved either with a type system supporting record polymorphism or by adopting an open-world semantics<sup>3</sup> for records and multigraphs, à la  $\psi$ -terms of LIFE. For simplicity, in this article we choose the latter.

We write  $\langle p_1, p_2 \rangle \rightsquigarrow \tau$  for the type of a template abstracting over feature path variables  $p_1$  and  $p_2$ ;  $t$  may contain occurrences of  $p_1$  and  $p_2$ . We write  $\tau_1 :: \kappa_1, \dots, \tau_n :: \kappa_n \Rightarrow \tau$  to express kinding constraints on the free type variables of  $\tau$ . The type of Elem is then:

$$\begin{aligned} \text{Elem} : \langle d, p_1, p_2 \rangle \rightsquigarrow \nu :: \text{Nodes}, \ell :: \text{Labels}, \tau :: \star \Rightarrow \\ \llbracket d : \mathbf{graph} \ \nu \ \ell \ [p_1 : \tau, p_2 : \{\tau\}] \rrbracket \rightarrow \text{prop} \end{aligned}$$

<sup>2</sup>We adopt a pattern matching notation with obvious meaning

<sup>3</sup>no closed arities

## 11.5 Parametric Principles

We now illustrate how typed templates are intended to be used to define *parametric principles*.<sup>4</sup> We write  $v \xrightarrow{l}_E v'$  for an edge  $(v, v', l) \in E$ ,  $v \rightarrow_E v'$  for one with any label,  $v \xrightarrow{+}_E v'$  for the transitive closure, and  $v \rightarrow_E D$  for the type-raised relation where  $D = \{v' \mid \forall v \rightarrow_E v'\}$ , etc. . .

**Tree Principle.** It stipulates the set  $L$  of edge labels and requires that 1) each node has at most one incoming edge, 2) there is precisely one node with no incoming edge (one root), and 3) there are no cycles:

$$\begin{aligned} \text{Tree}\langle d \rangle L \llbracket d = \text{Graph } V \ E \ A \rrbracket = \\ \forall v \in V \quad : \quad \forall M \subseteq V : M \rightarrow_E v \Rightarrow |M| \leq 1 \quad \wedge \\ \exists! v \in V \quad : \quad \forall M \subseteq V : M \rightarrow_E v \Rightarrow |M| = 0 \quad \wedge \\ \forall v \in V \quad : \quad \forall D \subseteq V : v \xrightarrow{+}_E D \Rightarrow v \notin D \end{aligned}$$

Here is the type constraint assigning singleton type  $!l$  to  $L$ :

$$\text{Tree} : \langle d \rangle \rightsquigarrow \nu :: \text{Nodes}, \ell :: \text{Labels} \Rightarrow !l \rightarrow \llbracket d : \text{graph } \nu \ \ell \ \_ \rrbracket \rightarrow \text{prop}$$

**Valency Principle.** Incoming and outgoing edges must comply with the *in* and *out* valencies which stipulate, for each label  $\ell \in L$ , how many edges labeled with  $\ell$  are licensed:

$$\begin{aligned} \text{Valency}\langle d, p_{\text{in}}, p_{\text{out}} \rangle L \llbracket d = \text{Graph } V \ E \ A \rrbracket = \\ \forall v \in V \forall \ell \in L \quad : \quad \forall M \subseteq V : M \xrightarrow{\ell}_E v \Rightarrow |M| \in A(v).p_{\text{in}}.\ell \quad \wedge \\ \forall v \in V \forall \ell \in L \quad : \quad \forall D \subseteq V : v \xrightarrow{\ell}_E D \Rightarrow |D| \in A(v).p_{\text{out}}.\ell \end{aligned}$$

Writing  $\mathbb{N}$  for the type of natural numbers, here is the type constraint:

$$\begin{aligned} \text{Valency} : \langle d, p_{\text{in}}, p_{\text{out}} \rangle \rightsquigarrow \nu :: \text{Nodes}, \ell :: \text{Labels} \Rightarrow \\ !l \rightarrow \llbracket d : \text{graph } \nu \ \ell \ [p_{\text{in}} : \ell \rightarrow \{\mathbb{N}\}, p_{\text{out}} : \ell \rightarrow \{\mathbb{N}\}] \rrbracket \rightarrow \text{prop} \end{aligned}$$

**Climbing Principle.** Originating from TDG, this principle expresses that the tree-shape on dimension  $d_1$  is a flattening of that of  $d_2$ : 1) the dominance relation on  $d_1$  must be a subset of that on  $d_2$ , 2) on  $d_1$ , each node must land on its  $d_2$ -mother or climb higher:

$$\begin{aligned} \text{Climbing}\langle d_1, d_2 \rangle \llbracket d_1 = \text{Graph } V \ E_1 \ A_1, d_2 = \text{Graph } V \ E_2 \ A_2 \rrbracket = \\ \forall v \in V \quad : \quad \forall D_1, D_2 \subseteq V : v \xrightarrow{+}_{E_1} D_1 \wedge v \xrightarrow{+}_{E_2} D_2 \Rightarrow D_1 \subseteq D_2 \quad \wedge \\ \forall U_1, U_2 \subseteq V : U_1 \xrightarrow{+}_{E_1} v \wedge U_2 \xrightarrow{*}_{E_1} \rightarrow_{E_2} v \Rightarrow U_1 \subseteq U_2 \\ \text{Climbing} : \langle d_1, d_2 \rangle \rightsquigarrow \nu :: \text{Nodes} \Rightarrow \\ \llbracket d_1 : \text{graph } \nu \ \_ \ \_, d_2 : \text{graph } \nu \ \_ \ \_ \rrbracket \rightarrow \text{prop} \end{aligned}$$

<sup>4</sup>For lack of space, we present only a few principles. For further information, the reader is referred to e.g. (Debusmann et al., 2004, Debusmann and Duchier, 2004).

**Lexicon Principle.** XDG grammars are typically *lexicalized*: the record assignments to nodes are then partially determined by a *lexicon Lex*. The lexicon principle stipulates that each node must be assigned a *lexical entry*:

$$\text{Lexicon}\langle d \rangle \text{ Lex} \llbracket d = \text{Graph } V \ E \ A \rrbracket = \forall v \in V : A(v) \in \text{Lex}$$

Here is the corresponding type constraint, stipulating that the graph on dimension  $d$  has no edges (since this dimension has only the purpose to carry the lexical entries):

$$\begin{aligned} \text{Lexicon} : \langle d \rangle \rightsquigarrow \nu :: \text{Nodes}, \tau :: \star \Rightarrow \\ \{\tau\} \rightarrow \llbracket d : \text{graph } \nu \ \epsilon_{\text{Labels}} \ \tau \rrbracket \rightarrow \text{prop} \end{aligned}$$

**Lookup Principle.** A lexical entry is normally a record having a feature for each dimension. The *lookup* principle looks up a lexical entry's subrecord for a particular dimension and equates it with the  $p_{\text{lex}}$  feature of the node's attributes:

$$\begin{aligned} \text{Lookup}\langle d_1, d_2, p_{\text{lex}} \rangle \llbracket d_1 = \text{Graph } V \ E_1 \ A_1, d_2 = \text{Graph } V \ E_2 \ A_2 \rrbracket = \\ \forall v \in V : A_1(v).p_{\text{lex}} = A_2(v).d_1 \end{aligned}$$

$$\begin{aligned} \text{Lookup} : \langle d_1, d_2, p_{\text{lex}} \rangle \rightsquigarrow \nu :: \text{Nodes}, \tau :: \star \Rightarrow \\ \llbracket d_1 : \text{graph } \nu \ \_ \ [p_{\text{lex}} : \tau], d_2 : \text{graph } \nu \ \_ \ [d_1 : \tau] \rrbracket \rightarrow \text{prop} \end{aligned}$$

## 11.6 Example ID/LP Grammar

We now describe how the grammar of (Duchier and Debusmann, 2001) can be assembled in our typed framework of parametric principles. To better illustrate the compositionality of our approach, we adopt an incremental presentation that derives more complex grammars from simpler ones through operations of composition and restriction:

$$\begin{aligned} (\text{Grammar } S_1) ++ (\text{Grammar } S_2) &= \text{Grammar } (S_1 \cup S_2) \\ (\text{Grammar } S_1) // S_2 &= \text{Grammar } (S_1 \cup S_2) \end{aligned}$$

The grammar requires 3 dimensions: ID for syntax, LP for topology, and LEX for parametrization by a lexicon. They are respectively characterized by the following sets of labels:

$$\begin{aligned} L_{\text{ID}} &= \{\text{det, subj, obj, vbse, vppt, vinf, prt}\} \\ L_{\text{LP}} &= \{\text{d, df, n, mf, vcf, p, pf, v, vxf}\} \\ L_{\text{LEX}} &= \emptyset \end{aligned}$$

Figure 4 shows how grammars for the 3 individual dimensions ( $G_{\text{ID}}$ ,  $G_{\text{LP}}$ ,  $G_{\text{LEX}}$ ) are stipulated by instantiation of principles. For example, for  $G_{\text{ID}}$ , instantiation of  $\text{Tree}\langle \text{id} \rangle$   $L_{\text{ID}}$  has three consequences: (1) signs are required to match  $\llbracket \text{id} = X \rrbracket$ , i.e. to have an id dimension, (2) the

directed graph on that dimension must have labels in  $L_{ID}$ , (3) this graph must be a tree.  $G_{ID+LEX}$  is a grammar where the ID dimension is restricted by lexical constraints from the LEX dimension: it is obtained by combining  $G_{ID}$  and  $G_{LEX}$  using the composition operator  $++$ , and connecting them by the  $\text{Lookup}\langle id, lex \rangle$  principle using the restriction operator  $//$ . Similarly for  $G_{LP+LEX}$ . Finally the full grammar  $G_{ID/LP}$  is obtained by combining  $G_{ID+LEX}$  and  $G_{LP+LEX}$  and mutually constraining them by the Climbing and Barriers principles.

$$\begin{aligned}
G_{LEX} &= \text{Grammar } \{\text{Lexicon}\langle lex \rangle \text{ Lex}\} \\
G_{ID} &= \text{Grammar } \{\text{Tree}\langle id \rangle L_{ID}, \text{Valency}\langle id, in, out \rangle L_{ID}\} \\
G_{ID+LEX} &= (G_{ID} ++ G_{LEX}) // \{\text{Lookup}\langle id, lex, lex \rangle\} \\
G_{LP} &= \text{Grammar } \{\text{Tree}\langle lp \rangle L_{LP}, \text{Valency}\langle lp, in, out \rangle L_{LP}, \\
&\quad \text{Order}\langle lp, on, order, pos, self \rangle, \\
&\quad \text{Projectivity}\langle lp, pos \rangle\} \\
G_{LP+LEX} &= (G_{LP} ++ G_{LEX}) // \{\text{Lookup}\langle lp, lex, lex \rangle\} \\
G_{ID/LP} &= (G_{ID+LEX} ++ G_{LP+LEX}) // \\
&\quad \{\text{Climbing}\langle lp, id \rangle, \text{Barriers}\langle lp, id, blocks \rangle\}
\end{aligned}$$

FIGURE 4: Defining grammars in an incremental and compositional way

## 11.7 Conclusion

In this paper, we have made a threefold contribution: (1) we described a novel system of restricted dependent types capable of precisely describing graphs and multi-dimensional graphs, (2) we introduced a notion of typed templates with which we could express parametric principles, (3) we showed how these could enable a modular and compositional approach to grammar design. Finally we illustrated our proposal with an example reconstruction of an earlier grammar.

## References

- Aspinall, David. 1995. Subtyping with singleton types. In *Computer Science Logic*, vol. 933 of *Lecture Notes in Computer Science*. Springer-Verlag.
- Aspinall, David and Martin Hofmann. 2005. Dependent types. In B. Pierce, ed., *Advanced Topics in Types and Programming Languages*, chap. I.2, pages 45–136. The MIT Press.
- Carpenter, Bob. 1992. *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 32nd edn.

- Debusmann, Ralph and Denys Duchier. 2004. XDG Development Kit. <http://www.ps.uni-sb.de/~rade/mogul/publish/doc/debusmann-xdk/>.
- Debusmann, Ralph, Denys Duchier, Alexander Koller, Marco Kuhlmann, Gert Smolka, and Stefan Thater. 2004. A Relational Syntax-Semantics Interface Based on Dependency Grammar. In *Proceedings of COLING 2004*. Geneva/CH.
- Debusmann, Ralph, Oana Postolache, and Maarika Traat. 2005. A Modular Account of Information Structure in Extensible Dependency Grammar. In *Proceedings of the CICLING 2005 Conference*. Mexico City/MEX: Springer.
- Duchier, Denys and Ralph Debusmann. 2001. Topological Dependency Trees: A Constraint-Based Account of Linear Precedence. In *Proceedings of ACL 2001*. Toulouse/FR.
- Gerdes, Kim and Sylvain Kahane. 2001. Word Order in German: A Formal Dependency Grammar Using a Topological Hierarchy. In *ACL 2001 Proceedings*. Toulouse/FR.
- Holan, Tomas, Vladislav Kubon, Karel Oliva, and Martin Platek. 2000. On Complexity of Word-Order. *Journal t.a.l.* pages 273–301.
- Hudson, Richard A. 1990. *English Word Grammar*. Oxford/UK: B. Blackwell.
- Mel’cuk, Igor. 1988. *Dependency Syntax: Theory and Practice*. Albany/US: State Univ. Press of New York.
- Mozart Consortium. 2005. The Mozart-Oz Website. <http://www.mozart-oz.org/>.
- Pierce, Benjamin. 2002. *Types and Programming Languages*. The MIT Press.
- Pollard, Carl and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago/US: University of Chicago Press.
- Sgall, Petr, Eva Hajicova, and Jarmila Panevova. 1986. *The Meaning of the Sentence in its Semantic and Pragmatic Aspects*. Dordrecht/NL: D. Reidel.
- Stone, Christopher and Robert Harper. 2005. Extensional equivalence and singleton types. *Transactions on Computational Logic* to appear.
- Tesniere, Lucien. 1959. *Elements de Syntaxe Structurale*. Paris/FR: Klincksieck.

---

## What feature co-occurrence restrictions have to do with type signatures

WIEBKE PETERSEN AND JAMES KILBURY <sup>†</sup>

### Abstract

Computational linguistics of the last quarter century has seen the development of a hierarchical and lexicalist treatment of linguistic information. In the course of this development, formal constraints which were stated in GPSG in terms of feature co-occurrence restrictions came to be formulated within HPSG in terms of type hierarchies, but the relations between these descriptive devices has received little attention. Formal Concept Analysis now provides a framework within which these relations can be made explicit.

**Keywords** GPSG, HPSG, FEATURE CO-OCCURRENCE RESTRICTION, TYPE SIGNATURE, FORMAL CONCEPT ANALYSIS

### 12.1 Introduction

A rapid and remarkable development took place within computational linguistics in the years immediately following the introduction of unification-based models of language, in particular *Lexical Functional Grammar* (LFG) and *Generalized Phrase Structure Grammar* (GPSG), which employ feature structures to represent linguistic information. By the end of the 1980s a consensus had emerged, according to which the lexicon, which pairs word forms with feature structures, constitutes the

---

<sup>†</sup>We wish to thank two anonymous reviewers for their comments on an earlier version of this paper. The work was carried out in the project “Modelling Sub-regularity in the Lexicon” (University of Düsseldorf, Sonderforschungsbereich 282 “Theory of the Lexicon”, funded by the Deutsche Forschungsgesellschaft).

main repository of information in a language. Furthermore, hierarchical structuring had come to be viewed as an essential aspect or perhaps even the most salient characteristic of the lexicon.

GPSG, as conceived in Gazdar and Pullum (1982) and even in Gazdar et al. (1985), still largely represents the older, dichotomous view of grammar versus lexicon. Here major aspects of linguistic structure were encoded in *syntactic* rules, many of which later came to be regarded as stating the possible complement structures of verbs, i.e. *lexical* information. While the question “How is a classification imposed on the content of the lexicon by the system of features” is raised (Gazdar et al., 1985, p. 13), the answer of GPSG does not explicitly model the hierarchical inheritance relations inherent in lexical classifications. Rather, these relations are captured in *logical constraints on feature structures* in the form of *feature co-occurrence restrictions* (FCRs) and *feature specification defaults* (FSDs), the latter of which are nonmonotonic.

GPSG uses FCRs to restrict the distribution of features and their values. A pair consisting of a feature and a feature value is called a *feature specification*. Whereas GPSG features are atomic symbols, feature values are either atomic symbols or categories,<sup>1</sup> i.e., sets of feature specifications (Gazdar et al., 1985, p. 22). FCRs are part of a grammatical theory and restrict the set of possible categories and their extensions in the theory. A typical FCR is  $[+INV] \supset [+AUX, FIN]$  (Gazdar et al., 1985, p. 28), which is  $[<INV,+>] \supset [<AUX,+>, <VFORM,FIN>]$  when written out fully.<sup>2</sup> The condition stated here is that in English the feature specification  $<INV,+>$  implies  $<AUX,+>$  and  $<VFORM,FIN>$ : if a verb occurs initially in a sentence containing a subject, then this verb must be a finite auxiliary.

From the start the *lexicalist* orientation was prominent in LFG (cf. Bresnan 1982, therein Kaplan and Bresnan 1982) and reached a peak in the radical lexicalism of Karttunen (1986), which uses the framework of categorial grammar to shift the entirety of linguistic description to the lexicon. The move toward the lexicalist view was independent of *hierarchical* modelling, which emerged in other work. In particular, Flickinger (1987) pioneered the explicit description of relations between English verb classes in terms of inheritance hierarchies. On a separate front, de Smedt (1984) initiated the use of inheritance-based representation formalisms to capture the structure of inflectional classes. Practical advantages of hierarchical lexica quickly

<sup>1</sup>Categories of GPSG correspond to the untyped feature structures of other unification-based formalisms.

<sup>2</sup>The feature specification  $<INV,+>$  marks sentence-initial verbs,  $<AUX,+>$  marks auxiliary verbs, and  $<VFORM,FIN>$  specifies that the verb is finite.



became apparent and include the economic representation, integrity, homogeneity, and updating of data. The grammar formalism PATR-II (cf. Shieber et al., 1983) provides *templates* as an indispensable formal device for stating inheritance relations in an inheritance hierarchy, the consequences of which are clear to the authors:

But our notation does not only allow convenient abbreviations; it also plays an important role in the linguist's use of the formalism. [...] perhaps most importantly, grammar writers can use the notational tools to express generalizations they could not state in the "pure" unification notation of the formalism. (Shieber et al., 1983, p. 62)

*Head-driven Phrase Structure Grammar* (HPSG), as presented in Pollard and Sag (1987), carries over much of GPSG but restructures the model in terms of the hierarchical lexicalist framework and, in particular, encodes as lexical much of the information that GPSG represented with syntactic rules. The introduction of the *sign* as a uniform data structure for the representation of both lexical and phrasal information, together with the integration of all linguistic levels within the sign, provides the last means needed in order to state all information about a language within an inheritance hierarchy defining relations between signs. In contrast to the *nonmonotonic* inheritance hierarchies of de Smedt, Flickinger, and others, which allow exceptions and defaults, HPSG employs *monotonic* inheritance. The distinction will play no role in the rest of this paper.

Crucially, HPSG adopts no obvious counterpart for the FCRs of GPSG, although the *conditional feature structures* (Pollard and Sag, 1987, p. 43) of HPSG could have been employed for this purpose. Instead, the HPSG strategy for avoiding lexical redundancy lies in the use of inheritance hierarchies: "Structuring the lexicon in terms of an inheritance hierarchy of types has made it possible to factor out information common to many lexical entries, thereby greatly reducing lexical redundancy" (Sag and Wasow, 1999, p. 202). The informational domain consists of typed feature structures. The types serve two functions: On the one hand they allow access to embedded feature structures appearing as values of features; this permits the formulation of generalizations about such substructures. On the other hand, the types bear appropriateness conditions which restrict the set of feature structures of this type; such statements are feature-type pairs. By ordering the types in an inheritance hierarchy, the so-called *type signature*, in which appropriateness conditions are inherited, further redundancies are avoided.

It was clear to linguists that HPSG had replaced the FCRs of GPSG with inheritance hierarchies of types, but the relations between these

TABLE 1: Lexemes classified with respect to their feature specifications in Gazdar et al. (1985)

	v:+	v:-	n:+	n:-	vform:bse	vform:fn	vform:pas	aux:+	aux:-	inv:+	inv:-	nform:nom	nform:it	v:VAL	n:VAL	vform:VAL	aux:VAL	inv:VAL	nform:VAL	pform:with	pform:VAL
sing	x			x	x				x		x			x	x	x	x	x			
sings	x			x		x			x		x			x	x	x	x	x			
sung	x			x			x		x		x			x	x	x	x	x			
can1	x			x		x		x		x				x	x	x	x	x			
can2	x			x		x		x		x				x	x	x	x	x			
can3	x			x	x			x		x				x	x	x	x	x			
child		x	x									x		x	x					x	
it		x	x										x	x	x					x	
little	x		x											x	x						
with		x		x										x	x					x	x

formal devices were misunderstood and hardly questioned. Clearly, the devices had to be related in some way, but no formal framework was available within which the relations could be made explicit. Gerdemann and King (1994, 1993) present a procedural method for transforming a type signature so that it expresses an FCR. With Formal Concept Analysis (FCA, cf. Ganter and Wille 1999) a general framework is now available which allows the equivalence of the devices to be explained in a transparent and declarative fashion, which we shall do after briefly introducing the FCA framework itself.

## 12.2 Basics of Formal Concept Analysis

FCA is a mathematical theory designed for data analysis. FCA starts with the definition of a *formal context*  $K$  as a triple  $(G, M, I)$  consisting of a set of *objects*  $G$ , a set of *attributes*  $M$ , and a binary *incidence relation*  $I \subseteq G \times M$  between the two sets. Table 1 shows a formal context in the form of a *cross table*.<sup>3</sup> FCA associates with each formal context a lattice of formal concepts. A *formal concept* of a context is a pair consisting of a set of objects, its *extent*, and a set of attributes, its *intent*. The extent consists exactly of the objects in the context for which all the attributes of the intent apply; the intent consists correspondingly of all attributes of the context which the objects from the extent have in common. In order to formally express the strong connection between the extent and the intent of a formal concept given by the binary relation  $I$ , two derivational operators are defined between

<sup>3</sup>The content of this formal context will be explained in Section 12.3.

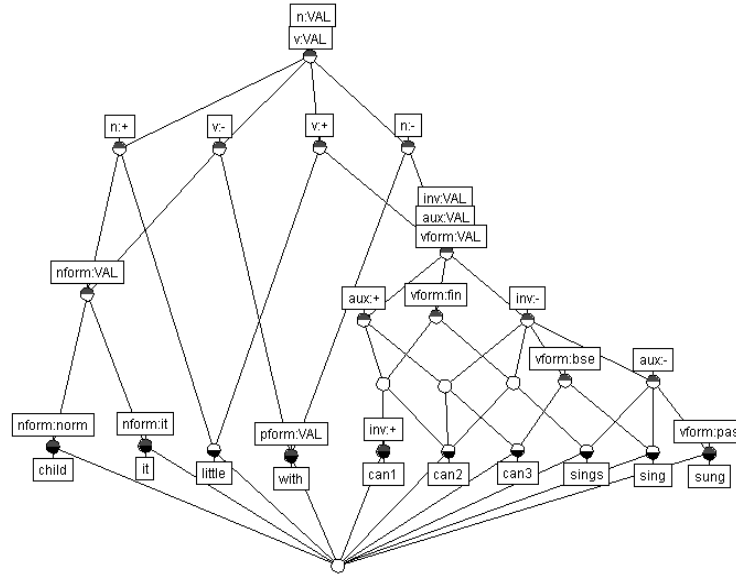


FIGURE 1: The concept lattice for the context of Table 1, which can be regarded as a type signature

the set of objects  $G$  and the attribute set  $M$  of a formal context: If  $A \subseteq G$  is a set of objects, then the set of the common attributes of  $A$  is  $A' := \{m \in M \mid \forall g \in A: (g, m) \in I\}$ , and if, analogously,  $B \subseteq M$  is a set of attributes, then the set of objects that have  $B$  in common is  $B' := \{g \in G \mid \forall m \in B: (g, m) \in I\}$ . A formal concept is thus a pair  $(A, B) \subseteq G \times M$  with  $A = B'$  and  $B = A'$ . Furthermore, the definition of the derivational operators guarantees that the set of all formal concepts of a formal context  $(G, M, I)$  equals  $\{(A'', A') \mid A \subseteq G\}$  and  $\{(B', B'') \mid B \subseteq M\}$ .<sup>4</sup>

The subconcept-superconcept relation on the set of all formal concepts of a context defines a partial order:

$$(A_1, B_1) \leq (A_2, B_2) \Leftrightarrow A_1 \subseteq A_2 \Leftrightarrow B_1 \supseteq B_2$$

This order relation corresponds to our intuitive notion of super- and subconcepts. Superconcepts are more general, encompass more objects, and are characterized by fewer attributes. The main theorem of FCA

<sup>4</sup>A detailed introduction to Formal Concept Analysis is given by Ganter and Wille (1999).

states that the set of all formal concepts of a formal context  $(G, M, I)$  ordered with respect to the subconcept-superconcept relation constitutes a complete lattice, which is called the *formal concept lattice* of  $(G, M, I)$ . Figure 1 shows the concept lattice corresponding to the formal context in Table 1. Formal concept lattices visualize structural connections between the data of a flat data table. Ganter and Wille (1998) stress:

It is our belief that the potential of Formal Concept Analysis as a branch of Applied Mathematics is just beginning to show. A typical task that concept lattices are useful for is to unfold given data, making their conceptual structure visible and accessible, in order to find patterns, regularities, exceptions, etc.

The definition of concept lattices allows an especially economical labelling of their Hasse diagrams (see Figure 1): instead of labelling every concept with its complete extent and intent, only the object and attribute concepts are labelled. The *object concept* of an object  $g$  is the smallest concept whose extent includes  $g$ , i.e.  $(g'', g')$ . The *attribute concept* of an attribute  $m$  is analogously the largest concept whose intent includes  $m$ , i.e.  $(m', m'')$ . In this way a concept lattice becomes an inheritance hierarchy. Any concept of the lattice inherits all objects which are labelled with subconcepts as its extent, and it inherits all attributes with which superconcepts are labelled as its intent. Inheritance hierarchies based on concept lattices are completely nonredundant, i.e., each attribute and each object appears exactly once in the hierarchy.

Besides the tabular form (as a formal context) and the hierarchical form (as a formal concept lattice) FCA provides a third device, namely attribute implications, to represent the mutual dependencies of the data to be analyzed. An *attribute implication* of a formal context is an implication of the form  $\mu \rightarrow \nu$ , where  $\mu$  and  $\nu$  are subsets of the attribute set ( $\mu$  is called the premise and  $\nu$  the conclusion of the implication). An attribute implication  $\mu \rightarrow \nu$  is *valid* in a context if and only if every object which has all attributes in  $\mu$  also has all attributes in  $\nu$ ; this is equivalent to the condition  $\nu \subseteq \mu''$ . The implications can be read off directly from the concept lattice: An implication  $m_1 \wedge m_2 \wedge \dots \wedge m_k \rightarrow m$  holds exactly when the greatest lower bound of the attribute concepts of  $m_1, \dots, m_k$  is a subconcept of the attribute concept of  $m$ . A set  $\mathcal{I}$  of attribute implications of a context  $K$  which is complete<sup>5</sup> and non-redundant<sup>6</sup> is called an *implication basis* of the context. Since for every implication basis of a formal context the union

---

<sup>5</sup>Every valid attribute implication of  $K$  can be derived from the members of  $\mathcal{I}$ .

<sup>6</sup>No real subset of  $\mathcal{I}$  forms a complete set of attribute implications of  $K$ .

TABLE 2: Basis of attribute implications of the formal context in Table 1

$\emptyset$	$\rightarrow$	$\{v : \text{VAL}, n : \text{VAL}\}$	(12.27)
$\{v : -, n : -\}$	$\leftrightarrow$	$\{\text{pform} : \text{VAL}\}$	(12.28)
$\{v : -, n : -\}$	$\leftrightarrow$	$\{\text{pform} : \text{with}\}$	(12.29)
$\{v : -, n : +\}$	$\leftrightarrow$	$\{\text{nform} : \text{VAL}\}$	(12.30)
$\{v : +, n : -\}$	$\leftrightarrow$	$\{\text{vform} : \text{VAL}\}$	(12.31)
$\{v : +, n : -\}$	$\leftrightarrow$	$\{\text{aux} : \text{VAL}\}$	(12.32)
$\{v : +, n : -\}$	$\leftrightarrow$	$\{\text{inv} : \text{VAL}\}$	(12.33)
$\{\text{inv} : +\}$	$\rightarrow$	$\{\text{vform} : \text{fin}, \text{aux} : +\}$	(12.34)
$\{\text{aux} : -\}$	$\rightarrow$	$\{\text{inv} : -\}$	(12.35)
$\{\text{vform} : \text{pas}\}$	$\rightarrow$	$\{\text{aux} : -\}$	(12.36)
$\{\text{vform} : \text{bse}\}$	$\rightarrow$	$\{\text{inv} : -\}$	(12.37)
$\{\text{nform} : \text{it}\}$	$\rightarrow$	$\{\text{nform} : \text{VAL}\}$	(12.38)
$\{\text{nform} : \text{nor}\}$	$\rightarrow$	$\{\text{nform} : \text{VAL}\}$	(12.39)
$\{\text{inv} : -\}$	$\rightarrow$	$\{\text{inv} : \text{VAL}\}$	(12.40)
$\{\text{aux} : +\}$	$\rightarrow$	$\{\text{aux} : \text{VAL}\}$	(12.41)
$\{\text{vform} : \text{fin}\}$	$\rightarrow$	$\{\text{vform} : \text{VAL}\}$	(12.42)
$\{n : +, n : -\}$	$\rightarrow$	$\perp$	(12.43)
$\{v : +, v : -\}$	$\rightarrow$	$\perp$	
$\vdots$		$\vdots$	
$\{\text{vform} : \text{bse}, \text{vform} : \text{pas}\}$	$\rightarrow$	$\perp$	

of every premise and the maximal corresponding conclusion forms an intent of one of the concepts of the context, the structure of the concept lattice is determined by the set of valid attribute implications up to isomorphism. Table 2 shows a basis of attribute implications of the context from Table 1.<sup>7</sup> Some abbreviations have been used in the table:  $A \leftrightarrow B$  denotes the two implications  $A \rightarrow B$  and  $B \rightarrow A$  and  $\perp$  is the (inconsistent) set of all attributes of the context of Table 1.

### 12.3 FCA relates FCRs and type signatures

The main idea for showing the convertability of FCRs and type signatures is to construct a suitable formal context in order to employ the methods of FCA. Figure 2 shows a lexical fragment consisting of 10 lex-

<sup>7</sup>Bases of formal contexts can be efficiently calculated with the help of the program *ConImp* ([http://www.mathematik.tu-darmstadt.de/ags/ag1/Software/DOS-Programme/Welcome\\_de.html](http://www.mathematik.tu-darmstadt.de/ags/ag1/Software/DOS-Programme/Welcome_de.html)).

sing :	$\begin{bmatrix} v : + \\ n : - \\ vform : bse \\ aux : - \\ inv : - \end{bmatrix}$	sings :	$\begin{bmatrix} v : + \\ n : - \\ vform : fin \\ aux : - \\ inv : - \end{bmatrix}$	sung :	$\begin{bmatrix} v : + \\ n : - \\ vform : pas \\ aux : - \\ inv : - \end{bmatrix}$
can :	$\begin{bmatrix} v : + \\ n : - \\ vform : fin \\ aux : + \\ inv : + \end{bmatrix}$	can :	$\begin{bmatrix} v : + \\ n : - \\ vform : fin \\ aux : + \\ inv : - \end{bmatrix}$	can :	$\begin{bmatrix} v : + \\ n : - \\ vform : bse \\ aux : + \\ inv : - \end{bmatrix}$
child :	$\begin{bmatrix} v : - \\ n : + \\ nform : norm \end{bmatrix}$	it :	$\begin{bmatrix} v : - \\ n : + \\ nform : it \end{bmatrix}$	little :	$\begin{bmatrix} v : + \\ n : + \end{bmatrix}$
with :	$\begin{bmatrix} v : - \\ n : - \\ pform : with \end{bmatrix}$				

FIGURE 2: Lexical fragment classified with respect to some features of Gazdar et al. (1985)

emes classified with respect to some of the features proposed in Gazdar et al. (1985).<sup>8</sup> The chosen features are exactly those which play a role in the first 4 FCRs given in Gazdar et al. (1985); these FCRs regulate the distribution of the features  $v$ ,  $n$ ,  $vform$ ,  $nform$ ,  $pform$ ,  $inv$ ,  $aux$ , and their values (see Table 3). The formal context in Table 1 is formed by taking each feature-value pair as an independent attribute of the context. GPSG also allows FCRs of the form  $[VFORM] \supset [+V, -N]$  (see FCR 2), which encode not only restrictions on feature-value pairs but also on the admissibility of certain features in the first place. Because of this, further attributes of the form  $feature:VAL$  have been added in Table 1, where such an attribute applies to a word if there is some value  $value$  such that  $feature:value$  is a feature specification of the word. Feature structures with embedded structures can be represented in a formal context by *flattening* them and viewing the path-value pairs as attributes (cf. Sporleder, 2003, Petersen, 2004). We call such formal contexts obtained from feature structures *feature-structure contexts*.

To illustrate the relationship between FCRs and type signatures we

<sup>8</sup>The feature  $vform$  distinguishes parts of the verb paradigm (*bse*, base-form; *fin*, finite; *pas*, passive participle) and  $nform$  analogously distinguishes the special expletive pronoun *it* from normal nouns (*norm*).

TABLE 3: The first 4 FCRs from Gazdar et al. (1985)

FCR 1 :	[+INV]	⊃	[+AUX, FIN]
FCR 2 :	[VFORM]	⊃	[+V, -N]
FCR 3 :	[NFORM]	⊃	[-V, +N]
FCR 4 :	[PFORM]	⊃	[-V, -N]

will proceed as follows: First we show how a type signature can be derived from a feature structure context, and then we do the same for a system of FCRs. Finally, we demonstrate how a feature structure context can be constructed from either a type signature or a set of FCRs.

Figure 1 shows the concept lattice for the context in Table 1, which can be directly interpreted as a type signature of HPSG if, first, a unique type is assigned to each node of the lattice<sup>9</sup> and if, second, an additional type *VAL* with subtypes *+*, *-*, *it*, *norm*, *with*, *fin*, *bse*, and *pas* is added.<sup>10</sup> The feature labels then encode the appropriateness conditions associated with each type, and the subconcept relation corresponds to the subtype relation in the type signature. If one reads the hierarchy in Figure 1 as a type signature one sees that it makes extensive use of multiple inheritance. Due to the definition of formal concepts, however, it can never be the case that a type inherits incompatible information from its upper neighbors.<sup>11</sup>

As noted before, type signatures fulfill two tasks: they avoid redundancies by structuring the information in an inheritance hierarchy and they restrict the set of permissible feature structures. As Gerdemann and King (1993, 1994) note, feature structures must be well-typed and sort-resolved in order to be total models of linguistic objects and furthermore, type systems must be closed-world systems. In order to show that our type signature successfully restricts the set of permissible feature structures, we have to demonstrate that, on the one hand, no inadmissible structure will be sort-resolved and well-typed with respect to the type signature and, on the other hand, every permissible struc-

<sup>9</sup>The lowest node is assigned the type  $\perp$  for *bottom* and the highest gets  $\top$  for *top*. The type  $\perp$  expresses contradiction and ensures that unification never fails.

<sup>10</sup>Normally, one would add extra types *boolean*, *vform*, *pform*, and *nform* between *VAL* and its subtypes in order to explicitly state the value ranges of the attributes. For further details on the automatic construction of type signatures from formal concept lattices see Petersen (2004).

<sup>11</sup>The principle of *unique feature introduction* is never violated because, by adding attributes of the form *attribute:VAL* to the formal context, the features are introduced on their own, unique attribute concepts.

ture is well-typed and sort-resolved. The former follows from the fact that the minimal concept of a concept lattice derived from a feature-structure context never represents an object of the context, since the values of an attribute mutually exclude each other; hence, the extent of the minimal concept of such a context is always empty. It follows that every *atom* (i.e. direct upper neighbor of the minimal concept) is an object concept and thus represents a permissible feature structure. The latter follows from the fact that all object concepts of the linguistic objects of Table 1 are atoms of the lattice. This property is not a necessary consequence of using a feature-structure context; it rather follows from the principle of choosing sort-resolved feature structures as total models of linguistic objects. This principle would be violated, e.g., if the example data were classified by attributes like *non-auxiliary verb*, *auxiliary verb*, and *inverted auxiliary verb*. In that case, the inverted auxiliary verb *can1* would belong to a subclass of the class of non-inverted auxiliary verbs like *can2*. By introducing the Boolean-valued attribute *inv*, the property of being non-inverted is marked explicitly and each linguistic object is described by an atom of the concept lattice.

The hierarchy in Figure 1 encodes so much information about the distribution of atomic values that it suffices to pair phonological forms with types in the lexicon in order to obtain adequate feature structures. For realistic lexica such a procedure leads to enormous type signatures since every lexical feature structure must have its own type.<sup>12</sup> Moreover, it conflicts with the idea of types when they, e.g. in the case of *with*, cover only a single object. On this issue Pollard and Sag (1987, p. 192) state:

[...] lexical information is organized on the basis of relatively few — perhaps several dozen — word types arranged in cross-cutting hierarchies which serve to classify all words on the basis of shared syntactic, semantic, and morphological properties. By factoring out information about words which can be predicted from their membership in types (whose properties can be stated in a single place once and for all), the amount of idiosyncratic information that needs to be stipulated in individual lexical signs is dramatically reduced.

Having constructed type signatures from formal concept lattices, we will now show how to derive FCRs from a feature-structure context as given in Table 1. The FCRs of GPSG are nothing other than implications that are compatible with the data of Table 1. In order to restrict the set of admissible categories sufficiently, the FCRs must reflect the mutual dependencies between the data of the context. Hence, the data

---

<sup>12</sup>Petersen (2004) presents a *folding strategy* to reduce the number of types.



of the context must respect the FCRs and the object concepts of the context must be derivable from the FCRs. The latter ensures that the FCRs license no feature structure with features which do not co-occur in the input structures.

From what we have seen in Section 12.2 it is obvious that the set of attribute implications from Table 2 is already an adequate set of FCRs for the input structures of Figure 2 if one interprets  $\perp$  as contradiction. In order to explain why our automatically derived set of implications contains so many more elements than the four given by Gazdar et al. (1985), we will compare the two sets:

It is apparent that all four FCRs from Table 3 can be derived from the attribute implications in the implication basis of Table 2: FCR 1 corresponds to implication 12.34; FCR 2 is contained in equivalence 12.31, FCR 3 in equivalence 12.30, and FCR 4 in equivalence 12.28.

However, Table 2 includes further implications, some of which bear no real information in the sense of GPSG since they either result from the sparse input data (cf. equivalence 12.29), from the special role of the feature value *VAL* (cf. implications 12.38–12.42), or from the fact that no knowledge about the exclusivity of features is implemented in FCA (cf. implication 12.43). Implication 12.35 follows from FCR 1 on the condition that, first, whenever *inv* is specified then *aux* is also specified and vice versa (equivalence 12.33) and, second, *inv* is restricted to the values + and –; implication 12.37 follows analogously from FCR 1.

The implications 12.27, 12.32, and 12.33 are missing in the FCRs and moreover, only one direction of the equivalences 12.28, 12.30, and 12.31 is stated in the FCRs. These implications regulate when categories necessarily must be specified with respect to certain features without saying anything about the concrete feature values. A very surprising gap in the list of FCRs is evident in implication 12.36, according to which passive verbs are not auxiliaries. This fact cannot be derived from the FCRs in Table 3. The GPSG grammar given in Gazdar et al. (1985) thus allows the feature-specification bundle  $\{[PAS], [+AUX], [-INV]\}$ , which encodes a non-inverted auxiliary in passive. This demonstrates that the automatically extracted basis of attribute implications is more explicit than the manually formulated FCRs, which arise from linguistic intuition and miss statements which probably were too obvious for the investigators. Hence, FCA can be a powerful tool for tracking down gaps in intellectually constructed theories.

The manually constructed FCRs of GPSG theories are not always as easy to read off a basis of attribute implications as in the example. Each attribute implication  $A \rightarrow B$  encodes a *cumulated Horn clause* of the

form  $\bigwedge A \rightarrow \bigwedge B$  or  $\bigwedge A \rightarrow \perp$  if  $B''$  equals the set of all attributes. But FCRs are not necessarily cumulated Horn clauses: some of the FCRs in Gazdar et al. (1985), p. 246, make use of negation and disjunction. The following consideration shows that such FCRs are implicitly encoded in the concept lattice of a feature-structure context, too. It is a well-known fact from propositional logics that each theory can be generated by conditionals, so called *observational statements* which use only disjunction and conjunction, but not negation. An FCR with a disjunctive premise can be directly transformed into a set of Horn clauses:

$$a \vee b \rightarrow c \quad \Leftrightarrow \quad (a \rightarrow c) \wedge (b \rightarrow c)$$

Hence, we can focus on the derivation of conditionals with disjunctive conclusions from concept lattices.

Let  $(G, M, I)$  be a formal context. A theory of observational statements over  $M$  is said to be complete if every conditional which is compatible with the data of the context is entailed by the theory; we call such a theory a *complete observational theory* of the context. The *information domain* of such a complete observational theory, i.e. the set of all maximal consistent subsets of  $M$  w.r.t. the theory, consists exactly of the intents of the object concepts.<sup>13</sup> These considerations open a way to derive a complete observational theory and hence a complete set of FCRs from a concept lattice:

Each formal concept  $(A, B)$  of the lattice which is not an object concept corresponds to an observational statement whose premise is the conjunction of the elements of the intent  $B$  and whose conclusion is the disjunction of the conjunctions of its subconcept intents minus  $B$ . Adding the corresponding statement of a concept to the theory amounts to removing the concept intent from the informational domain of the theory. For example, the statement corresponding to the attribute concept of *nform:val* is

$$\begin{aligned} n : \text{VAL} \wedge v : \text{VAL} \wedge v : - \wedge n : + \wedge \text{nform} : \text{VAL} \\ \rightarrow \text{nform} : \text{norm} \vee \text{nform} : \text{it}, \end{aligned} \quad (12.44)$$

which can be simplified by Table 2 to

$$\text{nform} : \text{VAL} \rightarrow \text{nform} : \text{norm} \vee \text{nform} : \text{it}.$$

This states that if an object bears the feature *nform*, then the latter must be specified for the value *norm* or *it*. As a second example, we look at the concept with the extent  $\{\text{can2}, \text{can3}\}$ : here the simplified

---

<sup>13</sup>The information domain of a complete Horn theory of a formal context, i.e. a complete theory consisting only of Horn clauses, equals the set of the intents of the formal concepts.

corresponding observational statement is

$$\text{aux} : + \wedge \text{inv} : - \rightarrow \text{vform} : \text{fin} \vee \text{vform} : \text{bse}.$$

Ganter and Krausse (1999) present an alternative procedure for systematically constructing a complete observational theory of a formal context. Depending on the concrete task for which the complete theory of a formal context is used, it often makes sense to restrict the theory class by dispensing with the disjunctive or conjunctive operators (Osswald and Petersen, 2002, 2003).

It remains to show how feature structure contexts can be derived either from a type signature or from FCRs. Given a type signature, the adequate feature structure context can be directly constructed from the set of totally well-typed and sort-resolved feature structures. A detailed description of the construction method (even for type signatures with co-references) is given in Petersen (2004, 2005). Given a set of FCRs, the corresponding formal context is equivalent to the information domain of the FCRs (see Osswald and Petersen, 2003).

## 12.4 Conclusion

We have seen that FCA provides three different ways to display data: in tabular form, as a hierarchy, and as a set of implications. None of the three reduce the structure of the given data, and that is why it is possible to switch from one representation to another without losing information. Hence, FCA is superior to many alternative approaches to induction inasmuch as it is neutral with respect to the analyzed data and describes them completely.<sup>14</sup> The relations between formal contexts and the corresponding concept lattices is absolutely transparent since there is exactly one of the latter for each context.

FCA allows us to capture the relationship between FCRs and type signatures explicitly. The prerequisite for this is the construction of a suitable feature-structure context.

It is remarkable that FCA has not yet been adopted as a standard tool in linguistics. Until now only a few linguists are employing FCA (for an overview see Priss, 2003); most of them explore the utility of FCA in lexical semantics and wordnets (e.g. Janssen, 2002). Our work shows that FCA can play an important methodological role for linguists in that it helps them to discover generalizations missed with intuitive

---

<sup>14</sup>Ganter and Wille (1998) on the disadvantages of FCA: “Nevertheless it may be exponential in size, compared to the formal context. Complexity therefore is, of course, a problem, even though there are efficient algorithms and advanced program systems. A formal context of moderate size may have more concepts than one would like to see individually.”

procedures and because it facilitates a better understanding of the relationships between different formal theories. We hope that FCA will soon come to take on a more important role in linguistics.

## References

- Bresnan, Joan. 1982. *The Mental Representation of Grammatical Relations*. Cambridge, MA: The MIT Press.
- de Smedt, Koenraad. 1984. Using object-oriented knowledge representation techniques in morphology and syntax programming. In *Proceedings of the European Conference on Artificial Intelligence 1984*, pages 181–184.
- Flickinger, Dan. 1987. *Lexical Rules in the Hierarchical Lexicon*. Ph.D. thesis, Stanford University.
- Ganter, Bernhard and R. Krausse. 1999. Pseudo models and propositional horn inference. Tech. rep., MATH-AL-15-1999, Technische Universität Dresden, Germany.
- Ganter, Bernhard and Rudolf Wille. 1998. Applied lattice theory: Formal concept analysis. In G. Grätzer, ed., *General Lattice Theory*, pages 591–605. Basel: Birkhäuser Verlag.
- Ganter, Bernhard and Rudolf Wille. 1999. *Formal Concept Analysis. Mathematical Foundations*. Berlin: Springer.
- Gazdar, Gerald, Ewan Klein, Geoff Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Oxford: Blackwell.
- Gazdar, Gerald and Geoffrey K. Pullum. 1982. *Generalized phrase structure grammar: a theoretical synopsis*. Bloomington, Indiana: Indiana University Linguistics Club.
- Gerdemann, Dale and Paul John King. 1993. Typed feature structures for expressing and computationally implementing feature cooccurrence restrictions. In *Proceedings of 4. Fachtagung der Sektion Computerlinguistik der Deutschen Gesellschaft für Sprachwissenschaft*, pages 33–39.
- Gerdemann, Dale and Paul John King. 1994. The Correct and Efficient Implementation of Appropriateness Specifications for Typed Feature Structures. In *Proceedings of the 15th Conference on Computational Linguistics (COLING-94)*, pages 956–960. Kyoto, Japan.
- Janssen, Maarten. 2002. *SIMuLLDA – a Multilingual Lexical Database Application using a Structural Interlingua*. Ph.D. thesis, Universiteit Utrecht.
- Kaplan, Ronald M. and Joan Bresnan. 1982. Introduction: grammars as mental representations of language. In Bresnan (1982), pages xvii–lii.

- Karttunen, Lauri. 1986. Radical lexicalism. Tech. Rep. CSLI-86-68, CSLI, Stanford. Reprinted in M. Baltin and A. Koch, eds., *Alternative Conceptions of Phrase Structure*, pages 43–65. Chicago: CUP. 1989.
- Osswald, Rainer and Wiebke Petersen. 2002. Induction of classifications from linguistic data. In *Proceedings of the ECAI-Workshop on Advances in Formal Concept Analysis for Knowledge Discovery in Databases*. Lyon.
- Osswald, Rainer and Wiebke Petersen. 2003. A logical approach to data-driven classification. *Lecture Notes in Computer Science* 2821:267–281.
- Petersen, Wiebke. 2004. Automatic induction of type signatures. Unpublished manuscript.
- Petersen, Wiebke. 2005. *Induktion von Vererbungshierarchien mit Mitteln der formalen Begriffsanalyse*. Ph.D. thesis, Heinrich-Heine-Universität Düsseldorf. (in progress).
- Pollard, Carl and Ivan A. Sag. 1987. *Information-Based Syntax and Semantics*. Stanford, CA: CSLI Lecture Notes.
- Priss, Uta. 2003. Linguistic applications of formal concept analysis. In *Proceedings of ICFCA 2003*. (to appear).
- Sag, Ivan and Thomas A. Wasow. 1999. *Syntactic Theory: A Formal Introduction*. Stanford, CA: CSLI.
- Shieber, Stuart, Hans Uszkoreit, Fernando Pereira, Jane Robinson, and Mabry Tyson. 1983. The formalism and implementation of PATR-II. In B. J. Grosz and M. Stickel, eds., *Research on Interactive Acquisition and Use of Knowledge*, techreport 4, pages 39–79. Menlo Park, CA: SRI International. Final report for SRI Project 1894.
- Sporleder, Caroline. 2003. *Discovering Lexical Generalisations. A Supervised Machine Learning Approach to Inheritance Hierarchy Construction*. Ph.D. thesis, Institute for Communication and Collaborative Systems. School of Informatics. University of Edinburgh.



---

## Bias decreases in proportion to the number of annotators

RON ARTSTEIN AND MASSIMO POESIO <sup>†</sup>

### Abstract

The effect of the individual biases of corpus annotators on the value of reliability coefficients is inversely proportional to the number of annotators (less one). As the number of annotators increases, the effect of their individual preferences becomes more similar to random noise. This suggests using multiple annotators as a means to control individual biases.

**Keywords** CORPUS ANNOTATION, RELIABILITY, KAPPA

### 13.1 Introduction

One of the problems of creating an annotated corpus is inter-annotator reliability—the extent to which different annotators “do the same thing” when annotating the corpus. Among the factors that may affect reliability is what we will call the individual annotator bias, informally thought of as the differences between the individual preferences of the various annotators. Methods to control bias include the development of clear annotation schemes, detailed and explicit manuals, and extensive training. Nevertheless, some individual differences in the interpretation of such schemes and manuals will always remain. We suggest another means to control for bias—increasing the number of annotators. We give a proof that the effect of individual annotator bias on standard measures of reliability decreases in proportion to the number of anno-

---

<sup>†</sup>This work was supported in part by EPSRC project GR/S76434/01, ARRAU. We wish to thank Tony Sanford, Patrick Sturt, Ruth Filik, Harald Clahsen, Sonja Eisenbeiss, and Claudia Felser.

tators (or, to be pedantic, in proportion to the number of annotators less one).

In order to test inter-annotator reliability, two or more annotators annotate the same text, and their annotations are compared using some statistical measure. Since the publication of Carletta (1996) it has been common in computational linguistics to use a family of related but distinct agreement coefficients often subsumed under the name “kappa”. Recently, Di Eugenio and Glass (2004) have pointed out that different members of this family make different assumptions about, among other things, individual annotator bias: some coefficients treat this bias as noise in the data (e.g.  $\pi$ , Scott, 1955), while others treat it as a genuine source of disagreement (e.g.  $\kappa$ , Cohen, 1960). Di Eugenio and Glass demonstrate, using examples with two annotators, that the choice of agreement coefficient can affect the reliability values.

In this paper we use the difference between the two classes of coefficients in order to quantify individual annotator bias. We then show that this measure decreases in proportion to the number of annotators. Of course, multiple annotators may still vary in their individual preferences. However, as the number of annotators grows, the effect of this variation as a source of disagreement decreases, and it becomes more similar to random noise.

While the results of this study are purely mathematical, they have also been tested in the field: we conducted a study of the reliability of coreference annotation using 18 subjects (the largest such study we know of), and we found that the differences between biased and unbiased agreement coefficients were orders of magnitude smaller than any of the other variables that affected reliability values. This shows that using many annotators is one way to overcome individual biases in corpus annotation.

### 13.2 Agreement among two coders: pi and kappa

We start with a simple case, of two annotators who have to classify a set of items into two categories. As a concrete example, we will call our annotators Alice and Bill, call the categories “yes” and “no”, and assume they classified ten items with the following results.

Alice:	Y Y N Y N Y N N Y Y
Bill:	Y Y N N Y Y Y N Y Y

Since Alice and Bill agree on the classification of seven of the ten items, we say that their observed agreement is 7/10 or 0.7. Generally, when two annotators classify a set of items into any number of distinct and mutually exclusive categories, their observed agreement is simply the



proportion of items on whose classification they agree.

Observed agreement in itself is a poor measure of inter-annotator reliability, because a certain amount of agreement is expected purely by chance; this amount varies depending on the number of categories and the distribution of items among categories. For this reason it is customary to report an agreement coefficient in which the observed agreement  $A_o$  is discounted by the amount of agreement expected by chance  $A_e$ . Two such coefficients, suitable for judging agreement between just two annotators, are  $\pi$  (Scott, 1955) and  $\kappa$  (Cohen, 1960); both are calculated according to the following formula.

$$\pi, \kappa = \frac{A_o - A_e}{1 - A_e}$$

The difference between  $\pi$  and  $\kappa$  is in the way the expected agreement is calculated. Both coefficients define expected agreement as the probability that the two annotators will classify an arbitrary item into the same category. But while  $\pi$  assumes that this probability is governed by a single distribution,  $\kappa$  assumes that each annotator has a separate probability distribution.

Let's see what this means in our toy example. According to  $\pi$ , we calculate a single probability distribution by looking at the totality of judgments: there are 13 "yes" judgments and 7 "no" judgments, so the probability of a "yes" judgment is 0.65 while that of a "no" judgment is 0.35; overall, the probability that the two annotators will classify an arbitrary item into the same category is  $0.65^2 + 0.35^2 = 0.545$ . According to  $\kappa$ , we calculate a separate probability distribution for each coder: for Alice the probability of a "yes" judgment is 0.6 and that of a "no" judgment is 0.4, while for Bill the probability of a "yes" judgment is 0.7 and that of a "no" judgment is 0.3; the overall probability that the two annotators will classify an arbitrary item into the same category is  $0.6 \cdot 0.7 + 0.4 \cdot 0.3 = 0.54$ , slightly lower than the probability calculated by  $\pi$ . This, in turn, makes the value of  $\kappa$  slightly higher than  $\pi$ .

$$\pi = \frac{0.7 - 0.545}{1 - 0.545} \approx 0.341 \quad \kappa = \frac{0.7 - 0.54}{1 - 0.54} \approx 0.348$$

More generally, for  $\pi$  we use  $P(k)$ , the overall probability of assigning an item to category  $k$ , which is the total number of such assignments by both coders  $\mathbf{n}_k$  divided by the overall number of assignments, which is twice the number of items  $\mathbf{i}$ . For  $\kappa$  we use  $P(k|c)$ , the probability of assigning an item to category  $k$  by coder  $c$ , which is the number of such assignments  $\mathbf{n}_{ck}$  divided by the number of items  $\mathbf{i}$ .

$$P(k) = \frac{1}{2\mathbf{i}} \mathbf{n}_k \quad P(k|c) = \frac{1}{\mathbf{i}} \mathbf{n}_{ck}$$

According to  $\pi$ , the probability that both coders assign an item to a particular category  $k \in K$  is  $P(k)^2$ , so the expected agreement is the sum of  $P(k)^2$  over all categories  $k \in K$ . As for  $\kappa$ , the probability that the two coders  $c_1$  and  $c_2$  assign an item to a particular category  $k \in K$  is  $P(k|c_1)P(k|c_2)$ , so the expected agreement is the sum of  $P(k|c_1)P(k|c_2)$  over all categories  $k \in K$ .

$$A_e^\pi = \sum_{k \in K} P(k)^2 \quad A_e^\kappa = \sum_{k \in K} P(k|c_1)P(k|c_2)$$

Since  $P(k)$  is the mean of  $P(k|c_1)$  and  $P(k|c_2)$  for each category  $k \in K$ , it follows that for any set of coding data,  $A_e^\pi \geq A_e^\kappa$ , and consequently  $\pi \leq \kappa$ , with the limiting case obtaining when the distributions of the two coders are identical.

### 13.3 Measuring the bias

Di Eugenio and Glass (2004) point out that  $\pi$  and  $\kappa$  reflect two different conceptualizations of the reliability problem (they refer to  $\pi$  and  $\kappa$  by the names  $\kappa_{S\&C}$  and  $\kappa_{C\&O}$ , respectively). For  $\pi$ , differences between the coders in the observed distributions of judgments are considered to be noise in the data, whereas for  $\kappa$  they reflect the relative biases of the individual coders, which is one of the sources of disagreement (Cohen, 1960, 40–41). Here we will show how this difference can be quantified and related to an independent measure—the variance of the individual coders’ distributions.

We should note that a single coder’s bias cannot be measured in and of itself—it can only be measured by comparing the coder’s distribution of judgments to some other distribution. Our agreement coefficients do not include reference to any source external to the coding data (such as information about the distribution of categories in the real world), and therefore we cannot measure the bias of an individual coder, but only the bias of the coders with respect to each other.

We are aware of several proposals in the literature for measuring individual coder bias. Zwick (1988) proposes a modified  $\chi^2$  test (Stuart, 1955), and Byrt et al. (1993) define a “Bias Index” which is the difference between the individual coders’ proportions for one category label (this only applies when there are exactly two categories). Since we are interested in the effect of individual coder bias on the agreement coefficients, we define  $B$ , the overall bias in a particular set of coding data, as the difference between the expected agreement according to  $\pi$

and the expected agreement according to  $\kappa$ .

$$\begin{aligned} B = A_e^\pi - A_e^\kappa &= \sum_{k \in K} P(k)^2 - \sum_{k \in K} P(k|c_1)P(k|c_2) \\ &= \sum_{k \in K} \left( \frac{P(k|c_1) + P(k|c_2)}{2} \right)^2 - P(k|c_1)P(k|c_2) \\ &= \sum_{k \in K} \left( \frac{P(k|c_1) - P(k|c_2)}{2} \right)^2 \end{aligned}$$

The bias is a measure of variance. Take  $c$  to be a random variable, with equal probabilities for each of the two coders:  $P(c_1) = P(c_2) = 0.5$ . For each category  $k \in K$ , we calculate the mean  $\mu$  and variance  $\sigma^2$  of  $P(k|c)$ .

$$\begin{aligned} \mu_{P(k|c)} &= \frac{P(k|c_1) + P(k|c_2)}{2} \\ \sigma_{P(k|c)}^2 &= \frac{(P(k|c_1) - \mu_{P(k|c)})^2 + (P(k|c_2) - \mu_{P(k|c)})^2}{2} \\ &= \left( \frac{P(k|c_1) - P(k|c_2)}{2} \right)^2 \end{aligned}$$

We find that the bias  $B$  is the sum of the variances of  $P(k|c)$  for all categories  $k \in K$ .

$$B = \sum_{k \in K} \sigma_{P(k|c)}^2$$

This is a convenient way to quantify the relative bias of two coders. In the next section we generalize  $\pi$  and  $\kappa$  to apply to multiple coders, and see that the bias drops in proportion to the number of coders.

### 13.4 Agreement among multiple coders

We now provide generalizations of  $\pi$  and  $\kappa$  which are applicable when the number of coders  $\mathbf{c}$  is greater than two. The generalization of  $\pi$  is the same as the coefficient which is called, quite confusingly,  $\kappa$  by Fleiss (1971). We will call it  $\pi$  because it treats individual coder bias as noise in the data and is thus better thought of as a generalization of Scott's  $\pi$ , reserving the name  $\kappa$  for a proper generalization of Cohen's  $\kappa$  which takes bias as a source of disagreement. As far as we are aware, ours is the first generalization of  $\kappa$  to multiple coders—other sources which claim to give a generalization of  $\kappa$  actually report Fleiss's coefficient (e.g. Bartko and Carpenter, 1976, Siegel and Castellan, 1988, Di Eugenio

and Glass, 2004).

With more than two coders we can no longer define the observed agreement as the percentage of items on which there is agreement, since there will inevitably be items on which some coders agree amongst themselves while others disagree. The amount of agreement on a particular item is therefore defined as the proportion of agreeing judgment pairs out of the total number of judgment pairs for the item. Let  $\mathbf{n}_{ik}$  stand for the number of times an item  $i$  is classified in category  $k$  (i.e. the number of coders that make such a judgment). Each category  $k$  contributes  $\binom{\mathbf{n}_{ik}}{2}$  pairs of agreeing judgments for item  $i$ ; the amount of agreement  $\text{agr}_i$  for item  $i$  is the sum of  $\binom{\mathbf{n}_{ik}}{2}$  over all categories  $k \in K$ , divided by  $\binom{\mathbf{c}}{2}$ , the total number of judgment pairs per item.

$$\text{agr}_i = \frac{1}{\binom{\mathbf{c}}{2}} \sum_{k \in K} \binom{\mathbf{n}_{ik}}{2} = \frac{1}{\mathbf{c}(\mathbf{c} - 1)} \sum_{k \in K} \mathbf{n}_{ik}(\mathbf{n}_{ik} - 1)$$

The overall observed agreement is the mean of  $\text{agr}_i$  for all items  $i \in I$ .

$$A_o = \frac{1}{\mathbf{i}} \sum_{i \in I} \text{agr}_i = \frac{1}{\mathbf{i}\mathbf{c}(\mathbf{c} - 1)} \sum_{i \in I} \sum_{k \in K} \mathbf{n}_{ik}(\mathbf{n}_{ik} - 1)$$

Since agreement is measured as the proportion of agreeing judgment pairs, the agreement expected by chance is the probability that any given pair of judgments for the same item would agree; this, in turn, is equivalent to the probability that two arbitrary coders would make the same judgment for a particular item by chance. For  $\pi$  we use  $P(k)$ , the overall probability of assigning an item to category  $k$ , which is the total number of such assignments by all coders  $\mathbf{n}_k$  divided by the overall number of assignments, which is the number of items  $\mathbf{i}$  multiplied by the number of coders  $\mathbf{c}$ . For  $\kappa$  we use  $P(k|c)$ , the probability of assigning an item to category  $k$  by coder  $c$ , which is the number of such assignments  $\mathbf{n}_{ck}$  divided by the number of items  $\mathbf{i}$ .

$$P(k) = \frac{1}{\mathbf{i}\mathbf{c}} \mathbf{n}_k \quad P(k|c) = \frac{1}{\mathbf{i}} \mathbf{n}_{ck}$$

According to  $\pi$ , the probability that two arbitrary coders assign an item to a particular category  $k \in K$  is  $P(k)^2$ , so the expected agreement is the sum of  $P(k)^2$  over all categories  $k \in K$ . As for  $\kappa$ , the probability that two particular coders  $c_m$  and  $c_n$  assign an item to category  $k \in K$  is  $P(k|c_m)P(k|c_n)$ ; since all coders judge all items, the probability that an arbitrary pair of coders assign an item to category  $k$  is the arithmetic mean of  $P(k|c_m)P(k|c_n)$  over all coder pairs  $c_m, c_n$ , and the expected

agreement is the sum of this probability over all categories  $k \in K$ .

$$A_e^\pi = \sum_{k \in K} P(k)^2 \quad A_e^\kappa = \sum_{k \in K} \frac{1}{\binom{c}{2}} \sum_{m=1}^{c-1} \sum_{n=m+1}^c P(k|c_m)P(k|c_n)$$

It is easy to see that  $A_e^\kappa$  for multiple coders is the mean of the two-coder  $A_e^\kappa$  values from section 13.2 for all coder pairs.

We start with a numerical example. Instead of two annotators we now have four; furthermore, it so happens that Claire gives exactly the same judgments as Alice, and Dave gives exactly the same judgments as Bill.

Alice, Claire: Y Y N Y N Y N N Y Y  
 Bill, Dave: Y Y N N Y Y Y N Y Y

The expected agreement according to  $\pi$  remains 0.545 as in the case of just Alice and Bill, since the overall proportion of “yes” judgments is still 0.65 and that of “no” judgments is still 0.35. But for the calculation of expected agreement according to  $\kappa$  we also have to take into account the expected agreement between Alice and Claire and the expected agreement between Bill and Dave. Overall, the probability that two arbitrary annotators will classify an item into the same category is  $\frac{1}{6}[0.6^2 + 4 \cdot 0.6 \cdot 0.7 + 0.7^2] + \frac{1}{6}[0.4^2 + 4 \cdot 0.4 \cdot 0.3 + 0.3^2] = 0.54333 \dots$ ; this value is still lower than the probability calculated by  $\pi$ , but higher than it was for two annotators. If we add a fifth annotator with the same judgments as Alice and Claire and a sixth with the judgment pattern of Bill and Dave, expected agreement according to  $\pi$  remains 0.545 while expected agreement according to  $\kappa$  rises to 0.544. It appears, then, that as the number of annotators increases, the value of  $A_e^\kappa$  approaches that of  $A_e^\pi$ . We now turn to the formal proof.

We start by taking the formulas for expected agreement above and putting them into a form that is more useful for comparison with one another.

$$\begin{aligned} A_e^\pi &= \sum_{k \in K} P(k)^2 = \sum_{k \in K} \left( \frac{1}{c} \sum_{m=1}^c P(k|c_m) \right)^2 \\ &= \sum_{k \in K} \frac{1}{c^2} \sum_{m=1}^c \sum_{n=1}^c P(k|c_m)P(k|c_n) \\ A_e^\kappa &= \sum_{k \in K} \frac{1}{\binom{c}{2}} \sum_{m=1}^{c-1} \sum_{n=m+1}^c P(k|c_m)P(k|c_n) \\ &= \sum_{k \in K} \frac{1}{c(c-1)} \left( \sum_{m=1}^c \sum_{n=1}^c P(k|c_m)P(k|c_n) - \sum_{m=1}^c P(k|c_m)^2 \right) \end{aligned}$$

The overall bias is the difference between the expected agreement according to  $\pi$  and the expected agreement according to  $\kappa$ .

$$\begin{aligned} B &= A_e^\pi - A_e^\kappa \\ &= \frac{1}{\mathbf{c} - 1} \sum_{k \in K} \frac{1}{\mathbf{c}^2} \left( \mathbf{c} \sum_{m=1}^{\mathbf{c}} P(k|c_m)^2 - \sum_{m=1}^{\mathbf{c}} \sum_{n=1}^{\mathbf{c}} P(k|c_m)P(k|c_n) \right) \end{aligned}$$

We now calculate the mean  $\mu$  and variance  $\sigma^2$  of  $P(k|c)$ , taking  $c$  to be a random variable with equal probabilities for all of the coders:  $P(c) = \frac{1}{\mathbf{c}}$  for all coders  $c \in C$ .

$$\begin{aligned} \mu_{P(k|c)} &= \frac{1}{\mathbf{c}} \sum_{m=1}^{\mathbf{c}} P(k|c_m) \\ \sigma_{P(k|c)}^2 &= \frac{1}{\mathbf{c}} \sum_{m=1}^{\mathbf{c}} (P(k|c_m) - \mu_{P(k|c)})^2 \\ &= \frac{1}{\mathbf{c}} \sum_{m=1}^{\mathbf{c}} P(k|c_m)^2 - 2\mu_{P(k|c)} \frac{1}{\mathbf{c}} \sum_{m=1}^{\mathbf{c}} P(k|c_m) + \mu_{P(k|c)}^2 \frac{1}{\mathbf{c}} \sum_{m=1}^{\mathbf{c}} 1 \\ &= \left( \frac{1}{\mathbf{c}} \sum_{m=1}^{\mathbf{c}} P(k|c_m)^2 \right) - \mu_{P(k|c)}^2 \\ &= \frac{1}{\mathbf{c}^2} \left( \mathbf{c} \sum_{m=1}^{\mathbf{c}} P(k|c_m)^2 - \sum_{m=1}^{\mathbf{c}} \sum_{n=1}^{\mathbf{c}} P(k|c_m)P(k|c_n) \right) \end{aligned}$$

The bias  $B$  is thus the sum of the variances of  $P(k|c)$  for all categories  $k \in K$ , divided by the number of coders less one.

$$B = \frac{1}{\mathbf{c} - 1} \sum_{k \in K} \sigma_{P(k|c)}^2$$

Since the variance does not increase in proportion to the number of coders, we find that the more coders we have, the lower the bias; at the limit,  $\kappa$  approaches  $\pi$  as the number of coders approaches infinity.

### 13.5 Conclusion

We have seen that one source of disagreement among annotators, individual bias, decreases as the number of annotators increases. This does not mean that reliability increases with the number of annotators, but rather that the individual coders' preferences become more similar to random noise. This suggests using multiple annotators as a means for controlling bias.

There is a further class of agreement coefficients which allow for gradient disagreements between annotators, for example weighted kappa  $\kappa_w$  (Cohen, 1968) and  $\alpha$  (Krippendorff, 1980). Passonneau (2004), for example, uses  $\alpha$  to measure reliability of coreference annotation, where different annotators may partially agree on the identity of an anaphoric chain. We cannot treat these coefficients here due to space limitations, but the same result holds for gradient coefficients—bias decreases in proportion to the number of annotators. We performed an experiment testing the reliability of coreference annotation among 18 naive subjects, using  $\alpha$  and related measures (Poesio and Artstein, 2005); we found that the effect of bias on the agreement coefficients was substantially lower than any of the other variables that affected reliability.

## References

- Bartko, John J. and William T. Carpenter, Jr. 1976. On the methods and theory of reliability. *Journal of Nervous and Mental Disease* 163(5):307–317.
- Byrt, Ted, Janet Bishop, and John B. Carlin. 1993. Bias, prevalence and kappa. *Journal of Clinical Epidemiology* 46(5):423–429.
- Carletta, Jean. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics* 22(2):249–254.
- Cohen, Jacob. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20(1):37–46.
- Cohen, Jacob. 1968. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin* 70(4):213–220.
- Di Eugenio, Barbara and Michael Glass. 2004. The kappa statistic: A second look. *Computational Linguistics* 30(1):95–101.
- Fleiss, Joseph L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76(5):378–382.
- Krippendorff, Klaus. 1980. *Content Analysis: An Introduction to Its Methodology*, chap. 12, pages 129–154. Beverly Hills: Sage.
- Passonneau, Rebecca J. 2004. Computing reliability for coreference annotation. In *Proceedings of LREC*. Lisbon.
- Poesio, Massimo and Ron Artstein. 2005. The reliability of anaphoric annotation, reconsidered: Taking ambiguity into account. In *Proceedings of the ACL Workshop on Frontiers in Corpus Annotation*. Ann Arbor.

- Scott, William A. 1955. Reliability of content analysis: The case of nominal scale coding. *Public Opinion Quarterly* 19(3):321–325.
- Siegel, Sidney and N. John Castellan, Jr. 1988. *Nonparametric Statistics for the Behavioral Sciences*, chap. 9.8, pages 284–291. New York: McGraw-Hill, 2nd edn.
- Stuart, Alan. 1955. A test for homogeneity of the marginal distributions in a two-way classification. *Biometrika* 42(3/4):412–416.
- Zwick, Rebecca. 1988. Another look at interrater agreement. *Psychological Bulletin* 103(3):374–378.



---

# The Proper Treatment of Coordination in Peirce Grammar

HANS LEISS

## Abstract

Peirce grammar provides a first-order, relational semantics for extensional fragments of natural language. Proposals in the literature to model natural language coordination by boolean operations in Peirce grammar lead to inadequate results. We propose to equip Peirce grammar with an additional sort of finite trees and to admit recursion on trees for defining the evaluation of expressions. We demonstrate how this leads to a better treatment of coordination and other recursive constructions, such as the inverse linking of quantifiers in nested noun phrases.

**Keywords** PEIRCE GRAMMAR, RELATIONAL SEMANTICS, COORDINATION, INVERSE LINKING

## 14.1 Introduction

A Peirce grammar is a context-free grammar with an interpretation of expressions as sets or binary relations on a universe  $V$ . Peirce grammars have evolved from an interpretation of context-free grammars in extended relation algebras by Suppes (1976), which in turn owe much to the work of Peirce (1932) on relation algebra as a semantics of natural language.

While relational semantics is well-established for programming languages, it is less known in the context of natural language, despite its roots. Peirce grammars have been used by Suppes (1976) and Böttner (1999) to model fragments of English and German. The values of adjectives, nouns, prepositions and verbs are sets or relations, those of individual names are atomic sets, but function words like determiners

and conjunctions are treated syncategorematically. The value of complex expressions is computed bottom-up by algebraic terms attached to grammar rules.

A distinctive feature of Peirce grammar is its algebraic semantics: there is no infinite hierarchy of types, no variables and variable binders, no application of meaning objects to other meaning objects, but just construction of meanings from basic ones by applications of a few given operations; there are no built-in relations between meanings such as generalized quantifiers as relations between sets, except for equality and a definable subsumption relation. Since the algebraic operations have an equational axiomatization, equational reasoning is sufficient to establish semantic equivalence of expressions, which also makes Peirce grammar attractive from a computational point of view.

It has been claimed that this framework is too weak to cope with recursive syntactic structures and needs infinitely many categories or grammar rules to fix this (cf. Jäger (2004)). After introducing Peirce grammar in section 2, we address the problem of recursive structures in section 3, by adding finite trees to Peirce algebras and structural recursion to the evaluation. We demonstrate our extension on two examples: coordination of nouns and adjectives and inverse linking of quantifiers in nested noun phrases. Relations to other theories are discussed in section 4, and advantages, drawbacks and open problems in section 5.

## 14.2 Peirce grammar

A Peirce algebra is a two-sorted algebra of so-called *sets* and *relations* with the usual boolean operations on each sort as well as some operations relating the two sorts. A Peirce grammar is a context-free grammar  $G$  with an evaluation function  $[[\cdot]] : L(G) \rightarrow \mathcal{P}$ , assigning to each expression  $w \in L(G)$  a meaning  $[[w]]$  in a Peirce algebra  $\mathcal{P}$ . In particular, the meaning objects are first-order objects; higher-order objects like functions in Montague-grammar are not admitted.

### 14.2.1 Peirce algebra

**Definition 30** Let  $V$  be a set, called the *universe*.  $\mathcal{P}_V = (\mathcal{B}_V, \mathcal{R}_V, ;, \smile, \circ)$ , the *Peirce algebra over  $V$* , consists of

1. the boolean algebra  $\mathcal{B}_V = (2^V, \cup, \cap, \bar{\phantom{x}}, \emptyset, V)$  of all subsets of  $V$ ,
2. the relation algebra  $\mathcal{R}_V = (2^{V \times V}, \cup, \cap, \bar{\phantom{x}}, \emptyset, V \times V, ;, \smile, \circ, I)$  of all binary relations on  $V$ , where  $;, \smile, \circ, I$  are
  - (a) the *relative product*  $S;T := \{\langle a, c \rangle \mid \exists b(R(a, b) \wedge T(b, c))\}$ ,
  - (b) the *converse*  $S^\smile := \{\langle b, a \rangle \mid S(a, b)\}$ , and
  - (c) the *identity relation*  $I := \{\langle a, a \rangle \mid a \in V\}$ ,

3. the *peircean product* : from  $\mathcal{R}_V \times \mathcal{B}_V$  to  $\mathcal{B}_V$ , where

$$R : B := \{a \in V \mid \exists b \in V (R(a, b) \wedge B(b))\}$$

is the *inverse image* of the set  $B$  under the relation  $R$ , and

4. the (right) *cylindrification*  $^c$  :  $\mathcal{B}_V \rightarrow \mathcal{R}_V$ , where

$$A^c := \{\langle a, b \rangle \in V \times V \mid A(a) \wedge b \in V\} = A \times V$$

is the (right) *cylinder* over the set  $A$ .

An equational axiomatization of the properties of the operations of  $\mathcal{P}_V$  by Brink et al. (1994) defines the notion of a Peirce algebra  $\mathcal{P} = (\mathcal{B}, \mathcal{R}, :, ^c)$ , where  $\mathcal{B} = (B, +, \cdot, \bar{\phantom{x}}, 0, 1)$  is a boolean algebra,  $\mathcal{R} = (R, +, \cdot, \bar{\phantom{x}}, 0, 1, ;, \check{\phantom{x}}, I)$  a relation algebra, and “ $:$ ” :  $R \times B \rightarrow B$  and  $^c : B \rightarrow R$  are analogs of the Peirce product and cylindrification.

We will use *Peirce-terms* built from set- and relation variables, the constants  $0, 1, I$  and the function symbols  $+, \cdot, \bar{\phantom{x}}, :, \check{\phantom{x}}$ . Each term has a type, *set* or *rel.* The ambiguity of  $0, 1, +, \cdot, \bar{\phantom{x}}$  is easily resolved by the context, or indicated as in  $+^B$  versus  $+^R$ , and likewise for the type of variables.

Many useful operations on sets and relations can be defined and their equational properties be proved in Peirce algebra. Some of these, with their interpretation in  $\mathcal{P}_V$ , are: the *relative sum*  $\ddagger$  of two relations,

$$R \ddagger S := \overline{\overline{R}; \overline{S}} = \{\langle a, b \rangle \in V \times V \mid \forall c \in V (R(a, c) \vee S(c, b))\},$$

the *peircean sum*  $\ddagger$  of a relation and a set,

$$R \ddagger B := \overline{\overline{R} : \overline{B}} = \{a \in V \mid \forall b \in V (R(a, b) \vee B(b))\},$$

the *domain* and *range* of a relation,

$$\text{dom}(R) := R : 1, \quad \text{ran}(R) := \check{R} : 1 = \{b \in V \mid \exists a \in V R(a, b)\},$$

the *cartesian product* of two sets,

$$A \times B := A^c \cdot (B^c)^\check{\phantom{x}} = (A \times V) \cap (V \times B).$$

### 14.2.2 Peircean operations as natural language constructs

Any Peirce algebra term  $t(x_1, \dots, x_n)$  can be translated into a first-order formula  $\varphi(X_1, \dots, X_n, y, z)$  with binary relations  $X_i$  and two individual variables  $y, z$ , but not conversely: the quantification in these formulas is of a special form. Basically, Peirce (1932) observed that verbs and nouns combine via a quantifier that connects one argument of the former with one of the latter, as in

$$Qz(R(x_1, \dots, x_{i-1}, z, x_{i+1}, \dots, x_n) \circ S(y_1, \dots, y_{j-1}, z, y_j, \dots, y_m)),$$

where  $Q$  is a quantifier and  $\circ$  a binary boolean operator. Some examples with  $n, m \leq 2$  are:

1. combining a transitive verb and a relational noun to a transitive verb:

$$\text{knows a brother of} = K ; B = \{\langle a, c \rangle \mid \exists b(a K b \wedge b B c)\}$$

$$\text{knows no brother of} = \overline{K} ; \overline{B} = \overline{K} \dagger \overline{B}$$

$$\text{knows all brothers of} = K \dagger \overline{B} = \{\langle a, c \rangle \mid \forall b(b B c \rightarrow a K b)\}$$

$$\text{knows not all brothers of} = \overline{K} \dagger \overline{B} = \overline{K} ; B$$

$$\text{knows only brothers of} = \overline{K} \dagger B = \{\langle a, c \rangle \mid \forall b(a K b \rightarrow b B c)\}$$

$$\text{knows not only brothers of} = \overline{K} \dagger B = K ; \overline{B}$$

2. combining two relational nouns to a relational noun:

$$\text{daughter of a brother of} = D ; B$$

$$\text{reader of all books by} = R \dagger \overline{B}$$

3. combining a transitive verb and a common noun to an intransitive verb:

$$\text{likes a book} = L : B = \{a \mid \exists b(L(a, b) \wedge B(b))\}$$

$$\text{likes all books} = L \dagger \overline{B} = \{a \mid \forall b(B(b) \rightarrow L(a, b))\}$$

Moreover, verbs of equal subcategorization combine by boolean operations, and the converse  $\checkmark$  amounts to the passive of transitive verbs.

### 14.2.3 Peirce grammar

From now on, we extend the Peirce-operations by a few *test functions*,  $E, U : \text{set} \rightarrow \text{set}$  and  $ER, UR : \text{rel} \rightarrow \text{rel}$ . They are to be interpreted as testing whether their argument is  $\neq 0$ , or  $= 1$ , and return one of the values 0 or 1 which serve as truth values *false* and *true*.

In a Peirce algebra  $\mathcal{P} = (\mathcal{B}, \mathcal{P}, :, \checkmark)$ , we identify the restrictions of  $\mathcal{B}$  resp.  $\mathcal{R}$  to its elements  $\{0, 1\}$  with the algebra  $\mathbb{B}$  of truth values.

**Definition 31** A *Peirce grammar*  $PG = (G, \text{type}, e)$  consists of

1. a context-free grammar  $G = (T, N, S, P)$ , with finite sets of terminals  $T$ , nonterminals  $N$ , syntax rules  $P \subseteq N \times (T \cup N)^*$ , and a sentence category  $S \in N$ ,
2. a function  $\text{type} : N \cup T \rightarrow \{\text{set}, \text{rel}\}$ , which may be partial on the set of terminals,
3. a mapping  $e$ , which assigns to each rule  $r = (A \rightarrow A_1 \cdots A_k) \in P$  a Peirce-term  $t_r(X_{i_1}, \dots, X_{i_n})$  of type  $\text{type}(A)$ , where  $X_j$  has type  $(A_j)$  when  $A_{i_1} \cdots A_{i_n}$ ,  $n \geq 1$ , is the subsequence of typed elements of  $A_1 \cdots A_k$ .

*Convention:* Instead of  $e(A \rightarrow \alpha) = t$  we write  $A \rightarrow \alpha[t]$ , using the typed terminals and nonterminals of  $\alpha$  as variables in  $t$ . Several rules for the nonterminal  $A$  may be combined as in

$$A \rightarrow \alpha_1[t_1] \mid \cdots \mid \alpha_n[t_n].$$

**Definition 32** An *interpretation*  $(\mathcal{P}, v)$  of a Peirce grammar  $(G, type, e)$  consists of a Peirce algebra  $\mathcal{P} = (\mathcal{B}, \mathcal{R}, :, ^c, E, U, ER, UR)$  with test functions and an evaluation function  $v : T \rightarrow \mathcal{P}$ , assigning to each typed terminal  $X$  an element  $v(X)$  of sort  $type(X)$ , i.e. of  $\mathcal{B}$  or  $\mathcal{R}$ .

For each syntactic tree-structure  $\tau$  of expressions  $w \in L(A)$ ,  $A \in N$ , the *value*  $\llbracket \tau \rrbracket_v^{\mathcal{P}}$  of  $\tau$  in  $\mathcal{P}$  under  $v$  is defined as follows:

1. If  $\tau$  is the one-node tree of  $w \in T$ , then  $\llbracket \tau \rrbracket_v^{\mathcal{P}} := v(w)$ .
2. If  $\tau$  is the combination of trees  $\tau_{i_1}, \dots, \tau_{i_n}$  by the rule  $A \rightarrow A_1 \cdots A_k [t]$  with typed constituents  $A_{i_1}, \dots, A_{i_n}$ , then

$$\llbracket \tau \rrbracket_v^{\mathcal{P}} := t^{\mathcal{P}}(\llbracket \tau_{i_1} \rrbracket_v^{\mathcal{P}}, \dots, \llbracket \tau_{i_n} \rrbracket_v^{\mathcal{P}}).$$

Typeless terminals in a rule  $r$  have syncategorematic meaning only, i.e. they are not assigned a value in  $\mathcal{P}$ , but are relevant for choosing  $t_r$ .

**Example 37** *This is a fragment of the grammar of Böttner (1999).*

$$\begin{array}{lcl} S & \rightarrow & UQ N VP [U(\overline{N} + VP)] \quad | \quad EQ N VP [E(N \cdot VP)] \\ VP & \rightarrow & IV [IV] \quad | \quad TVP [TVP] \\ TVP & \rightarrow & TV EQ N [TV : N] \quad | \quad TV UQ N [TV \dagger \overline{N}] \end{array}$$

The *VP*-rules say that the meaning of a *VP* is the meaning of the intransitive verb used, given by the evaluation  $v : T \rightarrow \mathcal{P}$  of an interpretation, or the meaning of the transitive verb phrase. The first *TVP*-rule says that the meaning of a transitive verb followed by an existential quantifier and a common noun is the inverse image  $[TV : N]$  of the set corresponding to  $N$  under the relation corresponding to *TV*. In the second *S*-rule, the quantifier *EQ* contributes to the sentence meaning by way of testing if the intersection of the *N*- and *VP*-meanings is nonempty. Note that *EQ* has different effects in the *TVP*- and *S*-rules.

Quantifiers and noun phrases, which in Montague grammar have higher-order functions as values, are not considered expressions in Peirce grammar; they have no meaning object attached at all. In fact, there is no way to have meaningful *NP*'s in Peirce grammar:

**Example 38** (Suppes (1976)) *There is no category NP in Peirce grammar that subsumes quantified noun phrases EQ N and UQ N in the sense that*

$$\begin{array}{lcl} S & \rightarrow & UQ N VP [U(\overline{N} + VP)] \\ & | & EQ N VP [E(N \cdot VP)] \end{array} \quad (14.45)$$

would be equivalent (i.e. generate the same sentences with the same values) to the rules

$$\begin{array}{lcl} S & \rightarrow & NP VP [s(NP, VP)] \\ NP & \rightarrow & UQ N [u(N)] \quad | \quad EQ N [e(N)] \end{array} \quad (14.46)$$

for suitably chosen functions  $s, u, e$  on sets: in a Peirce algebra where  $\mathcal{B}$  is the 2-element algebra  $\mathbb{B}$ , there are no functions  $s, u, e$  that give the expected values  $s(u(0), 0) = U(\bar{0} + 0) = 1, \dots, s(e(1), 1) = E(1 \cdot 1) = 1$ .

Therefore, Peirce grammar prefers “flat” syntactic structures: since the value of an expression is computed from the first-order values of its immediate constituents, only flat structures provide enough arguments to the evaluation functions to make the necessary distinctions. Alternatively, one can use groupings with different category names:

**Example 39** *Böttner (1999), rules 51–58, uses categories CPNP and APNP for phrases of conjunctions resp. alternatives of proper nouns, which get the same meaning but are treated differently when evaluating sentences:*

$$\begin{array}{lcl} \text{CPNP} & \rightarrow & \text{PN Conj PN } [PN + PN] \\ \text{APNP} & \rightarrow & \text{PN Disj PN } [PN + PN] \\ S & \rightarrow & \text{CPNP VP } [U(\overline{\text{CPNP}} + \text{VP})] \\ & | & \text{APNP VP } [E(\text{APNP} \cdot \text{VP})] . \end{array}$$

Thus, though “John and Mary” and “John or Mary” get the same set value, the two alternatives of the  $S$ -rule can use different Peirce-terms to define their meaning, as the syntactic construction of the subject is coded in its category name.

However, for nested occurrences of *Conj* and *Disj*, as in “John or Paul and Mary”, collecting the constituents’ meanings by set union  $+$  would erase differences that must be maintained.

This trick of coding the structure of a phrase in its category name can only be used for finitely many different phrase structures; otherwise, we’d need infinitely many categories in the grammar.

### 14.3 Peirce grammar with trees and recursive evaluation

To overcome the above problems, we propose to allow finite trees as another sort of objects in Peirce algebras. More precisely, we define:

**Definition 33** An *extended Peirce algebra*  $\mathcal{P}' = (\mathcal{B}, \mathcal{R}, \mathcal{T}, :, {}^c)$  is a Peirce algebra  $\mathcal{P} = (\mathcal{B}, \mathcal{R}, :, {}^c)$  expanded by the free algebra  $\mathcal{T}$  of finite ordered trees generated from the elements of  $\mathcal{B}$  and  $\mathcal{R}$  by a finite set of constructor functions. An *extended Peirce term* is a term that is built from variables of the sorts *set*, *rel* and *tree* by means of the Peircean operations, the constructor functions, and operations that are defined by structural recursion on trees.

**Definition 34** An *extended Peirce grammar*  $PG = (G, type, e)$  is like a Peirce grammar, except that  $type : N \cup T \rightarrow \{set, rel, tree\}$  may also assign the sort *tree* to terminals and nonterminals, and  $e$  assigns to each rule  $r$  of  $G$  an extended Peirce term of the appropriate type.

**Example 40** *There is an extended Peirce grammar with rules (14.46) which is equivalent to the Peirce grammar with rules (14.45). In the rules of (14.46), let  $u, e : set \rightarrow tree$  be unary constructor functions, and define  $s : tree \times set \rightarrow set$  by recursion on its first argument:*

$$\begin{aligned} s(u(N), VP) &:= U(\overline{N} + VP), \text{ else,} \\ s(e(N), VP) &:= E(N \cdot VP), \text{ else.} \end{aligned}$$

*These two clauses perform the case distinction on NPs which is impossible in Peirce algebra.*

### 14.3.1 Coordination

In a higher-order setting such as the one of Montague (1974) or Keenan and Faltz (1985), the meaning of coordinated phrases can be defined pointwise, as is done by

$$\llbracket NP_1 \text{ and } NP_2 \rrbracket := \lambda P^{(e \rightarrow t)}(\llbracket NP_1 \rrbracket(P) \wedge \llbracket NP_2 \rrbracket(P)),$$

for noun phrase meanings  $\llbracket NP \rrbracket : (e \rightarrow t) \rightarrow t$ , using the conjunction  $\wedge$  on the type  $t$  of truth values. A drawback of this approach is that even if the basic types  $e$  of individuals and  $t$  of truth values have a decidable equality, the types of functions  $(e \rightarrow t)$  and functionals  $(e \rightarrow t) \rightarrow t$  do not. Moreover, the higher-order value is obtained by abstracting from contexts  $P$  of  $e$ -type positions in sentences, ignoring positions of different type, such as  $e \times e$  for subjects of symmetric predicates, as in *John and Mary know each other*. Thus, a context-independent value for  $PN$ -conjunctions alone has sum-type  $(e \rightarrow t) + (e \times e \rightarrow t)$  or worse, so the background type theory would at least have to be a bit more complicated than the one usually assumed.

In a first-order setting like Peirce algebra, one might use the boolean operations on the sets and relations to interpret coordinations. Böttner (1999), p.84, interprets *Conj* resp. *Disj* for categories  $VP$ ,  $TVP$  (transitive verb phrase),  $AP$  and  $PP$  as intersection resp. union of sets or relations, but interprets both as union for proper and common nouns.

This does not work properly: (i) Conjunction of  $PP$ 's or  $P$ 's cannot be interpreted as intersection in *A tree was standing in front of and behind the house*, as the intersection of the two relations is empty. (ii) Conjunction of  $N$ 's cannot be interpreted as set union in *big (ants and elephants)*, where *big* has to be relativized to the two different nouns separately. (iii) Negation cannot be set complement resp. difference

in *John, but not Mary or men, but not women*, as then these would coincide with just *John or men*.

Since in contexts like *all (ants and elephants)*, conjunction of *N*'s *does* mean set union, while in (ii) it does not, the meaning of a coordinated expression may depend on the context of its use. Therefore, we better treat coordination as an abbreviation mechanism, reducing, in a context-dependent way, boolean combinations of phrases to boolean combinations of sentences.

When implementing this view by syntactic transformations, recursively embedded coordinations lead to recursive transformations, making it difficult to define the meaning of expressions by semantic attributes in the syntax rules (at least, of common tools like Yacc). Moreover, syntactic transformations must not copy expressions that lead to different values at different occurrences, or disturb quantifier scopes.

We propose a different route: don't expand coordinated phrases on the syntactic level, but during evaluation. For this it is necessary that from the semantic value of a coordinated phrase we can recover the form of the coordination and the meaning constituents. For example, to obtain

$$\llbracket (NP_1 \text{ but not } NP_2) VP \rrbracket = \llbracket (NP_1 VP) \text{ and not } (NP_2 VP) \rrbracket,$$

from the values  $\llbracket NP_1 \text{ but not } NP_2 \rrbracket$  and  $\llbracket VP \rrbracket$  of the constituents on the left, we need to be able to read off the values of both constituent *NP*'s from that of the coordinated one, but also that the second one was negated.

*Coordination rules.* To treat coordination in extended Peirce grammar, we use binary constructor functions  $\oplus$ ,  $\odot$  and  $\ominus$  to build trees of  $\mathcal{T}$  with only *set-* or *rel-*values at their leaves. These constructors are analogs of the boolean operations  $+$ ,  $\cdot$  and  $-$  (relative complement), used to delay evaluation.

The grammar rules may now use *set-* or *rel-*tree terms to define the value of coordinated expressions  $X'$  of category  $X \in \{A, N^{pl}, IV, NP\}$ :

$$X' \rightarrow X \text{ and } X [(X \odot X)], \quad (14.47)$$

$$| X \text{ or } X [(X \oplus X)], \quad (14.48)$$

$$| X \text{ but not } X [(X \ominus X)]. \quad (14.49)$$

We think of the elements of  $\mathcal{T}$  as second-class values. For example, while in  $\mathcal{P}'$  nouns are considered first-class values belonging to  $\mathcal{B}$  or  $\mathcal{R}$ , their coordinations are viewed as second-class values belonging to  $\mathcal{T}$ , such as *men*  $\ominus$  *women* in *(men but not women) like football*, where the set difference  $(\text{men} - \text{women}) \in \mathcal{B}$  clearly is wrong.

In a non-coordinating grammar rule, the evaluation function may



then recur along the tree structure of the values in  $\mathcal{T}$  of its coordinated constituents. For example, in

$$S' \rightarrow NP' IV' [distr_{NP,IV}(NP', IV')] \quad (14.50)$$

we may “distribute” the  $IV'$  to the coordinates of the subject- $NP$  and shift the connectives to the sentence level by clauses like

$$\begin{aligned} & distr_{NP,IV}((NP_1 \ominus NP_2), IV') \\ &= distr_{NP,IV}(NP_1, IV') \ominus distr_{NP,IV}(NP_2, IV'). \end{aligned}$$

When the value  $NP'$  in the subject is of the form  $e(N)$  or  $u(N)$  introduced by rules (14.46) and the value  $IV'$  is compound, the corresponding conjuncts can be interpreted by the boolean operations, as in

$$\begin{aligned} & distr_{NP,IV}(e(N), (IV_1 \odot IV_2)) \\ &= distr_{NP,IV}(e(N), (IV_1 \cdot IV_2)). \end{aligned}$$

When the value  $IV'$  is atomic, too,  $distr_{NP,IV}$  coincides with the function  $s$  of example 40, so that, for example,

$$distr_{NP,IV}(e(N), IV) = E(N \cdot IV).$$

Note that  $distr_{NP,IV}$  operates as a homomorphism on  $\mathcal{T}$  in its first argument. By using rule-specific functions  $distr$ , the constructors  $\odot, \oplus, \ominus$  may influence the meaning of expressions in a context-dependent way.

*Coordination of nouns and adjectives.* When several constituents in the same rule are coordinations, there is a question of relative scope of the connectives. If a constituent is to get narrow scope, the evaluation function has to recur on this constituent later. For example, to give the combinations of  $N$ 's higher precedence over those of classificatory adjectives  $A$ 's in modified nouns, for the rule

$$N' \rightarrow A' N' [distr_{A,N}(A', N')] \quad (14.51)$$

we define  $distr_{A,N}^{pl} : tree \times tree \rightarrow tree$  for  $\circ \in \{\odot, \oplus, \ominus\}$  as follows<sup>1</sup>:

$$\begin{aligned} distr_{A,N}^{pl}((A_1 \circ A_2), N) &= distr_{A,N}^{pl}(A_1, N) \circ distr_{A,N}^{pl}(A_2, N), \\ distr_{A,N}^{pl}(A, (N_1 \circ N_2)) &= distr_{A,N}^{pl}(A, N_1) \circ distr_{A,N}^{pl}(A, N_2), \\ distr_{A,N}^{pl}(A, N) &= A \cdot N, \quad \text{else.} \end{aligned} \quad (14.52)$$

---

<sup>1</sup>We restrict these to plural, since the corresponding clauses of  $distr_{A,N}^{sg}$  for singular might differ for *and* at least.

Then every interpretation of this grammar satisfies the equation

$$\begin{aligned}
& \llbracket (\textit{white but not grey}) (\textit{ants and elephants}) \rrbracket \\
&= \textit{distr}_{A,N}^{pl}(\textit{white} \ominus \textit{grey}, \textit{ant} \odot \textit{elephant}) \\
&= \textit{distr}_{A,N}^{pl}(\textit{white}, \textit{ant} \odot \textit{elephant}) \ominus \textit{distr}_{A,N}^{pl}(\textit{grey}, \textit{ant} \odot \textit{elephant}) \\
&= ((\textit{white} \cdot \textit{ant}) \odot (\textit{white} \cdot \textit{elephant})) \ominus ((\textit{grey} \cdot \textit{ant}) \odot \\
&\hspace{15em} (\textit{grey} \cdot \textit{elephant})) \\
&= \llbracket (\textit{white} (\textit{ants and elephants})) \textit{but not} \\
&\hspace{15em} (\textit{grey} (\textit{ants and elephants})) \rrbracket
\end{aligned}$$

Note, by the way, that this equality does not depend on expansions of coordinated  $N$ 's in sentences to sentence coordinations.

Ordinary Peirce grammar (cf. Böttner (1999), rule 57) interprets conjunction of nouns by set union  $+$ , so that, by laws of boolean algebra,

$$\begin{aligned}
\llbracket \textit{white} (\textit{ants and elephants}) \rrbracket &= \textit{white} \cdot (\textit{ant} + \textit{elephant}) \\
&= (\textit{white} \cdot \textit{ant}) + (\textit{white} \cdot \textit{elephant}).
\end{aligned}$$

This seems simpler than going via the above clause of  $\textit{distr}_{A,N}$  for classificatory adjectives, i.e.

$$\textit{distr}_{A,N}^{pl}(\textit{white}, \textit{ant} \odot \textit{elephant}) = (\textit{white} \cdot \textit{ant}) \odot (\textit{white} \cdot \textit{elephant}),$$

which keeps the component sets  $(\textit{white} \cdot \textit{ant})$  and  $(\textit{white} \cdot \textit{elephant})$  apart.

The reason for already keeping the components in  $\textit{ant} \odot \textit{elephant}$  apart is that the conjunction of common nouns is *not* set union in the scope of intensive adjectives ( $IA$ ), which denote partial order relations: in *big* (*ants and elephants*), we have to relativize –by a suitable operation  $r$ – the relation *big* to the sets *ant* and *elephant* separately and then take the union of the result, so that

$$\begin{aligned}
\llbracket \textit{big} (\textit{ants and elephants}) \rrbracket &= r(\textit{big}, \textit{ant}) + r(\textit{big}, \textit{elephant}) \\
&\neq r(\textit{big}, \textit{ant} + \textit{elephant}),
\end{aligned}$$

where only the set on the left contains some ants. In extended Peirce grammar, we can model this by a rule

$$N' \rightarrow IA' N' [\textit{distr}_{IA,N}(IA', N')] \quad (14.53)$$

with  $\textit{distr}_{IA,N}$  differing from  $\textit{distr}_{A,N}$  above by replacing (14.52) with

$$\textit{distr}_{IA,N}^{pl}(IA, N) := r(IA, N),$$

where the relativization  $r(R, B) := h(R \cdot (B \times B)) : 1$  is the domain of the “left half”  $h$  of the restriction of  $R$  to  $B^2$ . So, as with classificatory

---

<sup>2</sup>By the “left half”  $h(S)$  of a binary relation  $S$  we mean its edges from minimal

adjectives, we distribute the intensive adjective to conjuncts by

$$\text{distr}_{IA,N}^{pl}(big, ant \odot elephant) = r(big, ant) \odot r(big, elephant),$$

but do not and must not interpret noun conjunction as set union in the scope of an intensive adjective, as

$$r(big, ant + elephant) \neq r(big, ant) + r(big, elephant).$$

The relativization  $r$  operates in its second argument as a homomorphism on  $\mathcal{T}$ , but not on the first-class values  $\mathcal{B}$ . Note also that  $r$  allows us to interpret the attributive use of intensive adjectives while avoiding the comparison with artificial elements like “the medium-sized elephant” of Böttner (1999). Depending only on the values of nouns and the global *bigger-than*-relation with its converse *smaller-than*, we may obtain  $big$  (*white elephants*)  $\neq$  *white* (*big elephants*),  $big$  *ants*  $\subseteq$  *small animals*,  $small$  (*big elephants*)  $\subset$  *big elephants* etc.

Thus, noun conjunction is treated differently in the scope of classificatory and intensive adjectives; the pseudo-values of *set*-trees and *rel*-trees can be used to define the meaning of expressions containing coordinations by taking the context of the coordination into account.

### 14.3.2 Inverse linking

The question of relative scope for coordination connectives mentioned above arises also for coordinations of verbs applied to coordinated noun phrase arguments, as in *John or Mary could hear, but not see a man and a woman shout*. To give the subject *NP* widest scope, in

$$S \rightarrow NP VP [\text{distr}_{NP,VP}(NP, VP)]$$

we let  $\text{distr}_{NP,VP}$  recur on the *NP*-structure first, and in the *VP*-rule let the object *NP* have scope over the predicate (see (14.54),(14.55) below). Then  $\text{distr}_{NP,VP}$  also gives quantifiers in the subject wide scope over those in the object, subsuming the special cases of example 37.

(According to this scheme, active and passive of simple sentences with transitive verb and coordinated or quantified *NP*-complements

---

nodes,  $S \cap (\text{ran}(S) \times \text{ran}(S))$ , and the left half of  $S$  without edges from minimal or to maximal nodes,  $h(S \cap (\text{ran}(S) \times \text{dom}(S)))$ ). More precisely, we define  $h$  by

$$h(S) := \begin{cases} \emptyset, & \text{if } S = S \cdot (\check{S} : 1 \times S : 1) \\ S \cdot (\check{S} : 1 \times \check{S} : 1) + h(S \cdot (\check{S} : 1 \times S : 1)), & \text{else.} \end{cases}$$

In particular, if  $\leq$  is a finite linear order, then  $h(\leq)$  consists of the  $\leq$ -edges leaving the smaller half of the underlying set. Since  $h$  recurs along the inclusion  $\subseteq$  of relations and not along the structure of trees, we need to relax our definitions a bit to turn  $h(S)$  and hence  $r(R, B)$  into an extended Peirce term. Note that  $r(R, B_1 + B_2)$  is not the sum or any other algebraic function of  $r(R, B_1)$  and  $r(R, B_2)$ .

get different meanings, e.g. in *every child loves some adults* versus *some adults are loved by every child*.)

To handle scope relations that cannot be read off from the constituent structure, we have to relax the uniqueness of the term assignment  $e(r) = t_r$  and admit different  $t_r$  implementing different scope readings for the same grammar rule  $r$ . A case in question is the scope relation between dative and accusative object of a ditransitive verb; this is treated in Böttner (1999), rules R 105-115, by an extension of ordinary Peirce algebra to ternary relations. Here we consider a more difficult case of quantifier scope in nested *NPs*.

Peirce's operations model combinations of relational and common nouns like

$$\begin{aligned} \text{brother of a student} &= B : S \\ \text{friend of every brother of a student} &= \overline{F} : (B : S) = F \dagger \overline{(B : S)}. \end{aligned}$$

These amount to a reading where the syntactically innermost quantifier has narrowest scope:

$$\begin{aligned} &(\text{Some reader of (every paper of (some author))}) \text{ got bored} \\ &=_{a)} \exists x_1(\forall x_2(\exists x_3(A(x_3) \wedge P(x_2, x_3)) \rightarrow R(x_1, x_2)) \wedge B(x_1)). \end{aligned}$$

However, there is a different reading of *NP*'s of this form, where the syntactically innermost quantifier gets widest scope, the so-called *inverse linking* construction:

$$\begin{aligned} &(\text{Some reader of (every paper of (some author))}) \text{ got bored} \\ &=_{b)} \exists x_3(A(x_3) \wedge \forall x_2(P(x_2, x_3) \rightarrow \exists x_1(R(x_1, x_2) \wedge B(x_1)))). \end{aligned}$$

The inversion of quantifiers shows nicely when this is written with bounded quantifiers as

$$\begin{aligned} &(Q_1 RN_1 \text{ of } (Q_2 RN_2 \text{ of } (\dots \text{ of } (Q_n N) \dots)) IV \\ &= (Q_n x_n \in N) \dots (Q_2 x_2 RN_2 x_3)(Q_1 x_n RN_1 x_2) IV(x_n). \end{aligned}$$

In his review of Böttner (1999), Jäger (2004) notes that Peirce grammar seems unable to treat the inverse linking construction and could do so at best with infinitely many categories or grammar rules.

Let us see how this construction can be handled in extended Peirce grammar. To highlight the role of the quantifiers  $Q$  in the algebraic combinations, let  $\overset{Q}{c}$  (with  $c$  for copula) be the operation between two sets with  $Q$  in the subject:

$$A \overset{\forall}{c} B := U(\overline{A} + B), \quad A \overset{\exists}{c} B := E(A \cdot B).$$

Likewise, let  $\overset{Q}{\circ}$  be the module operation between a relation and a set

with  $Q$  in the object:

$$R \exists A := R : A, \quad R \forall A = R \dagger \bar{A}.$$

By means of these, we can translate the linking construction of  $NP$ 's in subject position as in:

$$\begin{aligned} (Q N) IV &\mapsto N \overset{Q}{\mathcal{C}} IV \\ (Q_1 RN_1 \text{ of } (Q_2 N)) IV &\mapsto N \overset{Q_2}{\mathcal{C}} (RN_1 \overset{Q_1}{\mathcal{Q}} IV) \\ (Q_1 RN_1 \text{ of } (Q_2 RN_2 \text{ of } (Q_3 N))) IV &\mapsto N \overset{Q_3}{\mathcal{C}} (RN_2 \overset{Q_2}{\mathcal{Q}} (RN_1 \overset{Q_1}{\mathcal{Q}} IV)). \end{aligned}$$

Thus, if the value of the subject  $NP$  in a sentence  $NP VP$  codes the syntactic structure of the  $NP$  together with the meaning of the nouns used, the modified Peirce grammar rule is

$$S \rightarrow NP VP [distr_{NP,VP}(NP, VP)] \quad (14.54)$$

where the function  $distr_{NP,VP}$  (extending the  $s$  of above) recurs along the  $NP$ -structure by:

$$\begin{aligned} distr_{NP,VP}((Q N), VP) &:= N \overset{Q}{\mathcal{C}} VP, \\ distr_{NP,VP}((Q RN \text{ of } NP), VP) &:= distr_{NP,VP}(NP, RN \overset{Q}{\mathcal{Q}} VP). \end{aligned}$$

Note that we need the converse of the relational noun. For example,

$$\begin{aligned} &(Some \text{ reader of } (every \text{ paper})) \text{ gets bored} \\ &= (\exists R \text{ of } (\forall P)) B \\ &= distr_{NP,VP}((\exists R \text{ of } (\forall P)), B) \\ &= distr_{NP,VP}((\forall P), \check{R} \exists B) \\ &= P \overset{\forall}{\mathcal{C}} (\check{R} \exists B) \\ &= \forall y(P(y) \rightarrow \exists x(\check{R}(y, x) \wedge B(x))) \\ &= \forall y(P(y) \rightarrow \exists x(R(x, y) \wedge B(x))) \end{aligned}$$

To handle the inverse linking construction in  $NPs$  in object position, we need relative operations  $\overset{Q}{\mathcal{Q}}$  determined by the quantifiers  $Q$ , combining two relations:

$$R \exists S := R ; S, \quad R \forall S := R \dagger \bar{S}.$$

A different translation is needed, which may return a set  $\neq 0, 1$ :

$$\begin{aligned} TV(Q N) &\mapsto TV \overset{Q}{\mathcal{Q}} N \\ TV(Q_1 RN_1 (Q_2 N_2)) &\mapsto (TV \overset{Q_1}{\mathcal{Q}} RN_1) (Q_2 N_2) \\ TV(Q_1 RN_1 (Q_2 RN_2 (Q_3 N))) &\mapsto ((TV \overset{Q_1}{\mathcal{Q}} RN_1) \overset{Q_2}{\mathcal{Q}} RN_2) \overset{Q_3}{\mathcal{Q}} N_3 \end{aligned}$$

Again, the innermost quantifier has widest scope. The grammar rule is

$$TVP \rightarrow TV NP [distr_{TV,NP}(TV, NP)] \quad (14.55)$$

where  $distr_{TV,NP}$  recurs along the structure of the object- $NP$  by

$$\begin{aligned} distr_{TV,NP}(TV, (Q N)) &:= TV \overset{Q}{?} N, \\ distr_{TV,NP}(TV, (Q RN \text{ of } NP)) &:= distr_{TV,NP}(TV \overset{Q}{?} RN, NP). \end{aligned}$$

For example, this gives

$$\begin{aligned} x \text{ likes } (every \text{ paper of } (some \text{ author})) &= x L (\forall P \text{ of } (\exists A)) \\ &= ((L \overset{\forall}{?} P) \overset{\exists}{?} A) (x) = \exists z (\forall y (L(x, y) \leftarrow P(y, z)) \wedge A(z)). \end{aligned}$$

Finally, note that for arguments  $(Q RN \text{ of } NP)$  where the  $NP$  is a conjunction, e.g. *a friend of John and Mary*, inverting the scopes by

$$distr_{NP,VP}((Q RN \text{ of } NP), VP) = distr_{NP,VP}(NP, RN \overset{Q}{?} VP)$$

produces the wrong reading with possibly different friends. The analysis where first the  $NP$  is combined with  $RN$  by

$$N' \rightarrow RN \text{ of } NP [distr_{RN,NP}(RN, NP)] \quad (14.56)$$

gives the reading with a common friend when  $distr_{RN,NP}$  recurs along its second argument with clauses like

$$\begin{aligned} distr_{RN,NP}(RN, (PN_1 \odot PN_2)) &= RN \overset{\forall}{?} (PN_1 + PN_2), \\ distr_{RN,NP}(RN, (PN_1 \oplus PN_2)) &= RN \overset{\exists}{?} (PN_1 + PN_2). \end{aligned}$$

Again, the coordination constructor influences the evaluation “later”.

#### 14.4 Comparison with other approaches

Peirce grammar differs from other grammatical theories in that meanings are first-order objects, abstract sets and relations, which can only be composed by algebraic operations. Extended Peirce grammar adds a further sort of meanings, finite trees of sets and relations, from which constituent meanings can be extracted. These “second-class” values have no ontological motivation — they only serve as intermediate stages in the evaluation of sentences, allowing us to give the context of an expression an access to the meaning *constituents* of the expression.

This makes a major difference both to the relational semantics of ordinary Peirce grammar and the higher-order semantics of Montague or categorial grammar: neither the boolean and peircean operations can be reversed, nor function application and abstraction. Thus, when a noun coordination like *ants and elephants* is interpreted by set union  $ant + elephant$  or pointwise function application  $\lambda x (ant(x) \vee elephant(x))$ , we cannot get back the meaning constituents *ant* and *elephant*. In extended Peirce grammar, we can get them back from the second-class value  $ant \odot elephant$  and hence we can correctly compute the relativization *big (ants and elephants)*.

Although the Boolean semantics of Keenan and Faltz (1985) is also higher-order, the difference to extended Peirce grammar is not that big. In Boolean semantics, every category  $C$  of conjoinable phrases comes with a boolean algebra  $\mathcal{B}_C$  of possible values, and the connectives *and*, *or*, and *not* for  $C$ -phrases are interpreted by the operations of  $\mathcal{B}_C$ . The point is that for categories of function type  $D/C$ , say, the algebra  $\mathcal{B}_{D/C}$  does not contain arbitrary functions from  $\mathcal{B}_C$  to  $\mathcal{B}_D$ , but boolean homomorphisms only; hence, each  $f \in \mathcal{B}_{D/C}$  has access to the meaning constituents of its argument, as in  $f(c_1 + c_2) = f(c_1) + f(c_2)$ .

While a Peirce algebra only has the two boolean algebras  $\mathcal{B}$  of sets and  $\mathcal{R}$  of binary relations, the tree algebra  $\mathcal{T}$  of an extended Peirce algebra can be chosen to subsume term algebras  $\mathcal{T}_C$  of phrases of category  $C$ . The evaluation terms in our grammar rules have mostly been used to define homomorphisms between such term algebras  $\mathcal{T}_C$ . We expect that a technical connection can be established between the quotients of  $\mathcal{T}_C$  modulo the boolean algebra axioms and the boolean algebras  $\mathcal{B}_C$  of Keenan and Faltz, provided we impose some invariance restrictions on the evaluation terms. But, unlike Boolean semantics, extended Peirce grammar does not assume that all categories of conjoinable expressions are interpreted in an ontologically well-motivated boolean algebra.

There is a minor difference to Boolean semantics concerning the implicit scope rules of natural language. Consider simple sentences built by applying a boolean combination (BC) of transitive verbs to boolean combinations of noun phrases,

$$BC(TVs)(BC(NPs), BC(NPs)).$$

We expect this to be evaluated by a boolean algebra homomorphism in each coordinate, i.e. for *atomic* expressions  $v, e$ , we have

$$v(e, BC(c_1, \dots, c_r)) = BC(v(e, c_1), \dots, v(e, c_r))$$

with the same boolean combination, and likewise for subject and predicate. What remains not completely determined, e.g. in German, is the relative scope of the quantifiers and connectives in the predicate and argument positions. Keenan and Faltz (1985) built specific scope rules into their interpretation, which we also did in the examples, but in general there are different readings which may be explicitly marked, as in *John and Mary are working on a (common) paper* vs. *John and Mary are working on a paper (each)*. In general, a non-deterministic evaluation is needed that chooses between various possible scope relations.

The simple algebraic semantics of Peirce grammar clearly poses severe restrictions, viz. not all first-order formulas built from unary and binary relations are expressible as Peircean terms. Extensions to rela-

tions of arity  $\geq 3$  are possible, but an equational axiomatization with additional primitives, like argument permutation, seems missing. In Peirce grammar, one can define the *at least two*-quantifier  $\exists^{\geq 2}$  via the diversity relation  $\bar{I}$  and thus handle a distributive reading of plurals. But it is impossible to define *at least n* for  $n \geq 3$ , unless, for example, one admits  $n$ -ary relations with a constant for  $n$ -ary diversity.

What about generalized quantifiers like *most*? In the algebra  $\mathcal{P}_V$  of all subsets and relations on  $V$  one can define the copula-, module- and relation-operations  $\overset{M}{c}$ ,  $\overset{M}{M}$ ,  $\overset{M}{M}$  from the set-theoretic definition  $M(A, B) : \iff |A \cdot B| > |A \cdot \bar{B}|$  of the quantifier  $M$ , and then evaluate simple sentences of the form  $(Q_1 N_1) TV (Q_2 N_2)$  as before by  $N_1 \overset{Q_1}{c} (TV \overset{Q_2}{c} N_2)$ . But an algebraic treatment of *most* would need an equational axiomatization of  $\overset{M}{c}$ ,  $\overset{M}{M}$ ,  $\overset{M}{M}$ , which does not seem to exist.

Up to now, Peirce grammar has, to my knowledge, only been used to model extensional features of natural language. In particular, it is not clear how to handle verbs with propositional arguments.

Concerning syntax, the proposed extension of Peirce grammar allows us to handle recursive syntactic constructions and extensively use traditional constituent structures of context-free grammars. This is arguably a step away from Peirce's original emphasis on  $n$ -ary relations and their combinations, which is closer in spirit to dependency grammar.

We have only dealt with coordination of words or constituents. In transformational grammar, "non-constituent"-coordinations have been considered the result of transformations of coordinations, such as right or left peripheral extraction or argument cluster coordination:

$$(RPE) (\alpha\gamma) \text{ Conj } (\beta\gamma) \rightsquigarrow [\alpha \text{ Conj } \beta] \gamma$$

$$(LPE) (\gamma\alpha) \text{ Conj } (\gamma\beta) \rightsquigarrow \gamma [\alpha \text{ Conj } \beta]$$

$$(ACC) (\alpha\gamma\beta) \text{ Conj } (\alpha'\gamma\beta') \rightsquigarrow \alpha\gamma\beta \text{ Conj } [\alpha'\beta'],$$

where restrictions on the categorial status of the expression sequences  $\alpha, \beta, \gamma, \alpha', \beta'$  have been the subject of debate. At least simple cases of these can be handled in Peirce grammar using "flat" syntax rules. For example, the reading of

$$(RPE) [all \text{ men } like, \text{ but some women dislike}] \text{ some jokes}$$

where *some jokes* has wide scope and cannot be distributed "back", gets its value by the Peirce term  $E(J \cdot ((L^{\forall} M) \cdot (\bar{L}^{\exists} W)))$ , but I did not consider more complicated *NPs* here. For examples like

$$(ACC) \text{ Mary wrote a paper and } [(John \text{ and Bill}) \text{ (two reviews)}],$$

constructed by a "flat" syntax-rule

$$S \rightarrow NP_1^{nom} TV NP_2^{acc} \text{ Conj } NP_3^{nom} NP_4^{acc} [t],$$



the value can be defined by the extended Peirce-term

$$t := f(NP_1, TV, NP_2) \wedge f(NP_3, TV, NP_4)$$

where  $f$  is one of the following, depending on whether coordinations and quantifiers in the subject are to have wide scope or not:

$$\begin{aligned} f(NP_1, TV, NP_2) &:= \text{distr}_{NP,VP}(NP_1, \text{distr}_{TV,NP}(TV, NP_2)), \\ f(NP_1, TV, NP_2) &:= \text{distr}_{NP,VP}(NP_2, \text{distr}_{TV,NP}(TV^\smile, NP_1)). \end{aligned}$$

Note that the value  $TV$  from the initial constituent  $(NP_1 TV NP_2)$  is reused to evaluate the non-constituent  $[NP_3 NP_4]$ .

This sketchy treatment of non-constituent coordination seems more in line with ideas used in HPSG, see Beavers and Sag (2004), than with those of combinatory categorial grammar, which rely on function types. A treatment with product types is proposed in Houtman (1994).

#### 14.5 Conclusion and open problems

We have proposed a solution to what was called the ‘one big thread to the entire project of relational grammar’ in Jäger (2004), recursivity. Essentially, we turn syntactic structure to values and admit evaluation of expressions by structural recursion on those values, keeping the grammar finite. For programming languages, where programs are large, this would give values of intolerable size. For natural language, where sentences are short, the tree values are shallow. However, if their leaves are sets and relations in extension, we may also have a size and hence complexity problem. So we better interpret the grammar in a Peirce algebra  $\mathcal{P}$  where sets and relations are abstract elements, not in a full algebra  $\mathcal{P}_V$ . It deserves to be investigated if we can turn the definable elements of a given  $\mathcal{P}_V$  into an abstract interpretation  $\mathcal{P}$  with efficiently computable operations.

The examples given show that our proposal is a useful and necessary modification of Peirce grammar, but it needs to be clarified what we have to pay. By adding the sort of trees we loose the equational axiomatizability of Peirce algebra, since inequational Horn formulas are needed to state the freeness assumptions for constructors (c.f. Maher (1988)). Using the semantic annotations of grammar rules, we may check the semantic validity of inferences based on grammatical form. While this can be done by equational reasoning for Peirce grammar, for extended Peirce grammar we will also need to reason by induction on the tree structure, which seems acceptable. A less tolerable step away from the equational theory of Peirce algebra is our use of recursion on the subsumption order to define the relativization operation.

Our extension may be (mis)used to deviate radically from the strictly

bottom-up way of evaluation in ordinary Peirce grammar: one can delay evaluation to first-class values until the complete constituent structure of an expression is mirrored in an intermediate second-class value.

Finally, trees as values offer an advantage over higher-order semantic values of other frameworks: one may look for decidable relations between trees that imply semantic equivalence. This also can be relevant for checking the validity of grammatical inferences.

### Acknowledgement

I wish to thank the referees for critical questions and Michael Böttner for copies of papers on Peirce and for keeping in touch over the years.

### References

- Beavers, John and Ivan A. Sag. 2004. Coordinate ellipsis and apparent non-constituent coordination. In S. Müller, ed., *Proceedings of the HPSG-04 Conference*. Stanford University, CSLI Publications.
- Böttner, Michael. 1999. *Relationale Grammatik*. Tübingen: Max Niemeyer Verlag.
- Brink, Chris, Katharina Britz, and Renate A. Schmidt. 1994. Peirce algebras. *Formal Aspects of Computing* 6:339–358.
- Houtman, Joop. 1994. *Coordination and Constituency. A Study in Categorical Grammar*. Ph.D. thesis, Rijksuniversiteit Groningen.
- Jäger, Gerhard. 2004. Book Review of “Relationale Grammatik” by Michael Böttner. *Journal of Logic, Language and Information* 13(4):521–525.
- Keenan, Edward L. and Leonard M. Faltz. 1985. *Boolean Semantics for Natural Language*. Dordrecht: D. Reidel.
- Maher, M. J. 1988. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *3rd IEEE Symposium on Logic in Computer Science*, pages 348–357. IEEE Computer Society Press.
- Montague, Richard. 1974. The proper treatment of quantification in ordinary english. In R. Thomason, ed., *Formal Philosophy. Selected Papers of Richard Montague*. Yale Univ. Press.
- Peirce, Charles Sanders. 1932. *Collected Papers*, vol. III, chap. The Logic of Relatives, pages 228–345. Cambridge, Mass.: Harvard University Press. (Originally published in: *The Monist*, vol.7, pp.161–217, 1897).
- Suppes, Patrick. 1976. Elimination of quantifiers in the semantics of natural language by use of extended relation algebras. *Revue Internationale de Philosophie* 117/118:243–259.

---

## Strong connectivity hypothesis and generative power in TAG

ALESSANDRO MAZZEI, VINCENZO LOMBARDO AND  
PATRICK STURT <sup>†</sup>

### Abstract

Dynamic grammars are relevant for psycholinguistic modelling and speech processing. However, formal treatments of dynamic grammars are rare, and there are few studies that examine the relationship between dynamic and phrase structure grammars. This paper introduces a TAG related dynamic grammar (DVTAG) together with a study of its expressive power. We also shed a new perspective on the *wrapping* operation in TAG.

**Keywords** DYNAMIC GRAMMARS, TAG, WRAPPING, INCREMENTALITY

### 15.1 Introduction

Incrementality is a feature of human language analysis that is very relevant for language modeling and speech processing. An incremental processor takes a string in left-to-right order and starts to build a syntactic and semantic representation before reaching the end of the sentence. The strong connectivity hypothesis is a parsimonious way to formalize the incrementality of the syntactic process: people incorporate each word into a single, totally connected syntactic structure before any further words follow (Stabler, 1994). Strong connectivity is supported by some psycholinguistic evidence (Kamide et al., 2003, Sturt and Lombardo, 2005).

Some traditional approaches to syntactic processing (both derivation

---

<sup>†</sup>We thank an anonymous reviewer and Jim Rogers for many useful comments.

and parsing) use a generative grammar and implement connectivity in the derivation/parsing algorithm (Abney and Johnson, 1991). An alternative approach is to change the perspective of investigation and model the strongly connected syntactic process in the grammar formalism. Dynamic systems model cognitive processes as sequences of states connected through transitions (Gelder, 1998). Like in automata theory, a state encodes the syntactic process until some point in the input string; then a string (one or several words) realizes the transition to the subsequent state. However, in dynamic approaches the automaton is not the result of a compilation of a generative grammar, but the formalism itself (cf. Woods (1986)), without any necessary involvement of generative rules.

The dynamic approach is not new in mathematical linguistics. Both left-associative grammars (Hausser, 1992) and dynamic dependency grammars (Milward, 1994) are examples of *dynamic grammars*. The common traits of dynamic grammars are the definition of a (non necessarily finite) recursive set of states, a finite subset of initial states, a finite set of axioms or rules. In addition to these, the definition of a dynamic grammar includes a way to assemble multiple applications of the axioms (or rules), usually an explicit or implicit specification of sequencing. However, an interesting abstraction in Milward's specification, called a deduction rule, allows several possibilities of assembling, and so provides a dynamic account of non-constituent coordination (Milward, 1994). In order to specify what are the legal strings of a language, both Hausser and Milward indicate a finite set of final states (in fact both approaches originate in a categorial grammar paradigm). We can say that left-associative grammars and dynamic dependency grammars incorporate the derivation process entirely. However, this is not strictly necessary if we design the axioms (or rules) on a generative basis. In this paper we take a hybrid dynamic-generative approach: states are partial structures that are licensed by a recursive process from a finite basic lexicon; transitions extend the basic structures through a finite number of generative operations. A state  $S_i$  represents a partial structure that spans the left fragment of a sentence  $w_1 \dots w_i \dots w_n$ . In particular, we propose a Dynamic Version of TAG (DVTAG). DVTAG is a dynamic grammar that fulfills the strong connectivity hypothesis, has interesting consequences for constituency definition and linguistic descriptions, and shares basic tenets with LTAG (i.e. lexicalization, adjoining and extended domain of locality) (Mazzei, 2005). We provide a formal definition of DVTAG and a study of its expressive power. We show that DVTAG is a mildly context-sensitive formalism and is strongly equivalent to a restricted LTAG formalism, called *dynamic*

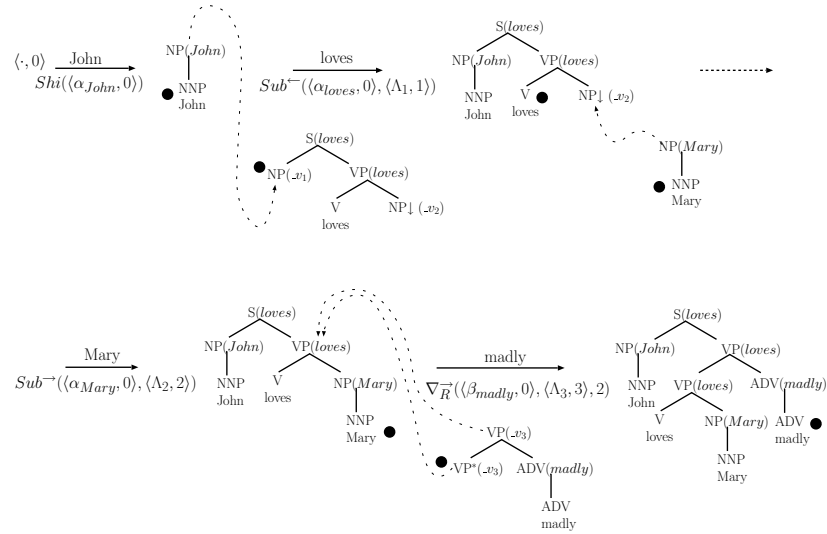


FIGURE 1: The DVTAG derivation of the sentence *John loves Mary madly*.

LTAG. Moreover we show that the introduction of wrapping operation, an alternative view of adjoining based on *flexible composition* of the derivation tree (Joshi, 2004), increases the generative power of DVTAG and dynamic LTAG.

### 15.2 Informal introduction to DVTAG

In Fig. 1 we can see the DVTAG derivation of the sentence *John loves Mary madly*. Like LTAG (Joshi and Schabes, 1997), a Dynamic Version of Tree Adjoining Grammar (DVTAG) consists of a set of elementary trees, divided into initial trees and auxiliary trees, and attachment operations for combining them. Lexicalization is expressed through the association of a lexical *anchor* with each elementary tree. To encode lexical dependencies, each node in the elementary tree is augmented with a feature indicating the lexical head that projects the node. The head variable is a variable in logic terms:  $v_3$  will be unified with the constant *loves* in the derivation of Fig. 1. The derivation process in DVTAG builds a constituency tree by combining the elementary trees via operations that are illustrated below. DVTAG implements the incremental process by constraining the *derivation process* to be a series of steps in which an elementary tree is combined with the partial tree

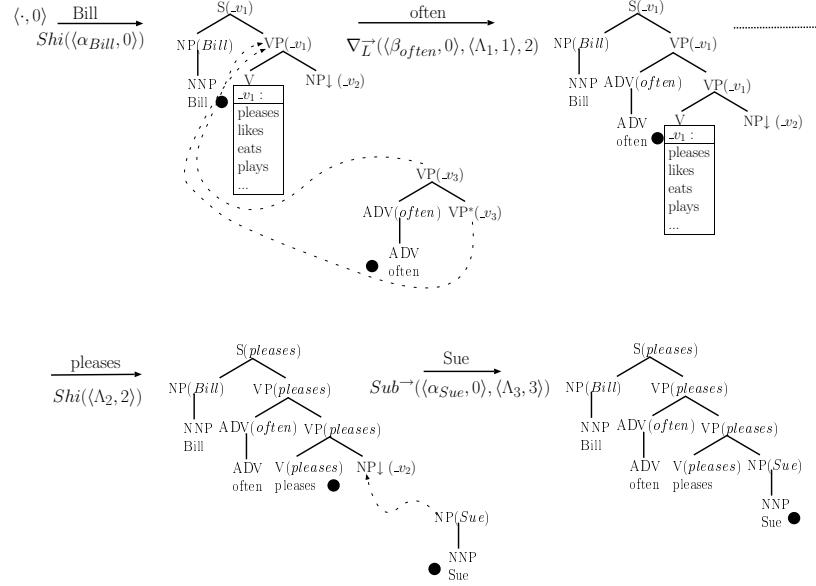


FIGURE 2: The DVTAG derivation of the sentence *Bill often pleases Sue*.

spanning the left fragment of the sentence. The result of a step is an updated partial structure. Specifically, at the processing step  $i$ , the elementary tree anchored by the  $i$ -th word in the sentence is combined with the partial structure spanning the words from 1 to  $i - 1$  positions; the result is a partial structure spanning the words from 1 to  $i$ . In DV-TAG the derivation process starts from an elementary tree anchored by the first word in the sentence and that does not require any attachment that would introduce lexical material on the left of the anchor (such as in the case that a Substitution node is on the left of the anchor). This elementary tree becomes the first left-context that has to be combined with some elementary tree on the right. At the end of the derivation process the left-context spans the whole sentence, and is called the *derived tree*: the last tree of Fig.1 is the derived tree for the sentence *John loves Mary madly*.

In DVTAG we always combine a left context with an elementary tree, then there are seven attachment operations. Adjoining is split into two operations: *adjoining from the left* and *adjoining from the right*. The type of adjoining depends on the position of the lexical material in-

troduced by the auxiliary tree with respect to the material currently dominated by the adjoined node (which is in the left-context). In Fig. 1 we have an adjoining from the right in the case of the left auxiliary tree anchored by *madly*, and in Fig. 2 we have an adjoining from the left in the case of the left auxiliary tree anchored by *often*. Inverse operations account for the insertion of the left-context into the elementary tree. In the case of *inverse substitution* the left-context replaces a substitution node in the elementary tree; in the case of *inverse adjoining from the left* and *inverse adjoining from the right*, the left-context acts like an auxiliary tree, and the elementary tree is split because of the adjoining of the left context at some node. In Fig. 1 we have an inverse substitution in the case of the initial tree anchored by *John*. Finally, the *shift* operation either scans a lexical item which has been already introduced in the structure or derives a lexical item from some predicted preterminal node. The grounding of the variable  $\_v_1$  in Fig. 2 is an example of shift.

It is important to notice that, during the derivation process, not all the nodes in the left-context and the elementary tree are accessible for performing operations: given the  $i - 1$ -th word in the sentence we can compute a set of accessible nodes in the left-context (the *left-context fringe*); also, given the lexical anchor of the elementary tree, that in the derivation process matches the  $i$ -th word in the sentence, we can compute a set of accessible nodes in the elementary tree (the *elementary tree fringe*). To take into account this feature, the elementary tree in DVTAG are dotted tree, i.e. a couple  $\langle \gamma, i \rangle$  formed by a tree  $\gamma$  and an integer  $i$  denoting the accessible fringe<sup>1</sup> of the tree (Mazzei, 2005). The DVTAG derivation process requires the full connectivity of the left-context at all times. The extended domain of locality provided by LTAG elementary trees appears to be a desirable feature for implementing full connectivity. However, each new word in a string has to be connected with the preceding left-context, and there is no *a priori* limit on the amount of structure that may intervene between that word and the preceding context. For example in the sentence *Bill often pleases Sue* there is an intervening modifier between an argument and its predicative head (Fig.2). The elementary tree *Bill* is linguistically motivated up to the NP projection; the rest of the structure depends on connectivity. These extra nodes are called *predicted nodes*. A predicted preterminal node is referred by a set of lexical items, that represent a *predicted head*. So, the extended domain of locality available in LTAG has to be further extended. In particular, some structures have to be

---

<sup>1</sup>In the figures we represent the integer using a dot.

predicted as soon as there is some evidence from arguments or modifiers on the left.

### 15.3 Formalizing DVTAG

Now we provide a formal definition of DVTAG using strings of a formal language (a, b c, etc). In the definitions and proofs presented here, we use DVTAG elementary trees without defining the notions of underspecified heads and without the specification of the head-variable (see previous section). In fact these two features are important for linguistic description, but are irrelevant with respect to the generative power. First, we provide some general terminology. Let  $\Sigma$  be an alphabet of terminal symbols, and  $V$  an alphabet of non-terminal symbols.  $a, b, c, d, \dots \in \Sigma$  indicate terminal symbols, where  $\varepsilon$  is the null string.  $A, B, C, D, \dots \in V$  indicate non-terminal symbols,  $x, y, w, z \in \Sigma^*$  are strings of terminals:  $|x|$  is the length of the string  $x$ . We use  $w_i$  to denote the  $i$ -th word in the sentence  $w$ . We denote initial trees with  $\alpha$ , auxiliary trees with  $\beta$ , derived and generic trees with  $\gamma, \delta, \zeta, \Lambda^2$ ;  $\mathcal{N}$  denotes a node belonging to a tree,  $label(\mathcal{N})$  the label of this node.  $foot(\beta)$  returns the foot node of an auxiliary tree  $\beta$ .  $YIELD(\gamma)$  is a function that returns the frontier of  $\gamma$  with the exception of foot nodes (i.e. overt terminal nodes and substitution nodes). Finally,  $YIELD_i(\gamma)$  is the function that returns the  $i$ -th element of the  $YIELD(\gamma)$  whether  $1 < i < |YIELD(\gamma)|$ , otherwise it is undefined. As usual in TAG, to denote the position of a node  $\mathcal{N}$  in a tree, we use its *Gorn Address*. Since DVTAG is a lexicalized formalism, each elementary tree is anchored by some lexical element. We refer to the leftmost lexical element in a tree with the expression *left-anchor*. We call a tree  $\gamma$  *direct* if  $YIELD_1(\gamma)$  is its left-anchor, moreover we call  $\gamma$  *inverse* if  $YIELD_1(\gamma)$  is a substitution node and  $YIELD_2(\gamma)$  is its left-anchor. While initial trees are not different from the ones in LTAG, we distinguish three types of auxiliary trees: *left auxiliary trees*  $\mathcal{A}_L$  as auxiliary trees where the foot node is the rightmost node of the frontier, *right auxiliary trees*  $\mathcal{A}_R$  as auxiliary trees where the foot node is the leftmost node of the frontier, *wrapping auxiliary trees*  $\mathcal{A}_W$  as auxiliary trees where the frontier extends on both the left and the right of the foot node. We introduce two useful notions that are borrowed from parsing algorithm tradition, namely *dotted tree* and *fringe*.

**Definition 35** A **dotted tree** is a pair  $\langle \gamma, i \rangle$  where  $\gamma$  is a tree and  $i$  is an integer such that  $i \in 0 \dots |YIELD(\gamma)|$ .

---

<sup>2</sup>Conventionally we always use  $\Lambda$  to indicate the left-context.



Given a set of (LTAG) elementary tree  $\mathcal{E}$  with  $\langle \mathcal{E}, 0 \rangle$ , we will indicate the set of dotted trees such that if  $\gamma \in \mathcal{E}$ , then  $\langle \gamma, 0 \rangle \in \mathcal{E}$ ; we represent a generic element of  $\langle \mathcal{E}, 0 \rangle$  with the *underspecified* dotted tree  $\langle \cdot, 0 \rangle$ . The fringe of  $\langle \gamma, i \rangle$  is a set of nodes (split in a left fringe and a right fringe) that includes all the nodes that are accessible for operating upon, that is the fringe defines the domain of the attachment operations.

**Definition 36** The **left-fringe** (Lfringe) of  $\langle \gamma, i \rangle$  is the set difference between the set of nodes on the path from  $YIELD_i(\gamma)$  to root and the set of nodes on the path from  $YIELD_{i+1}(\gamma)$  to root. The **right-fringe** (Rfringe) of  $\langle \gamma, i \rangle$  is the set difference between the set of nodes on the path from  $YIELD_{i+1}(\gamma)$  to root, and the set of nodes on the path from  $YIELD_i(\gamma)$  to root. Moreover, if there is a null lexical item  $\varepsilon$  on the left (on the right) of  $YIELD_i(\gamma)$  ( $YIELD_{i+1}(\gamma)$ ), all the nodes from  $\varepsilon$  up to the lowest common ancestor of  $\varepsilon$  and  $YIELD_i(\gamma)$  ( $YIELD_{i+1}(\gamma)$ ), belong to the right and left fringes.

In Fig. 3 left and right fringes are depicted as ovals in the dotted trees. Now we define seven attachment operations on dotted trees: two substitutions (similar to LTAG substitution), four adjoiningings (similar to LTAG adjoininging), and one shift.

The **Shift** operation  $Shi(\langle \gamma, i \rangle)$  is defined on a single dotted tree  $\langle \gamma, i \rangle$  and returns the dotted tree  $\langle \gamma, i + 1 \rangle$ . It can be applied only if a terminal symbol belongs to the right fringe of  $\langle \gamma, i \rangle$ : the dot is shifted on the right of the next overt lexical symbol of the yield  $YIELD_{i+1}(\gamma)$  (Fig. 3-a).

The **Substitution** operation  $Sub^\rightarrow(\langle \alpha, 0 \rangle, \langle \gamma, i \rangle)$  is defined on two dotted trees: a dotted tree  $\langle \gamma, i \rangle$  and an initial direct dotted tree  $\langle \alpha, 0 \rangle$ . If there is in the right fringe of  $\langle \gamma, i \rangle$  a substitution node  $\mathcal{N}$  and  $label(\mathcal{N}) = label(root(\alpha))$ , the operation returns a new dotted tree  $\langle \delta, i + 1 \rangle$  such that  $\delta$  is obtained by grafting  $\alpha$  in  $\mathcal{N}$  (Fig. 3-b).

The **Inverse Substitution** operation  $Sub^\leftarrow(\langle \zeta, 0 \rangle, \langle \gamma, i \rangle)$  is defined on two dotted tree: a dotted tree  $\langle \gamma, i \rangle$  and an inverse elementary dotted tree  $\langle \zeta, 0 \rangle$ . If  $root(\gamma)$  belongs to the left fringe of  $\langle \gamma, i \rangle$ , and there is a substitution node  $\mathcal{N}$  belonging to the right fringe of  $\langle \zeta, 0 \rangle$  such that  $label(\mathcal{N}) = label(root(\gamma))$ , the operation returns a new dotted tree  $\langle \delta, i + 1 \rangle$  such that  $\delta$  is obtained by grafting  $\gamma$  in  $\mathcal{N}$  (Fig. 3-c).

The **Adjoining from the left** operation  $\nabla_L^\rightarrow(\langle \beta, 0 \rangle, \langle \gamma, i \rangle, add)$  is defined on two dotted trees: a dotted tree  $\langle \gamma, i \rangle$  and a direct left or wrapping auxiliary dotted tree  $\langle \beta, 0 \rangle$ . If there is in the right fringe of  $\langle \gamma, i \rangle$  a non-terminal node  $\mathcal{N}$  such that  $label(\mathcal{N}) = label(root(\beta))$ , the operation returns a new dotted tree  $\langle \delta, i + 1 \rangle$  such that  $\delta$  is obtained by grafting  $\beta$  in  $\mathcal{N}$  (Fig. 3-d).

The **Adjoining from the right** operation  $\nabla_R^-(\langle\beta, 0\rangle, \langle\gamma, i\rangle, add)$  is defined on two dotted trees: a dotted tree  $\langle\gamma, i\rangle$  and a direct right auxiliary dotted tree  $\langle\beta, 0\rangle$ . If there is in the left fringe of  $\langle\gamma, i\rangle$  a non-terminal node  $\mathcal{N}$  such that  $label(\mathcal{N}) = label(root(\beta))$ , the operation returns a new dotted tree  $\langle\delta, i + 1\rangle$  such that  $\delta$  is obtained by grafting  $\beta$  in  $\mathcal{N}$  (Fig. 3-e).

The **Inverse adjoining from the left** operation  $\nabla_L^-(\langle\zeta, 0\rangle, \langle\gamma, i\rangle, add)$  is defined on two dotted trees: a dotted tree  $\langle\gamma, i\rangle$  and a direct elementary dotted tree  $\langle\zeta, 0\rangle$ . If  $foot(\gamma)$  belongs to the fringe of  $\langle\gamma, i\rangle$ , and there is a node  $\mathcal{N}$  belonging to right fringe of  $\langle\zeta, 0\rangle$  such that  $label(\mathcal{N}) = label(foot(\gamma))$ , the operation returns a new dotted tree  $\langle\delta, i + 1\rangle$  such that  $\delta$  is obtained by grafting  $\gamma$  in  $\mathcal{N}$  (Fig. 3-f).

**Inverse adjoining from the right** operation  $\nabla_R^-(\langle\zeta, 0\rangle, \langle\gamma, i\rangle, add)$  is defined on two dotted tree:  $\langle\gamma, i\rangle$  and the direct elementary dotted tree  $\langle\zeta, 0\rangle$ .  $\langle\zeta, 0\rangle$  has a null lexical item ( $\varepsilon$  node) as first leaf. If  $root(\gamma)$  belongs to the fringe of  $\langle\gamma, i\rangle$ , and there is a node  $\mathcal{N}$  belonging to left fringe of  $\langle\zeta, 0\rangle$  such that  $label(\mathcal{N}) = label(root(\gamma))$ , the operation returns a new dotted tree  $\langle\delta, i + 1\rangle$  such that  $\delta$  is obtained by grafting  $\gamma$  in  $\mathcal{N}$  (Fig. 3-g).

Using the definitions given above, we can formally define DVTAG.

**Definition 37** Let  $\langle\mathcal{E}, 0\rangle$  be a finite set of elementary dotted trees, a DVTAG  $G(\langle\mathcal{E}, 0\rangle)$  is a triple consisting of:

(1) **Set of initial left-context:** the finite set of direct dotted trees  $\langle\Lambda_0, 0\rangle \in \langle\mathcal{E}, 0\rangle$ .

(2) **Set of axiom schemata:** there are three kinds of schemata to update the left-contexts:

1.

$$\langle\cdot, 0\rangle \xrightarrow[\text{Shi}(\langle\Lambda_1, 0\rangle)]{a} \langle\Lambda_1, 1\rangle$$

where the terminal symbol  $\mathbf{a}$  is the left-anchor of the initial left-context  $\langle\Lambda_1, 0\rangle$ .

2.

$$\langle\Lambda_i, i\rangle \xrightarrow[\text{Shi}(\langle\Lambda_i, i\rangle)]{a} \langle\Lambda_{i+1}, i + 1\rangle$$

where the terminal symbol  $\mathbf{a}$  belongs to the right fringe of  $\langle\Lambda_i, i\rangle$  and  $i$  ranges over the natural numbers.

3.

$$\langle\Lambda_i, i\rangle \xrightarrow[\text{op}_a]{a} \langle\Lambda_{i+1}, i + 1\rangle$$

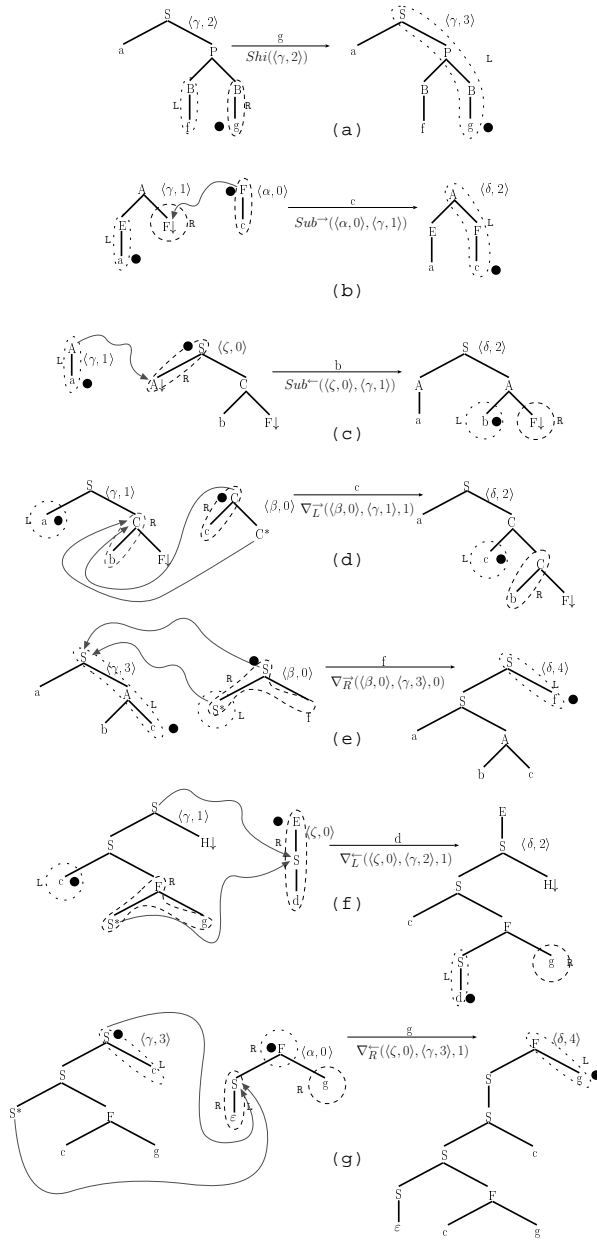


FIGURE 3: Operations in DVTAG.

where

$$op_a = \begin{cases} Sub^{\rightarrow}(\langle \alpha, 0 \rangle, \langle \Lambda_i, i \rangle) \\ Sub^{\leftarrow}(\langle \zeta, 0 \rangle, \langle \Lambda_i, i \rangle) \\ \nabla_L^{\rightarrow}(\langle \beta, 0 \rangle, \langle \Lambda_i, i \rangle, add) \\ \nabla_R^{\rightarrow}(\langle \beta, 0 \rangle, \langle \Lambda_i, i \rangle, add) \\ \nabla_L^{\leftarrow}(\langle \zeta, 0 \rangle, \langle \Lambda_i, i \rangle, add) \\ \nabla_R^{\leftarrow}(\langle \zeta, 0 \rangle, \langle \Lambda_i, i \rangle, add) \end{cases}$$

and  $\mathbf{a}$  is the left-anchor of  $\langle \alpha, 0 \rangle$ ,  $\langle \zeta, 0 \rangle$ ,  $\langle \beta, 0 \rangle$ ,  $\langle \zeta, 0 \rangle$  respectively. For  $\nabla_L^{\rightarrow}$ ,  $\nabla_R^{\rightarrow}$ ,  $\nabla_L^{\leftarrow}$ ,  $\nabla_R^{\leftarrow}$ ,  $\mathbf{add}$  is the Gorn address of the adjoining node, and where  $i$  ranges over the natural numbers.

**(3) Deduction rule** that specifies the sequentiality of the derivation process

$$\frac{\langle \Lambda_i, i \rangle \xrightarrow[op_{a_1} \dots op_{a_n}]{a_1 \dots a_n} \langle \Lambda_j, j \rangle \quad \langle \Lambda_j, j \rangle \xrightarrow[op_{b_1} \dots op_{b_m}]{b_1 \dots b_m} \langle \Lambda_k, k \rangle}{\langle \Lambda_i, i \rangle \xrightarrow[op_{a_1} \dots op_{a_n} \ op_{b_1} \dots op_{b_m}]{a_1 \dots a_n \ b_1 \dots b_m} \langle \Lambda_k, k \rangle}$$

DVTAG defines implicitly the infinite space of derivable left-contexts, i.e. the (recursive) set of left-contexts that are reachable with a finite number of operations starting from an initial left-context. In order to define the language generated by a DVTAG we have to define the notion of final left-context. Both dynamic dependency grammars (Milward, 1994) and left-associative grammars (Hausser, 1992) explicitly define the finite set of the final state. In contrast, in our hybrid dynamic-generative approach we define the set of final left-contexts on the basis of their structure. We call a left-context  $\langle \Lambda_n, n \rangle$  final if  $\Lambda_n$  has no substitution or foot nodes in the yield.

**Definition 38** A string  $w_1 \dots w_n$  ( $n$  natural number) is generated by a DVTAG  $G(\langle \mathcal{E}, 0 \rangle)$  if and only if  $\langle \cdot, 0 \rangle \xrightarrow[op_{w_1} \dots op_{w_n}]{w_1 \dots w_n} \langle \Lambda_n, n \rangle$  is derivable in  $G(\langle \mathcal{E}, 0 \rangle)$  and  $\langle \Lambda_n, n \rangle$  is final. Moreover, a tree  $\Lambda_n$  ( $n$  natural number) is generated by a DVTAG  $G(\langle \mathcal{E}, 0 \rangle)$  if and only if  $\langle \cdot, 0 \rangle \xrightarrow[op_{w_1} \dots op_{w_n}]{YIELD(\Lambda_n)} \langle \Lambda_n, n \rangle$  is derivable in  $G(\langle \mathcal{E}, 0 \rangle)$  and  $\langle \Lambda_n, n \rangle$  is final.

In contrast to the standard definition of generative grammars, the Definition 37 describes the derivation process as part of the definition of the grammar. Note that the notion of left-context corresponds to the notion of sentential form of the usual generative grammars. A dynamic grammar explicitly includes the derivability of a left-context, whereas this notion is outside of the grammar in the standard generative formalization. The dynamic approach allows us to define several deduction rules for derivation: Milward used this feature to take into account

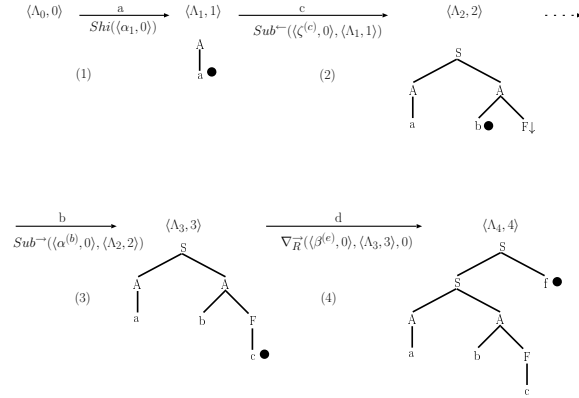


FIGURE 4: A DVTAG derivation for the sentence “abcf”.

the non-constituent coordination in the dynamic dependency grammars (Milward, 1994). Finally, we formalize the notion of *derivation chain*:

**Definition 39** Given a string  $w_1 \dots w_n$  derivable by  $G(\langle \mathcal{E}, 0 \rangle)$ , a **derivation chain**  $d$  is the sequence of left-contexts  $\langle \Lambda_0, 0 \rangle \langle \Lambda_1, 1 \rangle \dots \langle \Lambda_n, n \rangle$  such that  $\langle \Lambda_0, 0 \rangle$  is an initial left-context,  $\langle \Lambda_n, n \rangle$  is a final left-context and  $\langle \Lambda_{i-1}, i-1 \rangle \xrightarrow[\text{op } w_i]{w_i} \langle \Lambda_i, i \rangle$   $i \in 1 \dots n$ .

In Fig. 4 there is a DVTAG derivation chain for the sentence “abcf”, using some elementary dotted tree of Fig. 3

### 15.4 Generative power of DVTAG

In this section we show that for each DVTAG there exists a LTAG that generates the same tree language, i.e.  $\mathcal{L}(DVTAG) \subseteq \mathcal{L}(LTAG)$ ; then we show that  $\mathcal{L}(CFG) \subset \mathcal{L}(DVTAG)$ ; the consequence of these results is that DVTAG is a mildly context-sensitive formalism (Vijay-Shanker et al., 1990). In the proof of the second result we introduce a restricted LTAG formalism called dynamic LTAG (dLTAG). For the proofs that follow it is useful recall the notion of derivation tree in LTAG. Given a  $G_1(\mathcal{E})$  LTAG, a **derivation tree**  $D$  for  $G_1$  is a tree in which each node of the tree is the identifier of an elementary tree  $\gamma$  belonging to  $\mathcal{E}$ , and each arc linking a parent  $\gamma$  to a child  $\gamma'$  and labeled with Gorn address  $t$ , represents that the tree  $\gamma'$  is an auxiliary tree adjoined at node  $t$  of the tree  $\gamma$ , or that  $\gamma'$  is an initial tree substituted at the node  $t$  in  $\gamma$  (Joshi and Schabes, 1997). The constituency tree that results from the derivation described by a derivation tree  $D$  is called the derived tree

$derived(D)$ . In LTAG there are no constraints about the order in which the elementary trees are combined in the derivation tree. In order to compare DVTAG and LTAG derivations we introduce the notion of *partial derivation tree*. Given a derivation tree  $D$ , we define a **partial derivation tree**  $PD$  as a connected subgraph of  $D$ ; a partial derived tree  $derived(PD)$  is yielded from a partial derivation tree  $PD$ .

**Theorem 41** *Let<sup>3</sup>  $G_1(\mathcal{E})$  be a LTAG, let  $G_2(\langle \mathcal{E}, 0 \rangle)$  be a DVTAG and let  $D$  be a derivation tree of  $G_1(\mathcal{E})$  that derives the string  $w_1 \dots w_n$  from the elementary trees  $\gamma_1 \dots \gamma_n$ . There exists a sequence of partial derivation trees  $PD_1, PD_2, \dots, PD_n$  of  $D$  such that each  $PD_i$  is composed by the trees  $\gamma_1 \dots \gamma_i$  if and only if there exists a derivation chain  $d$  of  $G_2(\langle \mathcal{E}, 0 \rangle)$  such that  $d = \langle \Lambda_0, 0 \rangle \langle \Lambda_1, 1 \rangle \dots \langle \Lambda_n, n \rangle$  with  $\Lambda_i = derived(PD_i) \forall i \in 1 \dots n$ .*

**Proof** (*sketch*) From the definitions of partial derivation tree,  $PD_{i+1}$  is equal to  $PD_i$  but the node  $\gamma_{i+1}$ . Since LTAG operations preserve precedence and dominance relations among two nodes of the partial derived tree<sup>4</sup>, we have that  $w_1 \dots w_i$  is a prefix of  $YIELD(derived(PD_i))$ . As a consequence, in  $derived(PD_{i+1})$  there are no substitution nodes on the left of the left-anchor of  $\gamma_{i+1}$ . Since  $G_1$  and  $G_2$  use the same trees, we can show that there is a bijection between all the possible ways of inserting  $\gamma_{i+1}$  into  $PD_i$  in LTAG and the DVTAG operations that implement the transition between  $\langle \Lambda_i, i \rangle$  and  $\langle \Lambda_{i+1}, i + 1 \rangle$ . This is shown for each  $i$  through an induction process.  $\square$

A corollary of this theorem shows that a LTAG  $G_1$  generates only derivation trees  $D$  such that exists a sequence of partial derivation trees of  $PD_1, PD_2, \dots, PD_n$  such that each  $PD_i$  is composed by the trees  $\gamma_1 \dots \gamma_i$ , if and only if there is a DVTAG  $G_2$  that generates the same tree languages.

Given a partial derivation tree  $PD$  we write  $w_i < w_j$  if on the yield of  $derived(PD)$   $w_i$  precedes  $w_j$  and we write  $\gamma_i < \gamma_j$  if  $w_i < w_j$ , where  $w_i$  and  $w_j$  are the left-anchors of  $\gamma_i$  and  $\gamma_j$  respectively.

Now we define dLTAG and then we show that dLTAG respects the hypotheses of the theorem 41.

**Definition 40** A partial derivation tree  $PD$  is **dynamic** if: **(1)** Each node  $\gamma_i$  in  $PD$  has at most one child  $\gamma_j$  such that  $\gamma_j < \gamma_i$ , **(2)** If a node  $\gamma_i$  in  $PD$  has a child  $\gamma_j$  such that  $\gamma_i < \gamma_j$ , then  $\gamma_j$  does not have a child  $\gamma_k$  such that  $\gamma_k < \gamma_j$ .

<sup>3</sup>For space reasons in the proof sketch of this theorem we assume that each  $\gamma_i$  is anchored by only one lexical item  $w_i$ .

<sup>4</sup>I.e. the precedence and dominance relations among the nodes  $\mathcal{N}_1$  and  $\mathcal{N}_2$  in  $PD_i$  will be the same in  $PD_{i+j} \forall j \geq 0$ .

We define dLTAG by constraining the derivation trees to be dynamic, thus satisfying the hypotheses of theorem 41 as shown in the the following theorem.

**Theorem 42** *Let  $D$  be a derivation tree formed by the nodes  $\gamma_1 \dots \gamma_n$  of LTAG  $G_1(\mathcal{E})$  that generates the string  $w_1 \dots w_n$ . A sequence of partial derivation trees  $PD_1, PD_2, \dots, PD_n$  of  $D$  exists and  $PD_i$  is formed by  $\gamma_1 \dots \gamma_i$  ( $\forall i \in 1 \dots n$ ) if and only if  $D$  is dynamic.*

**Proof** (*sketch*) We can prove the theorem by reductio ad absurdum. Supposing that  $D$  is dynamic, we can show that if the strong connectivity of the sequence of partial derivation tree is not fulfilled by a node  $\gamma_i$  (hypotheses of theorem 42) we arrive at the absurdum that on the node  $\gamma_i$ ,  $D$  does not fulfill the dynamic definition. Similarly, supposing that  $D$  fulfills the strong connectivity hypothesis, we can show that the two conditions of the Definition 40 cannot be violated.  $\square$

A corollary of the theorems 41 and 42 states that DVTAG class of grammars generate the same tree languages of dynamic LTAG class of grammar. Given the constraints in the definition 40 we have that  $\mathcal{L}(dLTAG) \subseteq \mathcal{L}(LTAG)^5$ , since DVTAG is equivalent to dLTAG we have that  $\mathcal{L}(DVTAG) \subseteq \mathcal{L}(LTAG)$ . Now we show that  $\mathcal{L}(CFG) \subseteq \mathcal{L}(DVTAG)$  by showing that  $\mathcal{L}(CFG) \subseteq \mathcal{L}(dLTAG)$ . This fact is proved by exploiting the notion of normal form for derivation tree in LTIG (Schabes and Waters, 1995). LTIG is a particular LTAG that is strongly equivalent to CFG, with some limitations on the elementary trees and some limitations on the operations. It is possible to transform a derivation tree for a LTIG in normal form into a dynamic derivation tree with a simple algorithm (Mazzei, 2005). As a consequence, for each CFG there is dynamic LTIG that generates the same tree language, then there is a DVTAG that generates the same tree language. Now we use the results described above to prove that DVTAG is a mildly context-sensitive formalism, i.e. DVTAG generates cross-serial dependencies (1), generates the context-free languages (2), is parsable in polynomial time (3), has the constant-growth property (4). DVTAG fulfills properties (3) and (4) as a direct consequence of the inclusion in LTAG. Moreover we have sketched the proof that  $\mathcal{L}(CFG) \subseteq \mathcal{L}(DVTAG)$ , then DVTAG can generate all the context-free grammars. About (1) we need to introduce the wrapping perspective in DVTAG (next section).

---

<sup>5</sup>We can define a procedure that takes as input a dLTAG and returns a LTAG that using *selective adjoining constraints* generates the same tree language.

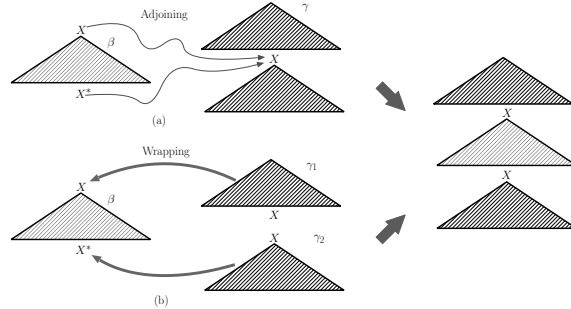


FIGURE 5: Adjoining operation (a) and wrapping operation (b).

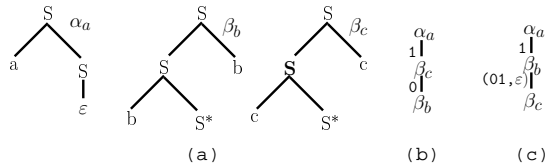


FIGURE 6: Derivation tree for the sentence “abcbc” in adjoining (b) and wrapping (c) perspective.

### 15.5 Wrapping and DVTAG generative power

Figure 5 shows the two perspectives for the adjoining operation, either inserting an auxiliary tree  $\beta$  into  $\gamma$  (Fig. 5-a) or splitting  $\gamma$  into  $\gamma_1$  and  $\gamma_2$  to include  $\beta$  (*wrapping perspective*, Fig. 5-b) (Joshi, 2004). If  $\gamma$  is an elementary tree, the wrapping operation does not change the weak and strong generative power of LTAG. However, since the wrapping operation can generate a wider set of derivation tree, on the subclass of dLTAG it does increase the strong generative power. A LTAG derivation that is not a dynamic derivation in standard LTAG can be a dynamic derivation in LTAG with wrapping operation. The LTAG elementary trees in Fig. 6-a generate the cross-serial dependencies (Joshi, 2004), but the derivation tree of Fig. 6-b for the string “abcbc” is not dynamic. As a consequence, we cannot derive the cross-serial dependencies in DVTAG. But if we rewrite this derivation tree by using wrapping, we obtain the tree of figure 6-c. In fact the adjoining of  $\beta_b$  in  $\beta_c$  becomes the wrapping of  $\beta_c$  in  $\beta_b$ . We can define a *dynamic wrapping* in DVTAG that maintains the strong equivalence between dynamic LTAG with wrapping and DVTAG with dynamic wrapping, and so we can



produce a derivation chain that derives cross-serial dependencies. Then DVTAG is mildly context-sensitive.

## 15.6 Conclusions

In this paper we have defined a dynamic version of TAG. We have sketched the basic issues of the formalism, and using these notions we have proved that DVTAG is mildly context sensitive. We have shown that DVTAG can generate cross-serial dependencies using a dynamic wrapping.

## References

- Abney, S. P. and M. Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *J. of Psycholinguistic Research* 20(3):233–250.
- Gelder, T. Van. 1998. The dynamical hypothesis in cognitive science. *Behavioral and Brain Sciences* 21:1–14.
- Hausser, R. 1992. Complexity in left-associative grammar. *Theoretical Computer Science* 106:283–308.
- Joshi, A. 2004. Starting with complex primitives pays off: Complicate locally, simplify globally. *Cognitive Science* 28(5):637–668.
- Joshi, A. and Y. Schabes. 1997. Tree-adjointing grammars. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages*, pages 69–123. Springer.
- Kamide, Y., G. T. M. Altmann, and S. L. Haywood. 2003. The time-course of prediction in incremental sentence processing: Evidence from anticipatory eye movements. *J. of Memory and Language* 49:133–156.
- Mazzei, A. 2005. *Formal and empirical issues of applying dynamics to Tree Adjoining Grammars*. Ph.D. thesis, Dipartimento di Informatica, Università degli studi di Torino.
- Milward, D. 1994. Dynamic dependency grammar. *Linguistics and Philosophy* 17(6):561–604.
- Schabes, Y. and R. Waters. 1995. Tree insertion grammar: A cubic-time, parsable formalism that lexicalizes Context-free grammar without changing the trees produced. *Computational Linguistics* 21(4):479–513.
- Stabler, E. P. 1994. The finite connectivity of linguistic structure. In C. Clifton, L. Frazier, and K. Reyner, eds., *Perspectives on Sentence Processing*, pages 303–336. Lawrence Erlbaum Associates.
- Sturt, P. and V. Lombardo. 2005. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science* 29(2):291–305.

- Vijay-Shanker, K., A. Joshi, and D. Weir. 1990. The convergence of mildly context-sensitive grammatical formalisms. In P. Sells, S. Shieber, and T. Wasow, eds., *Foundational Issues in Natural Language Processing*, pages 31–81. Cambridge MA: MIT Press.
- Woods, W. A. 1986. Transition network grammars for natural language analysis. In B. J. Grosz, K. Sparck Jones, and B. L. Webber, eds., *Natural Language Processing*, pages 71–87. Los Altos, CA: Kaufmann.

---

# Inessential Features, Ineliminable Features, and Modal Logics for Model Theoretic Syntax

HANS-JÖRG TIEDE <sup>†</sup>

## Abstract

While monadic second-order logic (MSO) has played a prominent role in model theoretic syntax, modal logics have been used in this context since its inception. When comparing propositional dynamic logic (PDL) to MSO over trees, Kracht (1997) noted that there are tree languages that can be defined in MSO that can only be defined in PDL by adding new features whose distribution is predictable. He named such features “inessential features”. We show that Kracht’s observation can be extended to other modal logics of trees in two ways. First, we demonstrate that for each stronger modal logic, there exists a tree language that can only be defined in a weaker modal logic with inessential features. Second, we show that any tree language that can be defined in a stronger modal logic, but not in some weaker modal logic, can be defined with inessential features. Additionally, we consider Kracht’s definition of inessential features more closely. It turns out that there are features whose distribution can be predicted, but who fail to be inessential in Kracht’s sense. We will look at ways to modify his definition.

**Keywords** MODEL THEORETIC SYNTAX, MODAL LOGIC, TREE AUTOMATA

## 16.1 Introduction

Model theoretic syntax is a research paradigm in mathematical linguistics that uses tools from descriptive complexity theory to formalize

---

<sup>†</sup>This research was supported by an Illinois Wesleyan University grant and a junior leave.

grammatical theories using logic. It is the aim of model theoretic syntax to find for a given grammatical theory the weakest logic that suffices to formalize it (Cornell and Rogers, 2000). At present there exist two main approaches to model theoretic syntax, those based on modal logics and those based on monadic second order logic. Since there are few, if any, linguistic examples that cannot be defined in the weakest modal logic that has been considered, it is not always clear what motivates the use of the more expressive logics. We want to argue here that there are reasons for using weaker frameworks. The present study does not aim at finding the “true” logic for model theoretic syntax. Instead, it is guided by the methodological principle that we should always use the weakest formalism that suffices to capture the phenomena under consideration, and that the use of stronger formalisms should be justified.

In Afanasiev et al. (2005), three different modal logics for the description of trees are discussed: a basic modal logic of trees,  $\mathcal{L}_{core}$ , Palm’s tense logic of trees,  $\mathcal{L}_{cp}$ , and Kracht’s dynamic logic of trees,  $PDL_{tree}$ . While the relationship between these logics and to others used in model theoretic syntax is well-understood, in that each stronger logic includes the weaker ones properly, and that all are properly included in Roger’s Monadic Second Order Logic of Trees (MSO) (Rogers, 1998), the relationship of these logics to tree languages is not as well understood. We will make use of Kracht’s (Kracht, 1997) concept of an inessential feature to get a better understanding of the tree languages that are definable or undefinable in these logics.

Kracht introduced the concept of an inessential feature to formalize the concept of a feature whose distribution is predictable from the other features. Two well-known linguistic examples of inessential features are the Slash feature of GPSG and Bar feature of X-bar theory. In addition to inessential features, Kracht also considered whether a feature is eliminable in some logic, which he identified with being globally explicitly definable. We will consider a slightly weaker notion of eliminability, but since we are mainly interested in *ineliminability*, this notion implies Kracht’s by contraposition. It was shown by Kracht that there exists a set of feature trees that is definable in  $PDL_{tree}$  and an inessential feature that is ineliminable in  $PDL_{tree}$ , but eliminable in MSO. The main purpose of this observation was to show that  $PDL_{tree}$  is strictly weaker than MSO over trees.

We show that Kracht’s theorem can be generalized to all three modal logics of trees. The proof of this result involves Thatcher’s theorem, which states that every regular tree language is the projection of a local tree language, and relates it to Kracht’s inessential features. When applied to deterministic bottom-up finite tree automata, Thatcher’s

construction of a local tree language introduces inessential features.

We also consider the definition of inessential features more closely. We want to argue that Kracht's formalization is too strong, as there are tree languages with features whose distribution can be predicted from the other features, but which fail to be inessential in Kracht's sense. Such features can be constructed by using Thatcher's construction on non-deterministic tree automata. It can be shown that such features can be turned into inessential features, which can then be eliminated in MSO. Thus, logics that can eliminate any inessential feature may be too strong. This can be seen as support for the use of weaker logics for model theoretic syntax.

## 16.2 Features and Ranked Alphabets

Kracht's definition of inessential features is given in the context of feature trees, in which each node is labeled with a set of boolean features. We are here considering trees to be terms over a ranked alphabet which differ from feature trees in that each node is labeled with a single symbol which has a fixed arity. We will use the following representation of boolean features as ranked symbols to translate feature trees into terms.

**Definition 41** Given a finite set of boolean features  $F = \{f_1, \dots, f_n\}$ , the (binary) ranked alphabet based on  $F$ ,  $\Sigma^F$ , is defined as

$$\Sigma^F = \{f_1, \neg f_1\} \times \dots \times \{f_n, \neg f_n\} \times \{0, 2\}$$

where each  $f_i, \neg f_i$  represents whether or not a feature holds at a given node and 0 or 2 represent the arity of the symbol. Thus,  $(f_1, \neg f_2, 0)$  would be a leaf symbol, and  $(f_1, \neg f_2, 2)$  would be an internal node symbol.  $\square$

The previous definition can be easily generalized to trees of any arity.

**Definition 42** A *tree* is a term over a finite ranked alphabet  $\Sigma$ . The set of  $n$ -ary function symbols in  $\Sigma$  will be denoted by  $\Sigma_n$ . The set of all trees over  $\Sigma$  is denoted by  $T_\Sigma$ ; a subset of  $T_\Sigma$  is called a *tree language*. The *yield* of a tree  $t$ , denoted by  $yield(t)$ , is defined by

$$\begin{aligned} yield(c) &= c \\ yield(f(t_1, \dots, t_n)) &= yield(t_1) \dots yield(t_n) \end{aligned}$$

with  $c \in \Sigma_0$  and  $f \in \Sigma_n, n > 0$ .  $\square$

We next define projections which we will use to study eliminability of features in the context of terms.

**Definition 43** Given a finite set of feature  $F = \{f_1, \dots, f_n\}$  and a feature  $f_i \in F$ , we define the *projection*,  $\pi$ , that eliminates  $f_i$  in the natural way:

$$\pi : \Sigma^F \rightarrow \Sigma^{F-\{f_i\}}$$

This definition can be extended to arbitrary subsets  $G \subseteq F$ , where

$$\pi : \Sigma^F \rightarrow \Sigma^{F-G}$$

Given a projection  $\pi$ , we extend  $\pi$  to a *tree homomorphism*  $\hat{\pi}$  as follows:

$$\begin{aligned} \hat{\pi}(c) &= \pi(c) \\ \hat{\pi}(f(t_1, \dots, t_n)) &= \pi(f)(\hat{\pi}(t_1), \dots, \hat{\pi}(t_n)) \end{aligned}$$

with  $c \in \Sigma_0$  and  $f \in \Sigma_n, n > 0$ . For a tree language  $L$ , we define  $\hat{\pi}(L) = \{\hat{\pi}(t) \mid t \in L\}$ .  $\square$

### 16.3 Regular Tree Languages, Local Tree Languages, and Thatcher's Theorem

The regular tree languages play a central role in model theoretic syntax because they correspond to the MSO-definable languages. There are different, equivalent ways of defining the regular tree languages. We will use bottom-up (frontier-to-root) finite tree automata, because they can be determinized.

**Definition 44** A (bottom-up, non-deterministic) *finite tree automaton* (FTA)  $M$  is a structure  $(\Sigma, Q, F, \Delta)$  where  $\Sigma$  is a ranked alphabet,  $Q$  is a finite set of states,  $F \subseteq Q$  is the set of final states, and  $\Delta$  is a finite set of transition rules of the form  $f(q_1, \dots, q_n) \rightarrow q$  with  $f \in \Sigma_n$  and  $q, q_1, \dots, q_n \in Q$ . An FTA is *deterministic* if there are no two transition rules with the same left-hand-side.  $\square$

**Definition 45** A *context*  $s$  is a term over  $\Sigma \cup \{x\}$  containing the zero-ary term  $x$  exactly once. We write  $s[x \mapsto t]$  for the term that results from substituting  $x$  in  $s$  with  $t$ .  $\square$

**Definition 46** Given a finite tree automaton  $M = (\Sigma, Q, F, \Delta)$  the derivation relation  $\Rightarrow_M \subseteq T_{Q \cup \Sigma} \times T_{Q \cup \Sigma}$  is defined by  $t \Rightarrow_M t'$  if for some context  $s \in T_{\Sigma \cup Q \cup \{x\}}$  there is a rule  $f(q_1, \dots, q_n) \rightarrow q$  in  $\Delta$ , and

$$\begin{aligned} t &= s[x \mapsto f(q_1, \dots, q_n)] \\ t' &= s[x \mapsto q] \end{aligned}$$

We use  $\Rightarrow_M^*$  to denote the reflexive, transitive closure of  $\Rightarrow_M$ . A finite automaton  $M$  *accepts* a term  $t \in T_\Sigma$  if  $t \Rightarrow_M^* q$  for some  $q \in F$ . The *tree language accepted* by a finite tree automaton  $M$ ,  $L(M)$ , is

$$L(M) = \{t \in T_\Sigma \mid t \Rightarrow_M^* q, \text{ for some } q \in F\}.$$

A tree language,  $L$ , is *regular* if  $L = L(M)$  for some FTA  $M$ .  $\square$

We will now consider the relationship between regular tree languages and context-free string languages. We assume that the reader is familiar with context-free grammars (CFGs) and their languages (CFLs).

**Theorem 43** (Thatcher, 1967) If  $L \subseteq T_\Sigma$  is regular, then

$$\{\text{yield}(t) \mid t \in L\}$$

is a CFL.  $\square$

While the yields of regular tree languages are CFLs, regular tree languages are more complex than the derivation trees of CFG. In order to compare the regular tree languages to the derivation trees of CFGs, we formalize the latter using the local tree languages.

**Definition 47** The *fork* of a tree  $t$ ,  $\text{fork}(t)$ , is defined by

$$\begin{aligned} \text{fork}(c) &= \emptyset \\ \text{fork}(f(t_1, \dots, t_n)) &= \{(f, \text{root}(t_1), \dots, \text{root}(t_n))\} \cup \bigcup_{i=1}^n \text{fork}(t_i) \end{aligned}$$

with  $c \in \Sigma_0$ ,  $f \in \Sigma_n$ ,  $n > 0$ , and  $\text{root}$  being the function that returns the symbol at the root of its argument. For a tree language  $L$ , we define

$$\text{fork}(L) = \bigcup_{t \in L} \text{fork}(t)$$

$\square$

The intuition behind the definition of *fork* is that an element of  $\text{fork}(T_\Sigma)$  corresponds to a rewrite rule of a CFG. Note that  $\text{fork}(T_\Sigma)$  is always finite, since  $\Sigma$  is finite.

**Definition 48** A tree language  $L \subseteq T_\Sigma$  is *local* if there are sets  $R \subseteq \Sigma$  and  $E \subseteq \text{fork}(T_\Sigma)$ , such that, for all  $t \in T_\Sigma$ ,  $t \in L$  iff  $\text{root}(t) \in R$  and  $\text{fork}(t) \subseteq E$ .  $\square$

We quote without proof the following two theorems by Thatcher (1967).

**Theorem 44** (Thatcher, 1967) A tree language is a set of derivation trees of some CFG iff it is local.  $\square$

**Theorem 45** (Thatcher, 1967) Every local tree language is regular.  $\square$

While there are regular tree languages that are not local, the following theorem, also due to Thatcher (1967), demonstrates that we can obtain the regular tree languages from the local tree languages via projections.

We will review the main points of the proof, because we will use some of its details later on.

**Theorem 46** (Thatcher, 1967) For every regular tree language  $L$ , there is a local tree language  $L'$  and a projection  $\pi$ , such that  $L = \hat{\pi}(L')$ .

**Proof** Let  $L$  be a regular tree language accepted by  $M = (\Sigma, Q, F, \Delta)$ . We define  $L'$  terms of  $R$  and  $E$  as follows:  $R = \Sigma \times F$  and

$$E = \{((f, q), (f_1, q_1), \dots, (f_n, q_n)) \mid f(q_1, \dots, q_n) \rightarrow q \in \Delta, \\ f_1, \dots, f_n \in \Sigma\}$$

We then define  $L' = \{t \in T_{\Sigma \times Q} \mid \text{root}(t) \in R, \text{fork}(t) \subseteq E\}$ . Notice that the trees in  $L'$  encode runs of  $M$ . That the tree homomorphism  $\hat{\pi}$  based on the projection  $\pi : \Sigma \times Q \rightarrow \Sigma$  maps  $L'$  to  $L$  can be easily verified.

It should be noted that, if  $M$  is deterministic, there exists exactly one accepting run for each tree in  $L(M)$  and thus the homomorphism  $\hat{\pi} : L' \rightarrow L$  is one-to-one.  $\square$

## 16.4 Modal Logics for Model Theoretic Syntax

Model theoretic syntax is concerned with the definability of grammatical theories in certain logics. While MSO has been a particularly successful logic for this purpose, modal logics have been used for model theoretic syntax from its inception. We now define three modal logics that were considered by Afanasiev et al. (2005).

**Definition 49** The syntax of formulas for all three modal logics is defined as follows:

$$\varphi := p_i \mid \neg\varphi \mid \varphi \wedge \psi \mid [\pi]\varphi$$

The syntax of programs is defined for each of the three logics:

$$\pi := \rightarrow \mid \leftarrow \mid \uparrow \mid \downarrow \mid \pi^* \quad (\mathcal{L}_{core})$$

$$\pi := \rightarrow \mid \leftarrow \mid \uparrow \mid \downarrow \mid \pi; \varphi? \mid \pi^* \quad (\mathcal{L}_{cp})$$

$$\pi := \rightarrow \mid \leftarrow \mid \uparrow \mid \downarrow \mid \varphi? \mid \pi; \sigma \mid \pi \cup \sigma \mid \pi^* \quad (\text{PDL}_{tree})$$

Given a logic  $\mathcal{L}$ , we will denote the set of formulas of  $\mathcal{L}$  over a finite set of atomic formulas  $F$  by  $\mathcal{L}^F$ .  $\square$

The following definition is adapted from Afanasiev et al. (2005). We consider only binary trees here.

**Definition 50** Let  $\{0, 1\}^*$  denote the set of finite sequences over  $\{0, 1\}$ . A *(binary) tree structure* is a tuple  $(T, R_{\downarrow}, R_{\rightarrow})$  where  $T$  is a binary tree domain, i.e.  $T \subseteq \{0, 1\}^*$ , such that if  $uv \in T$ , then  $u \in T$ , and if  $u1 \in T$ , then  $u0 \in T$ ,  $R_{\downarrow}$  is the daughter-of relation, i.e.  $(n, m) \in R_{\downarrow}$



iff  $m = n0$  or  $m = n1$ ,  $R_{\rightarrow}$  is the left-sister-of relation, i.e.  $(m, n) \in R_{\rightarrow}$  iff  $m = s0$  and  $n = s1$  for some  $s$ . A model is a pair  $\mathcal{M} = (\mathcal{T}, V)$ , such that  $\mathcal{T}$  is a tree structure and  $V : F \rightarrow \wp(T)$  is a valuation. We define  $\mathcal{M}, v \models \varphi$  in the usual way, the only interesting case being:

$$\mathcal{M}, v \models [\pi]\varphi \text{ iff for all } u, \text{ such that } (v, u) \in R_{\pi}, \mathcal{M}, u \models \varphi$$

and

$$\begin{aligned} R_{\uparrow} &= R_{\downarrow}^{-1} & R_{\pi \cup \sigma} &= R_{\pi} \cup R_{\sigma} \\ R_{\rightarrow} &= R_{\leftarrow}^{-1} & R_{\pi; \sigma} &= R_{\pi} \circ R_{\sigma} \\ R_{\pi^*} &= R_{\pi}^* & R_{\varphi?} &= \{(v, v) \mid \mathcal{M}, v \models \varphi\} \end{aligned}$$

where  $R^*$  denotes the transitive closure of  $R$  and  $\circ$  denotes relation composition.  $\square$

We can associate terms with tree models by identifying the atomic formulas with features.

**Definition 51** Let  $F$  be a finite set of features and  $\mathcal{L}$  be a logic. We say that  $L \subseteq T_{\Sigma^F}$  is *definable* in  $\mathcal{L}$  if there is a formula  $\varphi$  in  $\mathcal{L}^F$  such that

$$L = \{t \mid t, \varepsilon \models \varphi\}$$

where  $\varepsilon$  is the root of the tree. We write  $\mathcal{L}_1 \leq \mathcal{L}_2$  if any tree language definable in  $\mathcal{L}_1$  is definable in  $\mathcal{L}_2$ .  $\square$

The following two proposition relate tree languages to definability. The first is due to Blackburn and Meyer-Viol (1994) who proved it for a related logic.

**Proposition 47** (Blackburn and Meyer-Viol, 1994) Every local tree language is definable in  $\mathcal{L}_{core}$ .  $\square$

**Proposition 48** (Thatcher and Wright, 1968) A tree language is regular iff it is MSO-definable.  $\square$

The following, well-known, inclusions follow primarily from the definition of the three modal logics. Next, we will consider strictness of these inclusions.

**Theorem 49**  $\mathcal{L}_{core} \leq \mathcal{L}_{cp} \leq \text{PDL}_{tree} \leq \text{MSO}$

**Proof** The first two inclusions follow from Definition 49. The third inclusion follows from the fact that transitive closure is MSO-definable.  $\square$

**Proposition 50** (Schlingloff, 1992) Let  $F = \{a, b\}$ . The tree language  $L_1 \subseteq T_{\Sigma^F}$  such that each tree in  $L_1$  contains a path from the root to

a leaf at which exactly one  $a$  holds is not  $\mathcal{L}_{core}$ -definable, but is  $\mathcal{L}_{cp}$ -definable.  $\square$

**Proposition 51** Let  $\Sigma = \{\wedge, \vee, 0, 1\}$ . The tree language  $L_2 \subseteq T_\Sigma$  such that each tree in  $L_2$  evaluates to true is not  $\mathcal{L}_{cp}$ -definable, but is  $\text{PDL}_{tree}$ -definable.

**Proof** Potthoff (1994) showed that  $L_2$  is not definable in an extension of first-order logic with modular counting quantifiers, and since  $\mathcal{L}_{cp}$  is equivalent to first-order logic on trees (Afanasiev et al., 2005), the undefinability follows. That  $L_2$  is definable in  $\text{PDL}_{tree}$  is shown in Afanasiev et al. (2005).  $\square$

**Proposition 52** (Kracht, 1999, 2001) Let  $F = \{p, q\}$ . Let  $L_3 \subseteq T_{\Sigma^F}$  where each tree in  $L$  is a ternary branching tree such that  $p$  is true along a binary branching subtree and  $q$  is true at all leaves at which  $p$  is true. The language  $L_4 \subseteq T_{\Sigma\{q\}}$  obtained from the projection that eliminates  $p$  is not  $\text{PDL}_{tree}$ -definable, but is MSO-definable.  $\square$

Next, we will consider how languages that are undefinable in one of these logics can be defined with additional features.

### 16.5 Inessential and Ineliminable Features

The following definition of inessential features is adapted from Kracht (1997). Its purpose is to formalize the concept of a feature whose distribution in a language can be predicted from the other features.

**Definition 52** Let  $F$  be a finite set of features,  $G \subseteq F$ ,  $L \subseteq T_{\Sigma^F}$ , and  $\pi : \Sigma^F \rightarrow \Sigma^{F-G}$  be a projection. We call the features in  $G$  *inessential for  $L$*  if the homomorphism  $\hat{\pi} : L \rightarrow T_{\Sigma^{F-G}}$  based on  $\pi$  is one-to-one.  $\square$

The intuition for this definition of inessential features is that no two trees in  $L$  can be distinguished using features in  $G$ . Thus, given a tree  $t$  in  $\hat{\pi}(L)$ , we can recover the features from  $G$  in  $t$  using  $\hat{\pi}^{-1}$ , since  $\hat{\pi}$  is one-to-one. While being an inessential feature is defined with respect to a language, being eliminable is defined with respect to a logic and a language.

**Definition 53** Let  $F$  be a finite set of features,  $G \subseteq F$ ,  $L \subseteq T_{\Sigma^F}$ ,  $\pi : \Sigma^F \rightarrow \Sigma^{F-G}$  be a projection, and  $\mathcal{L}$  be a logic. Suppose that  $L$  is definable in  $\mathcal{L}^F$ . We say that  $G$  is *eliminable in  $\mathcal{L}$  for  $L$*  if  $\hat{\pi}(L)$  is definable in  $\mathcal{L}^{F-G}$ .  $\square$

It should be noted that this definition of eliminability does not coincide with Kracht's (Kracht, 1997), who defines eliminable as being globally explicitly definable. Kracht's definition implies the definition used here,

and thus is stronger. However, since we are interested in *ineliminability*, by contraposition, the definition employed here implies Kracht's definition of ineliminability. Kracht's proof of Proposition 52 depends on the following proposition.

**Proposition 53** (Kracht, 2001) The feature  $p$  in Proposition 52 is inessential for  $L_3$ , but ineliminable in  $\text{PDL}_{tree}$ .  $\square$

We now show how to generalize Kracht's theorem to  $\mathcal{L}_{core}$  and  $\mathcal{L}_{cp}$ :

**Theorem 54** There exists a set of features  $F$ , a tree language  $L \subseteq T_{\Sigma^F}$ , and a subset  $G \subseteq F$ , such that  $G$  is ineliminable in  $\mathcal{L}_{core}$  (resp.  $\mathcal{L}_{cp}$ ) but eliminable in  $\mathcal{L}_{cp}$  (resp.  $\text{PDL}_{tree}$ ).

**Proof** Both of these construction work the same way. Given two of our logics  $\mathcal{L}_1, \mathcal{L}_2$ , with  $\mathcal{L}_1 \leq \mathcal{L}_2$ , pick a tree language,  $L$ , that is not definable in  $\mathcal{L}_1$  but is definable in  $\mathcal{L}_2$ , which exists by Propositions 50 and 51.

By Theorem 49, we know that  $L$  is regular, and by Theorem 47, we know that any local tree language is definable in  $\mathcal{L}_1$ . Given a deterministic FTA  $M = (\Sigma, Q, F, \Delta)$ , with  $L = L(M)$ , we can use theorem 46 to construct a local tree language  $L' \subseteq T_{\Sigma \times Q}$  such that  $\hat{\pi}(L') = L$ . Now, the features in  $Q$  are inessential, since  $M$  is deterministic, but ineliminable, since  $L$  is undefinable in  $\mathcal{L}_1$ . However, since  $L$  is definable in  $\mathcal{L}_2$ , the features in  $Q$  are eliminable in  $\mathcal{L}_2$ .  $\square$

The previous theorem can be strengthened in that it can be used to *characterize* the tree languages that are undefinable in some logic  $\mathcal{L}_1$  but definable in some other logic  $\mathcal{L}_2$ , with  $\mathcal{L}_1 \leq \mathcal{L}_2$ .

**Theorem 55** Any tree language that is not definable in  $\mathcal{L}_{core}$  (resp.  $\mathcal{L}_{cp}$ ) but is definable in  $\mathcal{L}_{cp}$  (resp.  $\text{PDL}_{tree}$ ) can be defined with additional, inessential features in  $\mathcal{L}_{core}$  (resp.  $\mathcal{L}_{cp}$ ) that are not eliminable in  $\mathcal{L}_{core}$  (resp.  $\mathcal{L}_{cp}$ ).  $\square$

While it was pointed out by Volger (1999) that these and other logics that are used in model theoretic syntax are equivalent modulo a projection, the main contribution of these two theorems is that they connect Volger's observation to Kracht's inessential features. It thus demonstrates the central role that inessential features play in the comparison of logics for model theoretic syntax.

## 16.6 Inessential Features and Non-Deterministic Tree Automata

We now want to consider the definition of inessential features more closely. As was pointed out by Kracht (1997), the purpose of Definition

52 was to formalize the concept of a feature whose distribution is fixed by the other features. We now want to assess whether this formalization captures this concept correctly. For this assessment, the relationship between inessential features and Thatcher's theorem will again play a central role; but this time, we will consider the construction in Theorem 46 using non-deterministic FTAs.

Recall that the observation that Thatcher's theorem yields a language with inessential features depended on the use of a deterministic FTA, since each tree accepted by a deterministic FTA has exactly one accepting run. When we apply Thatcher's construction to non-deterministic tree automata, there can be two different accepting runs for a given tree, and so the added features fail to be inessential in Kracht's sense. However, it is clear that the distribution of the states that are used as extra features can be predicted from the other features, in the sense that we can label a tree that is accepted by a non-deterministic FTA with the states from an accepting run. It's just that there are potentially multiple such accepting runs.

Since bottom-up tree automata can be determinized, these features can be turned into inessential features using the power set construction, and since any inessential feature can be eliminated in MSO (Kracht, 1997), we can now eliminate an essential feature. This observation sheds light on the question whether the fact that certain logics cannot eliminate some inessential features is a strength or a weakness of that logic, i.e. whether or not we want logics for model theoretic syntax to be able to eliminate all inessential features. If we can turn essential features into inessential features and then eliminate them, a logic in which all inessential features can be eliminated may be too strong. This can be seen as support for the use of weaker logics for model theoretic syntax.

It should be noted that lifting the restriction that the homomorphism  $\hat{\pi}$  based on a projection  $\pi$  be one-to-one in order to extend Kracht's definition can easily make it vacuous, since any feature can be removed with a projection that is not one-to-one. What is needed is a mechanism that captures the essence of the example above. One approach might be to identify an inessential feature with a feature that is *determinizable* in the sense that a feature can be turned into an inessential feature using the power set construction. That this approach is not vacuous can be verified by applying Thatcher's construction to top-down (root-to-frontier) FTAs, which cannot be determinized. We leave the question how this definition of an inessential feature might relate to definability and eliminability as an open problem.

## 16.7 Conclusion

After significant progress in formalizing grammatical theories, one of the more pressing foundational questions in model theoretic syntax right now is how to assess in which logic to carry out this formalization. Since the logics considered here differ only with respect to which inessential features are eliminable, the central question for this assessment is whether the ineliminability of such features is a strength or a weakness of a given logic. It is argued here that, in some cases, ineliminability can be a strength. It would be interesting to consider inessential features from linguistic applications and assess their eliminability in the logics considered here.

## References

- Afanasiev, L., P. Blackburn, I. Dimitriou, B. Gaiffe, E. Goris, M. Marx, and M. de Rijke. 2005. PDL for ordered trees. *Journal of Applied Non-Classical Logic* 15(2):115–135.
- Blackburn, P. and W. Meyer-Viol. 1994. Linguistics, logic and finite trees. *Logic Journal of the IGPL* 2(1):3–29.
- Cornell, Thomas and James Rogers. 2000. Model theoretic syntax. In L. L.-S. Cheng and R. Sybesma, eds., *The GLOT International State-of-the Article Book*. Berlin: de Gruyter.
- Kracht, Marcus. 1997. Inessential features. In A. Lecomte, F. Lamarche, and G. Perrier, eds., *Logical aspects of computational linguistics*. Berlin: Springer.
- Kracht, Marcus. 1999. *Tools and techniques in modal logic*. Amsterdam: North-Holland.
- Kracht, Marcus. 2001. Logic and syntax—a personal perspective. In M. Zhakharyashev, K. Segerberg, M. de Rijke, and H. Wansing, eds., *Advances in modal logic, Vol. 2*. Stanford, CA: CSLI Publications.
- Potthoff, Andreas. 1994. Modulo-counting quantifiers over finite trees. *Theoretical Computer Science* 126(1):97–112.
- Rogers, James. 1998. *A descriptive approach to language-theoretic complexity*. Stanford, CA: CSLI Publications.
- Schlingloff, Bernd-Holger. 1992. On the expressive power of modal logics on trees. In A. Nerode and M. A. Taitslin, eds., *Logical Foundations of Computer Science - Tver '92, Second International Symposium, Tver, Russia, July 20-24, 1992, Proceedings*. Berlin: Springer-Verlag.

Thatcher, J. W. 1967. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences* 1:317–322.

Thatcher, J. W. and J. B. Wright. 1968. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory* 2:57–81.

Volger, Hugo. 1999. Principle languages and principle based parsing. In H.-P. Kolb and U. Mönnich, eds., *The Mathematics of Syntactic Structure*. Berlin: de Gruyter.

# Well-Nested Drawings as Models of Syntactic Structure

MANUEL BODIRSKY, MARCO KUHLMANN AND  
MATHIAS MÖHL

## Abstract

This paper investigates *drawings* (totally ordered forests) as models of syntactic structure. It offers a new model-based perspective on lexicalised Tree Adjoining Grammar by characterising a class of drawings structurally equivalent to TAG derivations. The drawings in this class are distinguished by a restricted form of non-projectivity (*gap degree at most one*) and the absence of interleaving substructures (*well-nestedness*).

**Keywords** MODEL-THEORETIC SYNTAX, TREE ADJOINING GRAMMAR

## 17.1 Introduction

There are two major approaches to formal accounts of the syntax of natural language, the proof-theoretic and the model-theoretic approach. Both aim at providing frameworks for answering the question whether a given natural language expression is grammatical. Their methodology, however, is rather different: In a proof-theoretic framework, one tries to set up a system of *derivation rules* (such as the rules in a context-free grammar) so that each well-formed natural language expression stands in correspondence with a derivation in that system. In contrast, in a model-theoretic framework, one attempts to specify a class of *models* for natural language expressions and a set of *constraints* on these models such that an expression is well-formed iff it has a model satisfying all the constraints. The main contribution of this

paper is the characterisation of a class of structures that provides a new model-based perspective on Tree Adjoining Grammar (TAG; Joshi and Schabes (1997)), a well-known proof-theoretic syntactic framework.

Every syntactic framework needs to account for at least two dimensions of syntactic structure: derivation structure and word order. The derivation structure captures linguistic notions such as dependency and constituency—the idea that a natural language expression can be composed of smaller expressions. Statements about word order are needed to account for the fact that not all permutations of the words of a grammatical sentence are necessarily grammatical themselves.

One of the scales along which syntactic frameworks can vary is the flexibility they permit in the relationship between derivation structure and word order. Context-free grammars do not allow any flexibility at all; derivation structure determines word order completely. In mildly context-sensitive grammar formalisms like TAG or Combinatory Categorical Grammar (Steedman, 2001), certain forms of discontinuous derivations are permitted (“crossed-serial dependencies”). Other frameworks, such as non-projective dependency grammar (Plátek et al., 2001), allow for even more flexibility to account for languages with free word order.

In this paper we introduce *drawings*, a simple class of structures for which the relaxation of the relationship between derivation structure and word order can be easily measured (§ 17.2). There is a natural way in which TAG derivations can be understood as drawings (§ 17.3). We show that the class of drawings induced in this way can be identified by two structural properties: a restriction on the degree of word order flexibility and a global property called *well-nestedness*, which disallows interleaving subderivations. In combination, these two properties capture the “structural essence” of TAG (§ 17.4). The paper concludes with a discussion of the relevance of our results and an outlook on future research (§ 17.5). For further details and full formal proofs, we refer to the extended version of this paper (Bodirsky et al., 2005).

## 17.2 Drawings

We start by introducing some basic terminology.

### 17.2.1 Relational structures

A *relational structure* is a tuple whose first component is a non-empty, finite set  $V$  of *nodes*, and whose remaining components are (in this paper) binary relations on  $V$ . The notation  $Ru$  stands for the set of all nodes  $v$  such that  $(u, v) \in R$ . We use the standard notations for the transitive ( $R^+$ ) and reflexive transitive ( $R^*$ ) closure of binary relations.

In this paper, we are concerned with two types of relational struc-



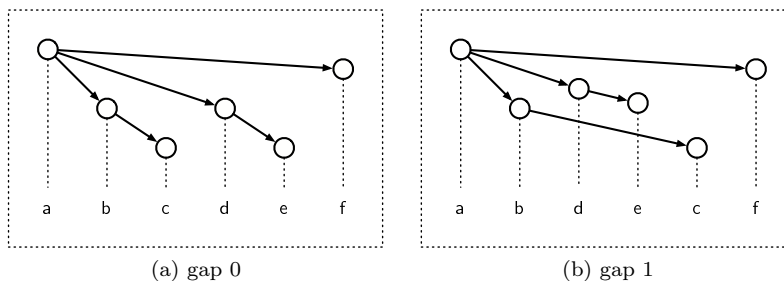


FIGURE 1: Two drawings

tures in particular: *forests* and *total orders*. A relational structure  $(V; \triangleleft)$  is called a *forest* iff  $\triangleleft$  is acyclic and every node in  $V$  has at most one predecessor with respect to  $\triangleleft$ . Nodes in a forest with no  $\triangleleft$ -predecessors are called *roots*. A *tree* is a forest that has exactly one root. A *total order* is a relational structure  $(V; \prec)$  in which  $\prec$  is transitive and for all  $v_1, v_2 \in V$ , exactly one of the following three conditions holds:  $v_1 \prec v_2$ ,  $v_1 = v_2$ , or  $v_2 \prec v_1$ . Given a total order, the *interval* between two nodes  $v_1$  and  $v_2$  is the set of all  $v$  such that  $v_1 \preceq v \preceq v_2$ . The *cover* (also known as *convex hull*) of a set  $V' \subseteq V$ ,  $\mathcal{C}(V')$ , is the smallest interval containing  $V'$ . A set  $V'$  is *convex* iff it is equal to its cover. A *gap* in a set  $V'$  is a maximal, non-empty interval in  $\mathcal{C}(V') - V'$ . We call the number of gaps in a set the *gap degree* of that set and write  $G_k(V)$  for the  $k$ -th gap in  $V$  (counted, say, from left to right).

### 17.2.2 Drawings and gaps

Drawings are relational structures with two binary relations: a forest to model derivation structure, and a total order to model word order.

**Definition 54** A *drawing* is a relational structure  $(V; \triangleleft, \prec)$  where  $(V; \triangleleft)$  forms a forest, and  $(V; \prec)$  forms a total order. Drawings whose underlying forest is a tree will be called *T-drawings*.

Note that, in contrast to ordered forests (where order is defined on the direct successors of each node), order in drawings is *total*. By identifying each node  $v$  in a drawing with the set  $(\triangleleft^*)v$  of nodes in the subtree rooted at  $v$ , we can lift the notions of cover and gap as follows:  $\mathcal{C}(v) := \mathcal{C}((\triangleleft^*)v)$ ,  $G_k(v) := G_k((\triangleleft^*)v)$ . The gap degree of a drawing is the maximum among the gap degrees of its nodes.

Fig. 1 shows two drawings of the same underlying tree. The circles and solid arcs reflect the forest structure. The dotted lines mark the positions of the nodes with respect to the total order. The labels attached to the dotted lines give names to the nodes. Drawing 1a has gap degree

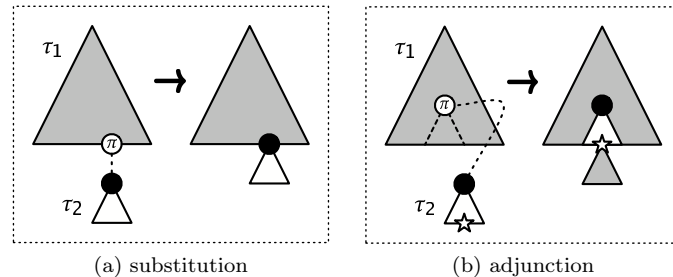


FIGURE 2: Combining tree structures in TAG

zero, since  $\mathcal{C}(v) = (\triangleleft^*)v$  for all nodes  $v$ . In contrast, drawing 1b has gap degree one, since the set  $\{d, e\} = \{b, d, e, c\} - \{b, c\} = \mathcal{C}(b) - (\triangleleft^*)b$  is a gap for node  $b$ , and no other node has a gap.

### 17.2.3 Related work

Our terminology can be seen as a model-based reconstruction of the terminology developed for non-projective dependency trees (Plátek et al., 2001), where gaps are defined with respect to tree structures generated by a grammar. The notion of gap degree is closely related to the notion of *fan-out* in work on (string-based) finite copying parallel rewriting systems (Rambow and Satta, 1999): fan-out measures the number of substrings that a sub-derivation *does* contribute to the complete yield of the derivation; dually, the gap degree measures the number of substrings that a sub-derivation *does not* contribute.

## 17.3 Drawings for TAG

Tree Adjoining Grammar (TAG) (Joshi and Schabes, 1997) is a proof-theoretic syntactic framework whose derivations manipulate tree structures. This section gives a brief overview of the formalism and shows how drawings model derivations in lexicalised TAGs.

### 17.3.1 Tree Adjoining Grammar

The building blocks of a TAG grammar are called *elementary trees*; they are successor-ordered trees in which each node has one of three types: *anchor* (or *terminal node*), *non-terminal node*, or *foot node*. Anchors and foot nodes must be leaves; non-terminal nodes may be either leaves or inner nodes. Each elementary tree can have at most one foot node. Elementary trees without a foot node are called *initial trees*; non-initial trees are called *auxiliary trees*. A TAG grammar is *strictly lexicalised*, if each of its elementary trees contains exactly one anchor.

Trees in TAG can be combined using two operations (Fig. 2): *Substitution* combines a tree structure  $\tau_1$  with an initial tree  $\tau_2$  by identifying a non-terminal leaf node  $\pi$  of  $\tau_1$  with the root node of  $\tau_2$  (Fig. 2a). *Adjunction* identifies an inner node  $\pi$  of a structure  $\tau_1$  with the root node of an auxiliary tree  $\tau_2$ ; the subtree of  $\tau_1$  rooted at  $\pi$  is excised from  $\tau_1$  and inserted below the foot node of  $\tau_2$  (Fig. 2b; the star marks the foot node). Combing operations are disallowed at root and foot nodes.

TAG *derivation trees* record information about how tree structures were combined during a derivation. Formally, they can be seen as unordered trees whose nodes are labelled with elementary trees, and whose edges are labelled with the nodes at which the combining operations took place. If  $v$  is a node in a derivation tree, we write  $\ell(v)$  for the label of  $v$ . An edge  $v_1 - \pi \rightarrow v_2$  signifies that the elementary tree  $\ell(v_2)$  was substituted or adjoined into the tree  $\ell(v_1)$  at node  $\pi$ .

TAG *derived trees* represent results of derivations; we write  $\text{drv}(D)$  for the derived tree corresponding to a derivation tree  $D$ . Derived trees are ordered trees made up from the accumulated material of the elementary trees participating in the derivation. In particular, each TAG derivation induces a mapping  $\rho$  that maps each node  $v$  in  $D$  to the root node of  $\ell(v)$  in  $\text{drv}(D)$ . In strictly lexicalised TAGs, a derivation also induces a mapping  $\alpha$  that maps each node  $v$  in  $D$  to the anchor of  $\ell(v)$  in  $\text{drv}(D)$ .

For derivation trees  $D$  in strictly lexicalised TAGs, we define

$$\begin{aligned} \text{derived}(v) &:= \{ \alpha(u) \mid v \triangleleft^* u \text{ in } D \} \quad \text{and} \\ \text{yield}(v) &:= \{ \pi \mid \pi \text{ is an anchor and } \rho(v) \triangleleft^* \pi \text{ in } \text{drv}(D) \}. \end{aligned}$$

The set  $\text{derived}(v)$  contains those anchors in  $\text{drv}(D)$  that are contributed by the partial derivation starting at  $\ell(v)$ ;  $\text{yield}(v)$  contains those anchors that are dominated by the root node of  $\ell(v)$ . To give a concrete example: Fig. 3 shows a TAG derivation tree (3a) and its corresponding derived tree (3b). For this derivation,  $\text{derived}(\textit{like}) = \{\textit{what}, \textit{Dan}, \textit{like}\}$  and  $\text{yield}(\textit{like}) = \text{derived}(\textit{like}) \cup \{\textit{does}\}$ .

### 17.3.2 TAG drawings

There is a natural relation between strictly lexicalised TAGs and drawings: given a TAG derivation, one obtains a drawing by ordering the nodes in the derivation tree according to the left-to-right order on their corresponding anchors in the derived tree.

**Definition 55** Let  $D$  be a derivation tree for a strictly lexicalised TAG. A drawing  $(V; \triangleleft, \prec)$  is a TAG-drawing iff (a)  $V$  is the set of nodes in  $D$ ; (b)  $v_1 \triangleleft v_2$  iff for some  $\pi$ , there is an edge  $v_1 - \pi \rightarrow v_2$  in  $D$ ; (c)  $v_1 \prec v_2$  iff  $\alpha(v_1)$  precedes  $\alpha(v_2)$  with respect to the leaf order in  $\text{drv}(D)$ .

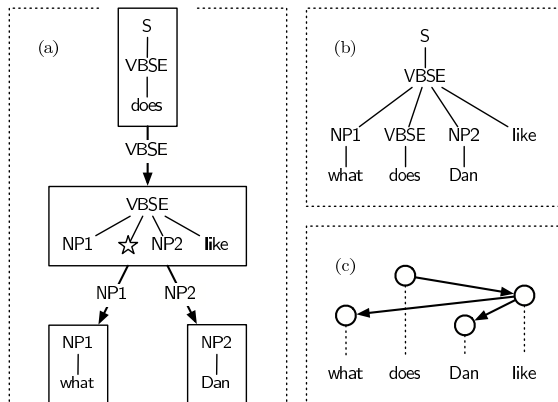


FIGURE 3: TAG derivation trees (a), derived trees (b), and drawings (c)

Fig. 3c shows the TAG drawing induced by the derivation in Figs. 3a–b.

### 17.4 The structural essence of TAG

Now that we have defined how TAG derivations induce drawings, we can ask whether *all* drawings (whose underlying forests are trees) are TAG drawings. The answer to this question is “no”: TAG drawings form a proper subclass in the class of all drawings. As the major technical result of this paper, we will characterise the class of TAG drawings by two structural properties: a restriction on the gap degree and a property we call *well-nestedness* (Definition 56). The relevance of this result is that it provides a characterisation of “TAG-ness” that does not make reference to any specific grammar, but refers to purely structural properties: well-nested drawings with gap degree at most one are “just the right” models for TAG in the sense that every TAG derivation induces such a drawing, and for any such drawing we can construct a TAG grammar that allows for a derivation inducing that drawing.

#### 17.4.1 TAG drawings are gap one

Gaps in TAG drawings correspond to adjunctions in TAG derivations: each adjunction may introduce material into the yield of a node that was not derived from that node. Since auxiliary trees have only one foot node, TAG drawings can have at most one gap.

**Lemma 56** *Let  $D$  be a TAG derivation tree, and let  $v$  be a node in  $D$ . Then (a)  $\text{derived}(v) \subseteq \text{yield}(v)$ , (b)  $\text{yield}(v) - \text{derived}(v)$  is convex, and (c)  $\text{derived}(v)$  contains at most one gap.*

**Corollary 57** *TAG drawings have gap degree at most one.*

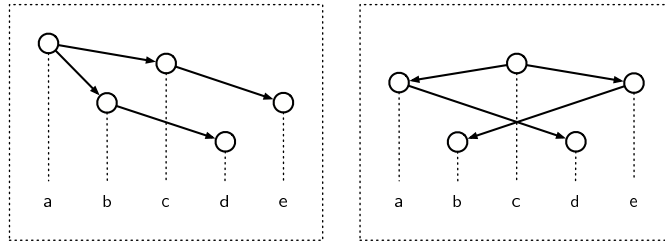


FIGURE 4: Two drawings that are not well-nested

#### 17.4.2 TAG drawings are well-nested

The gap restriction alone is not sufficient to characterise TAG drawings: there are drawings with gap degree one that cannot be induced by a TAG. Fig. 4 shows two examples. To see why these drawings cannot be induced by a TAG notice that in both of them, the cover of two nodes overlap ( $\mathcal{C}(b)$  and  $\mathcal{C}(c)$  in the left drawing,  $\mathcal{C}(a)$  and  $\mathcal{C}(e)$  in the right one). Since each node in a drawing corresponds to a sub-derivation on the TAG side, this would require the overlap of two yields in the derived tree, which is impossible. The present section will make this statement precise.

**Definition 56** Let  $T_1$  and  $T_2$  be disjoint subtrees in a drawing. We say that  $T_1$  and  $T_2$  *interleave* iff there are nodes  $l_1, r_1 \in T_1$  and  $l_2, r_2 \in T_2$  such that  $l_1 \prec l_2 \prec r_1 \prec r_2$ . A drawing is called *well-nested* iff it does not contain any interleaving subtrees.

Well-nestedness is a purely structural property: it does not make reference to any particular grammar at all. In this respect, it is similar to the condition of *planarity* (Yli-Jyrä, 2003). In fact, one obtains planarity instead of well-nestedness from Definition 56 if the disjointness condition is relaxed such that  $T_2$  may also be a subtree of  $T_1$ , and  $l_1, r_1$  are chosen from  $T_1 - T_2$ .

**Lemma 58** TAG drawings are well-nested.

#### 17.4.3 Constructing a TAG grammar for a drawing

To complete our characterisation of TAG drawings, we now present an algorithm that takes a well-nested drawing with gap degree at most one and constructs a TAG grammar whose only derivation induces the original drawing. Correctness of the algorithm establishes the following

**Lemma 59** Each well-nested  $T$ -drawing that has gap degree at most one is a TAG drawing.

The algorithm (fully discussed in the extended version of this paper) performs a pre-order traversal of the tree structure underlying the drawing. For each node  $v$ , it constructs an elementary tree whose anchor is  $v$  and whose non-terminal nodes license exactly the combining operations required by the outgoing edges of  $v$ . Children  $w$  of  $v$  that do not have a gap induce sites for substitutions (non-terminal leaf nodes), other children induce sites for adjunctions (non-terminal inner nodes). If  $v$  itself has a gap, it also needs to include a foot node. The order and dominance relation on the nodes in the constructed elementary tree are determined by the order and nesting of the *scopes* of the nodes: the scope of an anchor is the anchor itself, the scope of a non-terminal node is the cover of the corresponding child node, and the scope of a foot node is the gap that triggered the inclusion of this foot node. Well-nestedness ensures that the nesting of the scopes can be translated into a tree relation between the nodes in the elementary tree.

The combination of Lemmata 56, 58 and 59 implies

**Theorem 60** *A  $T$ -drawing is a TAG drawing iff it is well-nested and has gap degree at most one.*

## 17.5 Conclusion

This paper introduced *drawings* as models of syntactic structure and presented a novel perspective on lexicalised TAG by characterising a class of drawings structurally equivalent to TAG derivations. The drawings in this class—we called them TAG drawings—have two properties: they have a *gap degree* of at most one and are *well-nested*. TAG drawings are suitable structures for a model-theoretic treatment of TAG.

We believe that our results can provide a new perspective on the treatment of languages with free word order in TAG. Since TAG’s ability to account for word order variations is extremely limited, various attempts have been made to move TAG into a description-based direction.<sup>1</sup> Drawings allow us to analyse these proposals with respect to the question how they extend the class of *models* of TAG, and what new *descriptive means* they offer to talk about these models. We feel that these issues were not clearly separated in previous work on model-theoretic TAG (Palm, 1996, Rogers, 2003).

A model-theoretic approach to natural language processing lends itself to constraint-based processing techniques. We have started to investigate the computational complexity of constraint satisfaction problems on TAG drawings by defining a relevant constraint language and formulating a constraint solver that decides in polynomial time whether

---

<sup>1</sup>Kallmeyer’s dissertation (Kallmeyer, 1999) provides a comprehensive summary.

a formula in that language can be satisfied on a well-nested drawing (Bodirsky et al., 2005). This solver can be used as a propagator in a constraint-based processing framework for TAG descriptions.

Our immediate future work will be concerned with the further development of our processing techniques into a model-based parser for TAGs. The current constraint solver propagates information about structures that are already known; a full parser would need to construct these structures in the first place. In the longer term, we hope to characterise other proof-theoretic syntactic frameworks in terms of drawings, such as Multi-Component TAG and Combinatory Categorical Grammar.

**Acknowledgements** We are grateful to Alexander Koller, Guido Tack and an anonymous reviewer for useful comments on earlier versions of this paper. The work of Kuhlmann and Möhl is funded by the Collaborative Research Centre 378 “Resource-Adaptive Cognitive Processes” of the Deutsche Forschungsgemeinschaft (DFG).

## References

- Bodirsky, Manuel, Marco Kuhlmann, and Mathias Möhl. 2005. Well-nested drawings as models of syntactic structure. Tech. rep., Saarland University. Electronic version available at <http://www.ps.uni-sb.de/Papers/>.
- Joshi, Aravind and Yves Schabes. 1997. *Handbook of Formal Languages*, vol. 3, chap. Tree Adjoining Grammars, pages 69–123. Springer.
- Kallmeyer, Laura. 1999. *Tree Description Grammars and Underspecified Representations*. Ph.D. thesis, Universität Tübingen.
- Palm, Adi. 1996. From constraints to TAGs: A transformation from constraints into formal grammars. In *Second Conference on Formal Grammar*. Prague, Czech Republic.
- Plátek, Martin, Tomáš Holan, and Vladislav Kuboň. 2001. On relaxability of word-order by d-grammars. In C. Calude, M. Dinneen, and S. Sburlan, eds., *Combinatorics, Computability and Logic*, pages 159–174. Springer.
- Rambow, Owen and Giorgio Satta. 1999. Independent parallelism in finite copying parallel rewriting systems. *Th. Computer Science* 223:87–120.
- Rogers, James. 2003. Syntactic structures as multi-dimensional trees. *Research on Language and Computation* 1:265–305.
- Steedman, Mark. 2001. *The Syntactic Process*. MIT Press.
- Yli-Jyrä, Anssi. 2003. Multiplanarity – a model for dependency structures in treebanks. In *Second Workshop on Treebanks and Linguistic Theories*, pages 189–200. Växjö, Sweden.





---

# A Linearization-based approach to Gapping

RUI P. CHAVES

## Abstract

Non-constituent Coordination phenomena have for a long time eluded a formally precise uniform account in Linguistic Theory, including Constraint-based grammar formalisms. This work provides a novel approach to Gapping phenomena (Ross, 1970) by extending recent Linearization-based accounts of NCC in Head-driven Phrase Structure Grammar (Pollard and Sag, 1994).

**Keywords** NON-CONSTITUENT COORDINATION, DOMAINS, HPSG

## 18.1 Introduction

Kathol (1995), Crysmann (2000, 2003), and Yatabe (2001) propose to use linearization domains to capture different kinds of Non-constituent Coordination (NCC) phenomena in HPSG. Recently, Beavers and Sag (2004) propose a uniform coordination construction to account for NCC phenomena in general. Gapping however, remains unaccounted for.

In this paper we show that a linearization approach to NCC can also accommodate Gapping, as well as Stripping phenomena. Section 2 overviews the proposal in Beavers and Sag (2004) and section 3 discusses the data as well as some linguistic claims found in the literature. Section 4 provides an integrated analysis and addresses some issues for long-distance NCC in general. Section 5 contains concluding remarks.

## 18.2 Non-Constituent Coordination and HPSG

Following insights in Crysmann (2000, 2003) and Yatabe (2001), Beavers and Sag (2004) recently propose a general constraint in HPSG that

uses structure-sharing constraints operating over order domains (DOM) (Reape, 1994, Kathol, 1995) to capture Constituent Coordination (CC), Right-Node Raising (RNR), and Argument Cluster Coordination (ACC) patterns (henceforth commas mark pauses):

- (14) a. Bill loves wine and Mary hates beer. (CC)  
 b. Bill cooked, and Mary ate, a pizza. (RNR)  
 c. He gave a rose to Ann, and an orchid to Tracy. (ACC)

The relevant coordination construction is given in (15),<sup>1</sup> where daughter domain lists are split, and partially concatenated in the mother:

(15)  $cnj-cx \Rightarrow$

$$\left[ \begin{array}{l} \text{MTR} \left[ \begin{array}{l} \text{DOM } \boxed{A} \oplus \boxed{B_1} \oplus \boxed{C} \oplus \boxed{B_2} \oplus \boxed{D} \\ \text{SYN } \boxed{0} \end{array} \right] \\ \\ \text{DTRS} \left\{ \begin{array}{l} \left[ \begin{array}{l} \text{DOM } \boxed{A} \left\langle \left[ \begin{array}{l} \text{FRM } \boxed{F_1} \\ \text{HD } \boxed{H_1} \end{array} \right] \right\rangle, \dots, \left[ \begin{array}{l} \text{FRM } \boxed{F_n} \\ \text{HD } \boxed{H_n} \end{array} \right] \right\rangle \oplus \\ \boxed{B_1}_{ne-list} \oplus \left\langle \left[ \begin{array}{l} \text{FRM } \boxed{G_1} \\ \text{HD } \boxed{I_1} \end{array} \right] \right\rangle, \dots, \left[ \begin{array}{l} \text{FRM } \boxed{G_m} \\ \text{HD } \boxed{I_m} \end{array} \right] \right\rangle, \\ \text{SYN } \boxed{0} \\ \text{CRD } - \end{array} \right] \\ \\ \left[ \begin{array}{l} \text{DOM } \boxed{C} \left\langle \left( \left[ \text{SYN } cnj \right] \right) \right\rangle \oplus \left\langle \left[ \begin{array}{l} \text{FRM } \boxed{F_1} \\ \text{HD } \boxed{H_1} \end{array} \right] \right\rangle, \dots, \left[ \begin{array}{l} \text{FRM } \boxed{F_n} \\ \text{HD } \boxed{H_n} \end{array} \right] \right\rangle \oplus \\ \boxed{B_2}_{ne-list} \oplus \boxed{D} \left\langle \left[ \begin{array}{l} \text{FRM } \boxed{G_1} \\ \text{HD } \boxed{I_1} \end{array} \right] \right\rangle, \dots, \left[ \begin{array}{l} \text{FRM } \boxed{G_m} \\ \text{HD } \boxed{I_m} \end{array} \right] \right\rangle \\ \text{SYN } \boxed{0} \\ \text{CRD } + \end{array} \right] \end{array} \right\} \end{array} \right]$$

$n, m \geq 0$

This construction conjoins two constituents and splits the list of domain objects of each conjunct into three sublists. Crucially, the peripheral lists  $\boxed{A}$  and  $\boxed{D}$  may or not be empty. In the latter case, the members of these lists must share category and morphological form between conjuncts (via HEAD and FORM). The mother's domain corresponds to the concatenation of the shared peripheries  $\boxed{A}$  and  $\boxed{D}$ , and the non-shared domain objects contributed by the daughters. But note that the shared material only occurs peripherally in the mother node. This concatenation pattern of DOM lists in the mother node allows (15) to yield CC if both peripheries are empty ( $\boxed{A} = \langle \rangle, \boxed{D} = \langle \rangle$ , in (16a)), RNR if the right periphery is non-empty ( $\boxed{A} = \langle \rangle, \boxed{D} = \langle [a, pizza] \rangle$ , in (16b)), and ACC if the left periphery is non-empty ( $\boxed{A} = \langle [gave] \rangle, \boxed{D} = \langle \rangle$ , in (16c)):

<sup>1</sup>A *hd-mk-cxt* construction ensures the base case (Beavers and Sag, 2004, 59-60).

- (16) a. [DOM ⟨[Bill], [loves], [wine], [and], [Mary], [hates], [beer]⟩]  
 b. [DOM ⟨[Bill], [cooked], [and], [Mary], [ate], [a, pizza]⟩]  
 c. [DOM ⟨[gave], [a, rose], [to, Ann], [and], [an, orchid], [to, Tracy]⟩]

Because in Beavers and Sag (2004) morph forms are shared (rather than phonology or entire domain objects), one correctly rules out (17):<sup>2</sup>

- (17) a. # I ran out of luck and ~~ran~~ down a stone pillar.  
 b. # Ned said Mia, and Tom said Bob, are a nice couple.  
 c. # John bought ~~some old books~~ and Mary sold some old book.

But (15) fails to capture Gapping since sharing would have to be non-peripheral and systematically located in the non-initial conjunct:

- (18) a. John will bring dessert, and Mary, wine.  
 b. Ann reads stories to her kids, and Maria, to the students.  
 c. Tim wrote a book in London, and his brother, in Paris.

### 18.3 Gapping Data

Gapping operates independently from other NCC phenomena given that it may co-occur with RNR for instance, as seen in (19):

- (19) a. I tried to argue with Mia, and Greg, with Kate, that by noon  
 the show would probably be sold-out.  
 b. I told Maria, and David, Anna, that the lecture was canceled.

Also, Gapping is able to occur in comparatives (Hendriks, 1995):

- (20) a. Paula kissed more boys than Sue girls.  
 b. ?\*Paula kissed more than Sue kissed girls. (RNR)  
 c. ?\*Paula kissed more boys than hugged girls. (ACC)

It is also well-known that Gapping does not preserve verbal inflection:

- (21) a. John admires Neil Young, and his friends, Elvis Costello.  
 b. Sam was buying balloons, and the other kids, food and drinks.

Gapping cannot be arbitrarily embedded in NP islands for instance:

- (22) \*Alan went to London, and Bill met a girl who said Jim, to Paris.

#### 18.3.1 Discourse Anaphora

We do not address VP or N Ellipsis presently, as it is known to differ significantly from NCC, e.g. allowing for non-linguistic antecedents:

- (23) a. [Hankamer brandishes cleaver, advances on Sag]  
 Sag: Don't! My god, don't (Hankamer and Sag, 1976, 409)

<sup>2</sup>But note that an extra constraint like  $[\text{SYN } 0] \in [A] \oplus [B_1]$  may be required to prevent RNR the local head: ‘\*John and Mary smiles’ (cf. Crysmann (2002, 301)).

- b. [John opens a book, and as Mia opens a newspaper, says:]  
# (And) Mia, a newspaper.
- c. [mechanic approaches his boss after examining the race cars]  
One with a broken gear, the other two are ok.

Cases of so-called N-Gapping (Jackendoff, 1971) are very different from Verbal Gapping. N-Gapping can also occur as discourse anaphora with different agreement (cf. (23c) and (24a)), and reside in non-conjuncts as in (24b):

- (24) a. I have one car with flat tires and two with a broken gear.
- b. After his car was stolen, Tim sought to buy one with an alarm.

This suggests that ‘Nominal Gapping’ should rather be analyzed as anaphoric Nominal Ellipsis (see also Neijt (1979, 29)). Further evidence comes from the fact that, unlike in Verbal Gapping, such cases can be arbitrarily embedded, and may even occur in the initial conjunct:

- (25) a. Six candidates abandoned the interview and I met a colleague who mentioned that two even failed to show up.
- b. John’s photo of Anna is good, and I met someone who agreed that Fred’s of Lynn was not bad either.
- (26) Tim only gulped *one* early in the morning, but his sister managed to eat *three* chocolate bars before lunch.

For the present work, we thus assume that Gapping only applies to verbal constituents. Of course, many dependents can also be gapped along with the verb, as in (18) above and in (27) (Pesetsky, 1982, 645):

- (27) a. This doctor said I should eat tuna fish, and that doctor, salmon.
- b. Timmy thinks mom bought a new bike, and Annie, a puppy.

See Lappin (1999) for a HPSG account of antecedent-contained Ellipsis.

### 18.3.2 Locality Constraints

Neijt (1979, 138) argues that Wh-islands block Gapping with examples such as the one given in (28a). However, the oddness is probably pragmatic rather than syntactic, given the structurally identical sentence in (28b) (assume for instance that Bo and Mia are team leaders):

- (28) a. \*John wondered what to cook today and Peter, tomorrow.
- b. Bo decided who is working tomorrow and Mia, the next day.

Lasnik and Saito (1992) and others argue that conjuncts containing gaps cannot be larger than IP, based on examples like (29) below:

- (29) \*I think that John saw Bill, and that Mary ~~saw~~ Susan.

In HPSG terms, this is equivalent to assuming that Gapping only applies to unmarked clauses, i.e. phrases with the feature MARKING *none*. On the other hand, unmarked subordinate clauses can easily gap:

(30) I hope Ann enjoys her coloring book, and John, his new bike.

This contrasts with RNR for instance, known to cross CP boundaries:

- (31) a. Sandy doubted he could buy, and Carol knew he couldn't buy,  
the collected works of Rosa Luxemburg.  
b. I've been wondering whether, but wouldn't positively want to  
state that, your theory is correct. (Bresnan, 1974)

The difficulty in gapping marked clauses is intriguing, and suggests that a generalization is perhaps being missed. An alternative approach to this issue might be to consider the interaction of prosodic or psycholinguistic processing factors. Consider for example Sohn (1999, 368), where it is claimed that it is not possible to gap embedded constituents:

(32) ?? Jo said the boy likes cheese, and Tom ~~said the boy likes~~ pickles.

Such claim is refuted by examples like the ones given in (27) and (33):

- (33) One reviewer said that the paper had nine typos, and the other reviewer, only two.

Hankamer (1973) proposes principles like 'The No-Ambiguity Constraint' (and more generally, the 'The Structural Recoverability Hypothesis') that prevent a Gapping operation from yielding a syntactically ambiguous gapped structure. In this case embedded Gapping is possible, but dispreferred since there is a tendency to match the right remnant to the closest NP. This is the preferential reading of the example in (34), with wide scope of *think*:

(34) I think that Mia wrote me an essay, and Sue, an entire paper.

Similar processing biases and exceptions motivated Kuno (1976) to propose 'perceptual' constraints like the 'Minimal Distance Principle' or the 'Tendency for Subject-Predicate Interpretation'. In this view, processing is biased to preferential parsings which sometimes do not arrive to a valid analysis and, due to the complexity of the structures, fail to re-process them adequately.<sup>3</sup> It is unclear if there is such an explanation for (29), and for now we opt for a syntactic account.

Gapping does require that the head of the conjunct is not a remnant (e.g. Jackendoff (1971), Sag (1976, 139–147), and Chao (1988, 19–34)):

- (35) a. John knows how to make spaghetti and Sue, macaroni.  
b. John knows how to prepare sushi and Sue, how to cook it.

---

<sup>3</sup>See also Keller (2001) for an experimental study on Gapping gradience in OT.

- c. \*John knows how to eat spaghetti and Bill wonders, macaroni.
- d. \*Kim knows how to make sushi and Bill learned how to, pizza.

(36) John is fond of Mary, and Bill (\*said he), of Sue.

The same applies to interrogative and passive sentential conjuncts:

- (37) a. Which cat likes olives, and which cat, grapes?  
b. \*I asked which cat likes olives, and you asked which cat, grapes.
- (38) a. Did Bill eat the peaches, and (\*did) Harry, the grapes?  
b. Tim was hassled by the Police, and Bob (\*was), by the FBI.

This simple generalization is able to straightforwardly rule out well-known data where the deleted verb is not the head of the conjunct:

- (39) a. \*Tim ate the rice, and that Harry ~~ate~~ the beans is fantastic.  
b. \*The man who sells books and the woman who ~~sells~~ flowers met outside.

We conclude that Gapping is not as conditioned by locality constraints as previously claimed, and that the fundamental syntactic condition for Gapping is the deletion of the local verbal head. Several island effects are also known to condition Gapping, and these will be predicted by independently motivated linearization constraints.

In the next section we provide a syntactic account of Gapping, while following the standard assumption that pragmatics, processing and prosodic factors can interact to promote or penalize intelligibility (as discussed above in (28), as well as in (64) and (65) below).

#### 18.4 Gapping in HPSG

The coordination construction proposed by Beavers and Sag (2004) uses constraints that quantify over tag indices (structure-sharing between  $n$  items inside potentially empty lists:  $n \geq 0$ ). Strictly speaking, this raises non-trivial issues for feature-logic formalisms and their implementation. But usually such notation is taken to abbreviate standard relational constraints. One can thus obtain an equivalent version of (15) that captures the sharing of FORM and HEAD values across potentially empty lists in RNR and ACC via an ancillary  $\mathbf{h\_f\_share}/2$  relation:

$$(40) \mathbf{h\_f\_share}(\boxed{1}, \boxed{2}) \leftarrow (\boxed{1}=\langle \rangle \wedge \boxed{2}=\langle \rangle) \vee$$

$$(\boxed{1}=\langle \left[ \begin{array}{cc} \text{FRM} & \boxed{3} \\ \text{SYN} | \text{HD} & \boxed{4} \end{array} \right] | L_1 \rangle \wedge \boxed{2}=\langle \left[ \begin{array}{cc} \text{FRM} & \boxed{3} \\ \text{SYN} | \text{HD} & \boxed{4} \end{array} \right] | L_2 \rangle \wedge \mathbf{h\_f\_share}(L_1, L_2))$$

This relation takes two potentially empty lists of domain objects and ensures that all members have pair-wise shared FORM and HEAD values. We adopt this alternative for perspicuity, without loss of generality.

In the case of Gapping however, the identity requirements seem to be weaker. This is the case of verbal inflection identity, not required in Gapping as seen in (21), but necessary for RNR:

(41) \*I said that the kids, and you claimed that the professor, was ill.

Identity in Gapping seems to be essentially categorial and semantic in nature, and thus we will define it as sharing of HEAD and RELN values between the gap and the antecedent, as illustrated below:

(42) Tracy arrives today, and her friends, tomorrow.

$$\left[ \begin{array}{l} \text{PHON } \langle \text{arrives} \rangle \\ \text{SYN} \mid \text{CAT} \mid \text{HEAD} \quad \boxed{1} \quad \left[ \begin{array}{l} \text{verb} \\ \text{VFORM } \textit{finite} \\ \text{AUX} \quad - \\ \text{INV} \quad - \end{array} \right] \\ \text{SEM} \mid \text{RELS} \quad \langle \text{RELN } \boxed{2} \textit{arrive}_1 \textit{-rel} \rangle \end{array} \right] \quad \left[ \begin{array}{l} \text{PHON } \langle \text{arrive} \rangle \\ \text{SYN} \mid \text{CAT} \mid \text{HEAD} \quad \boxed{1} \quad \left[ \begin{array}{l} \text{verb} \\ \text{VFORM } \textit{finite} \\ \text{AUX} \quad - \\ \text{INV} \quad - \end{array} \right] \\ \text{SEM} \mid \text{RELS} \quad \langle \text{RELN } \boxed{2} \textit{arrive}_1 \textit{-rel} \rangle \end{array} \right]$$

Note that only the name of the predicate relation is shared. Consequently, predicates with different semantic relations cannot be identified, including singular and plural nouns e.g. ‘ticket(*x*)’ and ‘ticket\*(*X*)’:

(43) #I bought the tickets today, and Bo ~~bought the ticket~~ yesterday.

Since agreement information is located in SEM|INDEX (see Pollard and Sag (1994, 76)), sharing HEAD of gapped items does not force identical agreement, but it does force identical verbal inflectional form (HEAD|VFORM). The data from Portuguese given below show that the verb tense form (besides semantics) is the relevant identity requirement:

(44) Eu chego            hoje e eles ~~chegam~~            amanhã.  
 I arrive<sub>sg.1p.pre</sub> today and they arrive<sub>pl.ms.3p.pre</sub> tomorrow

(45) \*Eu cheguei        ontem e eles ~~chegarão~~        amanhã.  
 I arrive<sub>sg.1p.pst</sub> yesterday and they arrive<sub>pl.ms.3p.fut</sub> tomorrow

We thus formalize the identity conditions for Gapping via a  $\text{h\_s\_share}/2$  relation over two lists of domain objects:

$$(46) \text{h\_s\_share}(\boxed{1}, \boxed{2}) \leftarrow (\boxed{1}=\langle \rangle \wedge \boxed{2}=\langle \rangle) \vee \\ \boxed{1}=\left\langle \left[ \begin{array}{l} \text{SYN} \mid \text{CAT} \mid \text{HEAD} \quad \boxed{h} \\ \text{SEM} \mid \text{RELS} \langle \text{RELN } \boxed{r_1} \rangle, \dots, \langle \text{RELN } \boxed{r_n} \rangle \end{array} \right] \boxed{L_1} \right\rangle \wedge \\ \boxed{2}=\left\langle \left[ \begin{array}{l} \text{SYN} \mid \text{CAT} \mid \text{HEAD} \quad \boxed{h} \\ \text{SEM} \mid \text{RELS} \langle \text{RELN } \boxed{r_1} \rangle, \dots, \langle \text{RELN } \boxed{r_n} \rangle \end{array} \right] \boxed{L_2} \right\rangle \wedge \text{h\_s\_share}(\boxed{L_1}, \boxed{L_2})$$

Finally, we can rewrite the coordination construction (15) using standard constraints, while extending it to Gapping phenomena:

(47)  $cnj-cx \Rightarrow$ 

$$\left[ \begin{array}{l} \text{MTR} \left[ \begin{array}{l} \text{DOM } \boxed{A_1} \oplus \boxed{L_1} \oplus \boxed{I_1} \oplus \boxed{R_1} \oplus \boxed{C} \oplus \boxed{L_2} \oplus \boxed{R_2} \oplus \text{ne-list} \oplus \boxed{D_2} \\ \text{SYN } \boxed{0} \end{array} \right] \\ \\ \text{DTRS} \left\langle \begin{array}{l} \left[ \begin{array}{l} \text{DOM } \boxed{A_1} \oplus \boxed{L_1} \oplus \boxed{I_1} \oplus \boxed{R_1} \oplus \text{ne-list} \oplus \boxed{D_1} \\ \text{SYN } \boxed{0} \\ \text{CRD } - \end{array} \right], \\ \left[ \begin{array}{l} \text{DOM } \boxed{C} \langle \langle \text{[SYN } cnj] \rangle \rangle \oplus \boxed{A_2} \oplus \boxed{L_2} \oplus \boxed{I_2} \oplus \boxed{R_2} \oplus \text{ne-list} \oplus \boxed{D_2} \\ \text{SYN } \boxed{0} \\ \text{CRD } + \end{array} \right] \end{array} \right\rangle \end{array} \right] \\
\wedge \text{h\_f\_share}(\boxed{A_1}, \boxed{A_2}) \wedge \text{h\_f\_share}(\boxed{D_1}, \boxed{D_2}) \wedge \text{h\_s\_share}(\boxed{I_1}, \boxed{I_2}) \\
\wedge \boxed{I_2} = \text{ne-list} \Rightarrow [\text{SYN } \boxed{0} [\text{HD } \textit{verb}, \text{MRK } \textit{none}]] \in \boxed{I_2}$$

The sharing relations are integrated in the implicational constraint in the usual way, as conjoined constraints. Gapping arises if the shared non-peripheral lists  $\boxed{I_1}$  and  $\boxed{I_2}$  are resolved as non-empty, because the latter is not appended to the mother domain. There is also an extra constraint requiring that, if Gapping occurs, the conjuncts  $\boxed{0}$  must be verbal and unmarked, and that the corresponding head domain object must reside in the gap (as discussed in the previous section). E.g.:

(48) John likes caviar, and Mary, beans.

$$\left[ \begin{array}{l} \text{MTR} \mid \text{DOM } \boxed{A1} \langle \rangle \oplus \boxed{L1} \langle \text{John} \rangle \oplus \boxed{I1} \langle \text{likes} \rangle \oplus \boxed{R1} \langle \text{caviar} \rangle \oplus \\ \boxed{C} \langle \text{and} \rangle \oplus \boxed{L2} \langle \text{Mary} \rangle \oplus \boxed{R2} \langle \text{beans} \rangle \oplus \boxed{D2} \langle \rangle \\ \\ \text{DTRS} \left\langle \begin{array}{l} \left[ \text{DOM } \boxed{A1} \langle \rangle \oplus \boxed{L1} \langle \text{John} \rangle \oplus \boxed{I1} \langle \text{likes} \rangle \oplus \boxed{R1} \langle \text{caviar} \rangle \oplus \boxed{D1} \langle \rangle \right], \\ \left[ \text{DOM } \boxed{C} \langle \text{and} \rangle \oplus \boxed{A2} \langle \rangle \oplus \boxed{L2} \langle \text{Mary} \rangle \oplus \boxed{I2} \langle \text{likes} \rangle \oplus \boxed{R2} \langle \text{beans} \rangle \oplus \boxed{D2} \langle \rangle \right] \end{array} \right\rangle \end{array} \right]$$

FIGURE 1: Gapping of a shared internal sub-list (schematic)

The peripheral lists  $\boxed{A_2}$  and  $\boxed{D_2}$  are resolved as empty, but the internal lists are not. In particular,  $\boxed{I_2} = \langle \text{likes} \rangle$ . Consider a larger gap:

(49) Mia can help me today, and Jess, tomorrow.

$$\boxed{L_2} = \langle \text{[Jess]} \rangle, \boxed{I_2} = \langle \text{[can], [help], [me]} \rangle, \boxed{R_2} = \langle \text{[tomorrow]} \rangle$$

Consider also Gapping in null-subject languages like Portuguese:

(50) Comprei um livro e a Ana ~~comprei~~ uma revista.(I) bought<sub>1st</sub> a book and the Ana bought<sub>3rd</sub> a magazine

$$\boxed{L_1} = \langle \rangle, \quad \boxed{I_1} = \langle \text{[comprei]} \rangle, \quad \boxed{R_1} = \langle \text{[um, livro]} \rangle$$

$$\boxed{L_2} = \langle \text{[a, Ana]} \rangle, \quad \boxed{I_2} = \langle \text{[comprou]} \rangle, \quad \boxed{R_2} = \langle \text{[uma, revista]} \rangle$$

Mixed RNR cases arise from non-empty resolutions of both  $\boxed{I_2}$  and  $\boxed{D_2}$ :



- (51) I told Maria, and David, Anna, that the lecture was canceled.

$$\begin{aligned} \boxed{A_2} &= \langle \rangle & \boxed{D_2} &= \langle [that, the, lecture, was, canceled] \rangle \\ \boxed{L_2} &= \langle [David] \rangle, \boxed{I_2} &= \langle [told] \rangle, \boxed{R_2} &= \langle [Anna] \rangle \end{aligned}$$

Hankamer and Sag (1976, 409) identify an ellipsis pattern known as Stripping, where all but a small number of items are peripherally elided. Typically, the remnant is intonationally marked:

- (52) a. Flowers grow well here, and sometimes herbs.  
 b. *You* could sleep in the living room today, or *Susan*.  
 c. Either Mia can help me today, or Jess.

Hankamer and Sag (1976), Chao (1988), Hendriks (1995), and others have argued that Gapping and Stripping are very similar operations. Like Gapping, Stripping may not be recovered from unspoken context, the gap cannot occur backwards, it occurs in comparatives. Also, it does not impose inflection identity nor is it possible in subordinate clauses:

- (53) a. Flowers grow well here, and sometimes herbs.  
 b. Trees grow well here, and sometimes wheat.  
 c. \*We grow flowers in here, and over there is the place where we sometimes herbs.

In our account, such instances of Stripping arise as a special case of the constraints in (47), where the non-shared  $\boxed{R_2}$  list is resolved as empty but  $\boxed{L_2}$  and the shared  $\boxed{I_2}$  are non-empty:<sup>4</sup>

- (54) Flowers grow well here, and sometimes herbs.  

$$\begin{aligned} \boxed{A_2} &= \langle \rangle \\ \boxed{L_2} &= \langle [sometimes], [herbs] \rangle, \boxed{I_2} = \langle [grow], [well], [here] \rangle, \boxed{R_2} = \langle \rangle \\ \boxed{D_2} &= \langle \rangle \end{aligned}$$

#### 18.4.1 Order Domains and Compaction

A Linearization approach to NCC can use independently motivated linear order restrictions to predict possible ellipsis patterns. For instance, NPs are usually assumed to be bounding categories (i.e. *compact*) as they become arguments of a subcategorizing head. In our approach to NCC, this correctly rules out the examples in (55) because the append constraints can only split lists of domains, not domain objects:

- (55) a. \*I bought a gold fish, and Tim, ~~bought~~ [a parrot].  
 b. \*Don is [a painter and a fan of jazz], and Bob ~~is a fan~~ of blues.  
 c. \*[The best swimmer] lost and [the best runner] won. (ACC)  
 d. \*John loves [the house], and Mary adores [the house]. (RNR)

---

<sup>4</sup>Cases like ‘Mia helped me today and ~~helped~~ Jess (too)’ are captured as ACC.

In the case of PPs and relative clauses, these are usually assumed to be compacted and liberated from the NP domain (see partial compaction in Kathol and Pollard (1995)), due to extraposition phenomena:

- (56) a. [I] [bought] [a book] [yesterday] [about Asian philosophy].  
 b. [I] [met] [a man] [yesterday] [who reminded me of Nixon].

In our linearization approach to Gapping, the compaction of these phrases predicts several ‘Complex NP Constraint’ effects:

- (57) a. \*A man who loves cats and a man who ~~loves~~ dogs met outside.  
 b. \*I met a man who owns a Rolls-Royce, and my friend, a Ferrari.

Moreover, the liberation of PPs predicts that these can be remnants:<sup>5</sup>

- (58) a. Reeves gave a talk about Superstring theory, and Dawkins, about the evolution of extended phenotypes.  
 b. Yesterday we traveled sixty miles, and on the day before, fifty.

Relative clauses on the other hand, are very poor remnants. This might result from a very strong tendency for the relative to be parsed as attaching to the closest NP remnant:

- (59)?One broker may prefer stocks that go up, and another, that go down.

Beavers and Sag (2004) suggest that subordinate sentences should remain uncompactified in order to capture long-distance RNR in (31) above. This is also the case for long-distance Gapping in (27) and for Gapping in verb clusters as in (35b). However, the typical assumption is that subordinate sentences are compacted in order to prevent interleaving effects with several kinds of items (e.g. Kathol (2000, 95–97,153)). In the case of verb clusters, interleaving effects motivate Kathol (2000, 209) to assume that these are uncompactified in German. This is also the case for English, due to floating quantification and ‘either’ interleaving:

- (60) They will either have to [ lower prices] [or fire some personnel].

But even if some verb clusters required compaction, long-distance NCC already suggests that domains embedded in compacted verb-headed constituents should be accessible. We thus assume recursive domains (as is the case of Reape (1994) and Beavers and Sag (2004), where domain objects are of type *sign*) and distinguish between two kinds of compacted domains: some domain objects (e.g. a compacted S) are ‘transparent’ to NCC sharing constraints in the sense that the embedded subdomains can be shared, while others are ‘opaque’ to sharing

<sup>5</sup>In the case of (55b), the partial NP compaction would yield the compacted domain [*a, musician, and, a, fan*], which still disallows the gap in (55b).

(e.g. NPs, even if embedded in a compacted S).

We thus extend the append relation ‘ $\oplus$ ’ used in (47) to allow sharing peripherally embedded DOM objects given a parametrical constraint  $\tau$ :

$$(61) \quad \begin{aligned} \oplus_{d(\tau)}(e\text{-list}, \boxed{1}, \boxed{1}) &\leftarrow true \\ \oplus_{d(\tau)}(\langle \boxed{1} \boxed{2} \rangle, \boxed{3}, \langle \boxed{1} \boxed{4} \rangle) &\leftarrow \oplus_{d(\tau)}(\boxed{2}, \boxed{3}, \boxed{4}) \\ \oplus_{d(\tau)}(\langle [\tau_{\text{DOM}} \boxed{1}] \rangle, \langle [\tau_{\text{DOM}} \boxed{2}] \boxed{4} \rangle, \langle [\tau_{\text{DOM}} \boxed{3}] \boxed{4} \rangle) &\leftarrow \oplus_{d(\tau)}(\boxed{1}, \boxed{2}, \boxed{3}) \end{aligned}$$

The first two clauses correspond to the standard append relation, but the third clause allows access to a peripheral domain object, provided that the constraint  $\tau$  is satisfied. This argument thus specifies which kind of compacted domain object can be split by the constraint (it may correspond to the HEAD *pos* value, a *sign* description, or even a topological type). This enables the compaction of nominals to be distinguished from the compaction of subordinate clauses, as in (27a):

$$(62) \quad \begin{aligned} \boxed{B} &= \langle [\text{FRM}(\textit{that})], [\text{FRM}(\textit{doc})], [\text{FRM}(\textit{said})], [\tau_{\text{DOM}} \langle [I], [\textit{should}], [\textit{eat}], [\textit{salmon}] \rangle] \rangle \\ &\textit{with recursive domains, } \boxed{B} = \boxed{L} \oplus_{d(\tau)} \boxed{I} \oplus_{d(\tau)} \boxed{D} \textit{ may resolve as:} \\ \boxed{L} &= \langle [\text{FRM}(\textit{that})], [\text{FRM}(\textit{doc})] \rangle \\ \boxed{I} &= \langle [\text{FRM}(\textit{said})], [\tau_{\text{DOM}} \langle [I], [\textit{should}], [\textit{eat}] \rangle] \rangle \\ \boxed{D} &= \langle [\tau_{\text{FRM}}(\textit{salmon})] \rangle \end{aligned}$$

The  $\tau$  argument in the append constraints in Gapping should thus allow splitting compacted S and VP domain objects, i.e.  $\tau = \textit{phrase}[\text{HD } \textit{verb}, \text{MOD } \textit{none}]$ . Other NCC phenomena can be less restricted in this regard, for instance RNR does not show complex NP island effects:

- (63) a. The man who sells, and the woman who buys, antiques in the market met outside.  
 b. I know a man who loves, and Sue met a woman who hates, hiking at night.

Accordingly, the append constraint that yields the  $\boxed{D_1}$  and  $\boxed{D_2}$  sublists in (47) should be  $\tau = \textit{phrase}[\text{HD } \textit{verb}]$ . This allows the parametric append to split any verb-headed phrasal domain (S, VP or RelC).

In what concerns deletion of prepositions, there is a gradience effect which may depend on the nature of the remnants:

- (64) a. \*Jim reads a book to Fred, and Mary, Peter.  
 b. ?John is going to Japan, and his sister, Australia.  
 c. Jim reads to his brother, and Mary, our kids.

We note that (64b) improves with a longer pause in the gap, presumably because it promotes contrast and helps to legitimize the remnant. The unacceptability of (64a) might be intensified by the oddness sometimes caused by adjacent remnants headed by identical items. Consider the

following well-known contrast, triggered by the post-gap remnants:<sup>6</sup>

- (65) a. John gave a bike to the boy, and Tim, a doll to the girl.  
 b. ?\*John gave the boy a bike, and Tim, the girl a doll.

Thus it may be that the ellipsis of argument marking prepositions is tolerated in certain conditions, or that ellipsis is possible in principle but that interacting factors cause gradient results. In that case, we would have to allow the parametric append to also access PPs.

#### 18.4.2 Discontinuous Gapping

Jackendoff (1971, 24–25), Sag (1976, 148–166) and others consider discontinuous Gapping deletions:

- (66) a. John kissed Susan at the party, and Peter, Mary.  
 b. Dexter wants Watford to win, and Warren, Ipswich.  
 c. Peter took Susan home, and John, Wendy.

Our sharing constraints need to be reformulated to also account for this pattern. This can be done by allowing the right periphery of the internal sub-lists to share a discontinuous list via the shuffle ‘ $\circ$ ’ operator:

$$(67) \text{ cnj-cx} \Rightarrow \left[ \begin{array}{l} \text{MTR} \left[ \begin{array}{l} \text{DOM } [A_1] \oplus [L_1] \oplus [I_1] \oplus [P_1 \circ R_1] \oplus [C] \oplus [L_2] \oplus [R_2] ]_{ne-list} \oplus [D_2] \\ \text{SYN } [0] \end{array} \right] \\ \\ \text{DTRS} \left\langle \begin{array}{l} \left[ \begin{array}{l} \text{DOM } [A_1] \oplus [L_1] \oplus [I_1] \oplus [P_1 \circ R_1] ]_{ne-list} \oplus [D_1] \\ \text{SYN } [0] \\ \text{CRD } - \end{array} \right] , \\ \left[ \begin{array}{l} \text{DOM } [C] \langle ([\text{SYN } cnj]) \rangle \oplus [A_2] \oplus [L_2] \oplus [I_2] \oplus [P_2 \circ R_2] ]_{ne-list} \oplus [D_2] \\ \text{SYN } [0] \\ \text{CRD } + \end{array} \right] \end{array} \right\rangle \end{array} \right]$$

$$\wedge \text{h\_f\_share}([A_1], [A_2]) \wedge \text{h\_f\_share}([D_1], [D_2])$$

$$\wedge \text{h\_s\_share}([I_1], [I_2]) \wedge \text{h\_s\_share}([P_1], [P_2])$$

$$\wedge [I_2] = ne-list \Rightarrow [\text{SYN } [0] [\text{HD } verb, \text{MRK } none]] \in [I_2]$$

The internal right periphery list is now a shuffle of two sub-lists  $[P_2] \circ [R_2]$ . This extra constraint allows sharing of non-adjacent items:

- (68)  $\langle [John], [took], [Wendy], [home] \rangle$  yields  $\langle [John], [Wendy] \rangle$   
 given the following resolution:  
 $[L_2] = \langle [John] \rangle$ ,  $[I_2] = \langle [took] \rangle$ ,  $[R_2] = \langle [Wendy] \rangle$ , and  $[P_2] = \langle [home] \rangle$

The non-discontinuous Gapping data discussed in the previous sections is obtained from resolving the shared  $[P_1]$  and  $[P_2]$  lists as empty.

<sup>6</sup>This may reduce to a generalization similar to the Obligatory Contour Principle.

## 18.5 Conclusion

We propose an account of Gapping in HPSG, integrated in a more general constraint for Non-Constituent Coordination, following Beavers and Sag (2004). Here, a unique coordination construction captures several patterns by allowing shared items to be absent in the mother, using structure-sharing constraints over the domain objects contributed by the local daughters. Further research is required to capture gradience effects (as experimentally observed in Keller (2001)), which may result from perceptual, contextual, and prosodic factors.

## Acknowledgments

I thank Palmira Marrafa, Ivan Sag, Philip Hofmeister, and Raquel Amaro for comments, discussion and encouragement. All omissions and remaining errors are solely mine. This research was supported by Fundação para a Ciência e a Tecnologia (SFRH/BD/13880/2003).

## References

- Beavers, John and Ivan A. Sag. 2004. Ellipsis and apparent non-constituent coordination. In S. Müller, ed., *Proceedings of the HPSG-2004 Conference, Center for Computational Linguistics, Katholieke Universiteit Leuven*, pages 48–69. Stanford: CSLI Publications.  
<http://csli-publications.stanford.edu/HPSG/5/>.
- Bresnan, Joan. 1974. On the position of certain clause-particles in phrase structure. *Linguistic Inquiry* 5:614–619.
- Chao, Wyn. 1988. *On Ellipsis*. New York and London: Garland.
- Crysmann, Berthold. 2000. Clitics and coordination in linear structure. In B. Gerlach and J. Grijzenhout, eds., *Clitics in Phonology, Morphology and Syntax*, vol. 36, pages 121–159. *Linguistics Today*, John Benjamins.
- Crysmann, Berthold. 2002. *Constraint-based Coanalysis - Portuguese Cliticisation and Morphology-Syntax Interaction in HPSG*. PhD Thesis, Saarland University.
- Crysmann, Berthold. 2003. An asymmetric theory of peripheral sharing in HPSG. In G. Jäger, P. Monachesi, G. Penn, and S. Wintner, eds., *Proceedings of Formal Grammar 2003*, pages 47–62. ESSLLI 2003, Vienna.  
<http://www.cs.haifa.ac.il/~shuly/fg/proceedings.html>.
- Hankamer, Jorge. 1973. Unacceptable ambiguity. *Linguistic Inquiry* 4:17–28.
- Hankamer, Jorge and Ivan A. Sag. 1976. Deep and Surface Anaphora. *Linguistic Inquiry* 7(3):391–428.

- Hendriks, Petra. 1995. *Comparatives and Categorical Grammar*. Phd. diss., University of Groningen.
- Jackendoff, Ray. 1971. Gapping and related rules. *Linguistic Inquiry* 2:21–35.
- Kathol, Andreas. 1995. *Linearization-Based German Syntax*. PhD Thesis, Ohio State University.
- Kathol, Andreas. 2000. *Linear Syntax*. Oxford University Press.
- Kathol, Andreas and Carl Pollard. 1995. Extraposition via complex domain formation. In *Proc. of the 33th ACL*, pages 174–180. Cambridge, Mass.
- Keller, Frank. 2001. Experimental evidence for constraint competition in gapping constructions. In G. Müller and W. Sternfeld, eds., *Competition in Syntax*, pages 211–248. Berlin: Mouton de Gruyter.
- Kuno, Susumo. 1976. Gapping: a functional analysis. *Linguistic Inquiry* 7:300–318.
- Lappin, Shalom. 1999. An HPSG account of antecedent contained Ellipsis. In S. Lappin and E. Benmamoun, eds., *Fragments: Studies in Ellipsis and Gapping*. Oxford University Press.
- Lasnik, Howard and M. Saito. 1992. *Move  $\alpha$ : Conditions on its application and output*. MIT Press.
- Neijt, Anneke. 1979. *Gapping*. Foris, Dordrecht.
- Pesetsky, David. 1982. *Paths and Categories*. Phd. dissertation, MIT.
- Pollard, Carl and Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. Chicago University Press, Chicago.
- Reape, Mike. 1994. Domain Union and Word Order Variation in German. In J. Nerbonne, K. Netter, and C. Pollard, eds., *German in Head-Driven Phrase Structure Grammar*, pages 151–197. CSLI Lecture Notes 46.
- Ross, John Robert. 1970. Gapping and the order of constituents. In M. Bierwisch and K. Heidolph, eds., *Progress in Linguistics*, pages 249–259. The Hague: Mouton.
- Sag, Ivan A. 1976. *Deletion and Logical Form*. Phd. Dissertation, MIT.
- Sohn, Keun-Won. 1999. Deletion or Copying? Right-Node Raising in Korean and Japanese. In *The Proceedings of the 2nd GLOW Meeting in Asia*.
- Yatabe, Shuichi. 2001. The syntax and semantics of left-node raising in Japanese. In D. Flickinger and A. Kathol, eds., *Proceedings of the 7th International Conference on Head-Driven Phrase Structure Grammar*, pages 325–344. Stanford, CA: CSLI Publications.  
<http://csli-publications.stanford.edu/HPSG/1/hpsg00.html>.

---

# Further Properties of Path-Controlled Grammars

CARLOS MARTÍN-VIDE AND VICTOR MITRANA

## Abstract

A path controlled grammar consists in a pair of CFGs,  $G_1, G_2$ , where  $G_2$  generates a language over the total alphabet of  $G_1$ . A word  $w$  generated by  $G_1$  is accepted only if there is a derivation tree  $\tau$  of  $w$  with respect to  $G_1$  such that there exists a path in  $\tau$ , from the root to a leaf node, having the nodes marked by a word  $x$  which is in  $L(G_2)$ . We show that these grammars are slightly more powerful than CFGs, but they preserve all the properties of CFGs: closure properties, pumping lemmas, decision properties, polynomial recognition. Moreover, many of these properties are mainly based on the corresponding properties of CFGs.

**Keywords** PATH-CONTROLLED GRAMMAR, CLOSURE PROPERTIES, DECISION PROPERTY, POLYNOMIAL RECOGNITION.

This work is a continuation of the investigation started in Marcus et al. (2001), where a new-old type of control on context-free grammars is considered. This type of control is extracted and abstracted from a paper (Bellert (1965)) with very solid linguistic motivations. The goal of this paper is to complete the picture of path-controlled grammars started in Marcus et al. (2001) with some mathematical properties which are missing from the aforementioned work: closure and decidability properties, including a polynomial recognition algorithm.

## 19.1 Introduction

The work by Bellert (1965) is very well motivated, starting from the assertion that the natural language is not context-free. The paper not only motivates a theoretical study by this assertion, but, after intro-

ducing the “relational grammars”, a consistent case study is examined, using the new formalism for modelling several constructions from Polish. Bellert applies her relational grammar to the generation of Polish kernel sentences. In this application, 13 relations are considered, mainly concerning the agreement in gender, number or case between the noun and the predicate.

Informally, the framework is the following one. One considers a context-free grammar (CFG for short)  $G = (N, T, S, P)$  and a way to select from all possible derivation trees in  $G$  only those trees with certain properties, specified by a set of *tuple-grammars* and a *relation* on the rules of these tuple-grammars. More specifically, one gives several  $k$ -tuples of CFGs,  $(G_1, \dots, G_k)$ , of the form  $G_i = (N_i, N \cup T, S_i, P_i)$ . Note that these grammars have as the terminal alphabet the total alphabet of  $G$ . The grammars from such a tuple work in parallel, over separate sentential forms, also observing the restriction imposed by a relation  $\rho \subseteq P_1 \times \dots \times P_k$ . In a derivation of the  $k$ -tuple of grammars  $(G_1, \dots, G_k)$  we have to synchronously use  $k$  rules related by  $\rho$ . The  $k$ -tuples  $(w_1, \dots, w_k)$  of words generated in this way by  $(G_1, \dots, G_k)$  are then used for selecting from the derivation trees of  $G$  only those trees which have paths marked by  $w_1, \dots, w_k$ , markers being assigned to nodes.

The definition of relational grammars contains an idea which has been investigated later in the regulated rewriting area: to impose some restriction on the paths present in a derivation tree of a CFG. One takes two CFGs,  $G_1, G_2$ , such that the rules of  $G_1$  are labeled in a one-to-one manner with labels from a finite set of labels  $V_L$  and  $G_2$  generates a language over  $V_L$ . A word  $w$  generated by  $G_1$  is accepted only if there is a derivation tree  $\tau$  of  $w$  with respect to  $G_1$  such that all independent paths in  $\tau$  are labeled (labels are now assigned to edges) by words in  $L(G_2)$ . In Khabbaz (1974) one defines an infinite hierarchy of languages based on this idea. The second class in this hierarchy is exactly the class of languages defined by a pair of grammars  $(G_1, G_2)$  as above with  $G_1$  a linear grammar. Since at each point in a derivation of a linear CFG there is at most one nonterminal to expand, every derivation is associated with a single control word. This approach has been extended to arbitrary CFGs in Weir (1992). Both formalisms are special cases of a more general formalism proposed by Ginsburg and Spanier (1968). The concept of *tree controlled grammar* introduced in Culik and Maurer (1977) appears to have some weak connections with this approach; a tree controlled grammar is also a pair of CFGs  $(G, G')$  but a derivation tree  $\tau$  of a word in  $G$  is “accepted” if the words labelling the nodes considered from left to right on each level in  $\tau$ , excepting the



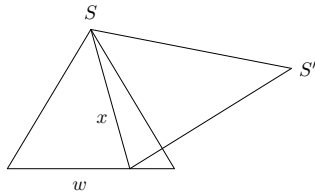


FIGURE 1: A Word Generated by  $G_1$  Under the Partial Control of  $G_2$

last one, can be generated by  $G'$ .

In Marcus et al. (2001) one proposes a bit different approach: one takes two CFGs,  $G_1, G_2$ , where  $G_2$  generates a language over the total alphabet of  $G_1$ . A word  $w$  generated by  $G_1$  is accepted only if there is a derivation tree  $\tau$  of  $w$  with respect to  $G_1$  such that there exists a path in  $\tau$ , from the root to a leaf node, having the nodes marked by a word  $x$  which is in  $L(G_2)$ . For an intuitive representation of this idea, one should have in mind the fact that a word  $w$  is generated by  $G_1$  under the “partial control” of  $G_2$  if there exist two derivation trees,  $\tau_1, \tau_2$ , the first one with respect to  $G_1$  and the second one with respect to  $G_2$ , such that the first tree has  $w$  as its frontier, while the second tree, having  $x$  as its frontier, is “orthogonal” to the first tree and its frontier word  $x$  describes a path in the first tree. Figure 1 illustrates this idea.

A pair of two grammars as above is called a path-controlled grammar in Marcus et al. (2001).

The main difference between the “partial control” defined above and the control on all independence paths proposed in Khabbaz (1974), Weir (1992), and Palis and Shende (1995) does not consists in the way of assigning words to paths (this translation can be done by a generalized sequential machine) but in the fact that the “partial control” requires the existence of just one path in the derivation tree labeled by a control word while the other control type requires that all independence paths to be labeled by control words. The relationship between the two formalisms seems an easy question at first sight: each path-controlled grammar can be simulated by a control grammar in the sense of Weir (1992) (more precisely, a labeled distinguished grammar with a control context-free language). However, we were not able to prove or disprove this; actually this depends on the definition of a control grammar. If one allows different labels for the same rule depending on the distinguished symbol identified in the right-hand side of that rule the statement is true. By space reason we skip the proof which is rather simple. However, if this is not permitted, then the problem could not be settled. It

is worth mentioning that the properties reported in Palis and Shende (1992) and Palis and Shende (1995), including the complexity of the recognition problem, were proved for this latter case. Consequently, it seems that they cannot be easily extended to path-controlled grammars.

Having this in mind, we considered that it is worth presenting a more complete picture of path-controlled grammars. In Marcus et al. (2001) one mainly investigates the generative power and a very few linguistically oriented properties of path-controlled grammars. These grammars are slightly more powerful than CFGs, but they preserve all the properties of CFGs: closure properties, pumping lemmas, decision properties, polynomial recognition. Moreover, many of these properties are mainly based on the corresponding properties of CFGs. As a consequence, these devices belong to the so-called *mildly context-sensitive grammars* proposed in Joshi (1985). This extra power appears to be adequate for handling various phenomena which require more formal power than CFGs like duplication, crossed dependencies, multiple agreements up to four positions. Path-controlled grammars may also be viewed as a tree-generating formalism with some attractive properties for characterizing the strong generative power of grammars. We are not concerned here with their capacity to characterize the structural description associated with sentences, though this is a natural matter of great importance. The goal of this paper is to complete the picture of path-controlled grammars started in Marcus et al. (2001) with some mathematical properties which are missing from the aforementioned work: closure and decidability properties, including a polynomial recognition algorithm.

## 19.2 Path-Controlled Grammars

We use the standard formal language theory notions and notations, as available in many monographs, see, e.g., Hopcroft and Ullman (1979). In particular,  $V^*$  is the free monoid generated by the alphabet  $V$  under the operation of concatenation,  $\lambda$  is the empty word, and  $|x|$  is the length of the word  $x \in V^*$ . As usual, when comparing two languages, the empty word is ignored, that is, we consider  $L_1$  equal to  $L_2$  if  $L_1 - \{\lambda\} = L_2 - \{\lambda\}$ .

Given a CFG  $G = (N, T, S, P)$  without erasing rules, with derivations in  $G$  we associate derivation trees in the well-known manner. Let  $S \Longrightarrow^* w$  be a terminal derivation in  $G$  and  $\tau$  its associated tree. Each path from the root of  $\tau$  to a leaf is described by a word of the form  $SA_1A_2 \dots A_r a$ , with  $A_i \in N$ ,  $1 \leq i \leq r$ ,  $r \geq 0$ , and  $a \in T$ . We denote

by  $path(\tau)$  the language of all these words, describing paths in  $\tau$ , by  $path(x)$  the union of all the languages  $path(\tau)$ , where  $\tau$  is a derivation tree for  $x$  in  $G$ , and by  $path(G)$  the union of all these languages. Note that we consider that any path in a derivation tree ends by a node labelled by a terminal symbol.

A *path-controlled* grammar is a pair  $\gamma = (G, G')$ , where  $G = (N, T, S, P)$  and  $G' = (N', N \cup T, S', P')$  are context-free grammars. The language generated by  $\gamma$  is

$$L_{\exists}(\gamma) = \{w \in L(G) \mid path(w) \cap L(G') \neq \emptyset\}.$$

In other words, the language generated by  $\gamma$  is the yield of the tree language obtained from the derivation trees of  $G_1$  which are “partially accepted” by  $G_2$ .

We denote by  $PC_{\exists}(CFG, CFG)$  the family of languages  $L_{\exists}(\gamma)$ , where  $\gamma = (G, G')$  is a path-controlled grammar having both component CFGs. It is of no interest to consider a control on paths imposed by regular grammars as this control does not increase the power of CFGs as shown in Marcus et al. (2001)

### 19.3 Closure Properties of $PC_{\exists}(CFG, CFG)$

**Proposition 1**  *$PC_{\exists}(CFG, CFG)$  is closed under the following operations: union, intersection with regular languages, left and right concatenation with context-free languages, substitution with  $\lambda$ -free context-free languages, non-erasing homomorphism. It is not closed under intersection.*

**Proof** We skip the proof for *union* and *right/left concatenation* with context-free languages which is trivial.

*Intersection with regular languages:* Let  $\gamma = (G, G')$  be a path-controlled grammar and  $\mathcal{A}$  be a finite automaton. By the standard proof one constructs the CFG  $G_1$  generating  $L(G) \cap L(\mathcal{A})$ . Remember that the nonterminals of this grammars are triples of the form  $(q, A, q')$ , where  $A$  is a nonterminal of  $G$  and  $q, q'$  are states of  $\mathcal{A}$ . We now construct a grammar  $G'_1$  generating the language  $s(L(G'))$ , where  $s$  is a finite substitution defined by

$$s(X) = \begin{cases} \{(q, X, q') \mid q, q' \text{ are states of } \mathcal{A}\}, & X \text{ is a nonterminal of } G \\ \{X\}, & X \text{ is a terminal of } G. \end{cases}$$

It is rather plain that  $L_{\exists}(G_1, G'_1) = L_{\exists}(\gamma) \cap L(\mathcal{A})$ .

*Substitution with  $\lambda$ -free context-free languages:* Let  $\gamma = (G, G')$  be a path-controlled grammar with  $G = (N, T, S, P)$ ,  $T = \{a_1, a_2, \dots, a_n\}$ , and  $s : T^* \rightarrow 2^{U^*}$  be a substitution such that  $s(a_i) = L(G_i)$ , with  $G_i = (N_i, U, S_i, P_i)$  being a CFG without  $\lambda$ -rules for all  $1 \leq i \leq n$ .

If  $\widehat{G}$  is the CFG generating the language  $s(L(G))$  and  $\widetilde{G}$  is a grammar generating the language  $g(L(G'))$ , where  $g$  is a substitution which keeps unchanged any symbol from  $N$  and replace each  $a_i$  by a word in the regular set  $S_i N_i^* U$ , then  $L_{\exists}(\widehat{G}, \widetilde{G}) = s(L_{\exists}(\gamma))$  holds. As a direct consequence, the closure of  $PC_{\exists}(CFG, CFG)$  under non-erasing homomorphism follows.

*Intersection:* In Marcus et al. (2001), one proves that path-controlled grammars with linear CFG components “can count to four”, that is they can generate the language  $\{a_1^n a_2^n a_3^n a_4^n \mid n \geq 1\}$ , but even path-controlled grammars with CFGs components “cannot count to five”. By the closure under left and right concatenation with context-free languages, the statement follows.  $\square$

We finish this section with a brief discussion about the significance of closure properties of a class of languages with respect to natural language. It should be emphasized that, though closure of operations is a very natural and elegant mathematical property, however non-closure properties seem to be natural too. One may argue that it is possible to construct a grammatical formalism for a language in the following way. One looks for a decomposition of the language into some independent fragments, constructs auxiliary grammars for such fragments, and then obtains the desired grammar by closure under union of the auxiliary grammars. Following this idea, the grammatical model should satisfy the closure property under union, which is fulfilled by almost all mildly context-sensitive devices including ours. Along the same lines, we can imagine another way of constructing a grammatical formalism for a language  $L$ . It consists of identifying a set of conditions  $c_1, c_2, \dots, c_n$  which define the correctness of  $L$ . Then, one tries to define a grammar generating the set  $L(c_i)$  of all words observing the condition  $c_i$  and ignoring the other conditions. It is expected that the intersection  $\bigcap_{i=1}^n L(c_i)$  gives exactly the language  $L$ . Now, the closure property which should be verified by the model is the one under intersection, which is not verified by many mildly context-sensitive mechanisms, among them the one investigated in this paper.

As far as the closure under concatenation is concerned, some authors consider that this is basic to natural languages while others have the opposite opinion, see, e.g., Kudlek et al. (2003) for such a discussion. It is worth mentioning here that the closure under concatenation of our grammatical formalism remains unsettled.

### 19.4 Decision Properties of $PC_{\exists}(CFG, CFG)$

We start this section by recalling a very simple result from Thatcher (1967) which will be useful in the sequel. Since we shall refer to this construction, for sake of completeness we prefer to give a construction as well

**Lemma 1** *For any context-free grammar  $G$ ,  $path(G)$  is regular.*

**Proof** Let  $G = (N, T, S, P)$  be a CFG, for later purposes we assume that it is reduced, namely each nonterminal is reachable from  $S$  and produces a terminal word. We construct the deterministic finite automaton  $\mathcal{A}_G = (Q, N \cup T, \delta, q_0, \{q_f\})$ , where  $Q = \{q_0, q_f\} \cup \{[A] \mid A \in N\}$  and  $\delta$  defined by

$$\begin{aligned} \delta(q_0, S) &= [S], & \delta([A], B) &= [B], & \delta([A], a) &= q_f, \\ & & \text{if } A \rightarrow uBv \in P, & & \text{if } A \rightarrow uav \in P, \end{aligned}$$

where  $u, v \in (N \cup T)^*$  and  $a \in T$ . Clearly,  $path(G) = L(\mathcal{A})$ . □

In the same work one proves that each language generated by a path-controlled grammar is a matrix language (see Abraham (1965) and Dassow and Păun (1989) for the definition and properties of matrix languages). Since the emptiness problem is decidable for matrix grammars via a reduction to the reachability problem in vector addition systems shown to be decidable in Kosaraju (1982), we have:

**Proposition 2** *The emptiness problem is decidable for path-controlled grammars.*

However, a direct proof can be obtained as an immediate consequence of the above lemma. Indeed, it is obvious that for any  $\gamma = (G, G')$ , where  $G$  is reduced,  $L_{\exists}(\gamma) \neq \emptyset$  if and only if  $path(G) \cap L(G') \neq \emptyset$ . It suffices to remember that the class of context-free languages is effectively closed under intersection with regular sets and the emptiness problem is decidable for CFGs.

In Marcus et al. (2001) one presents a pumping lemma for languages generated by path-controlled grammars very similar to that for context-free languages. Since the membership problem is decidable for matrix grammars without erasing rules, by the pumping lemma from Marcus et al. (2001) one immediately infers that:

**Proposition 3** *The finiteness problem is decidable for path-controlled grammars.*

However, we now present a more efficient algorithm for this problem. Let  $\gamma = (G, G')$  be a path-controlled grammar; without loss of generality we may assume that  $G$  is reduced and without chain rules of the form  $A \rightarrow B$ , where  $A$  and  $B$  are nonterminals. If this is not the case,

we eliminate these rules by the well known algorithm and then construct a *gsm*  $M$  which transforms each word in  $L(G')$  as follows: when  $M$  scans a nonterminal  $A$  of  $G$ , it nondeterministically either lets it unchanged, or removes it provided that there is a rule  $B \rightarrow A$  in  $G$  and  $B$  was just scanned before  $A$ . Therefore, subwords which correspond to recurrent derivations in  $G$  which produce nothing are either erased or not (the second case prevents the possibility to also have productive derivations with the same nonterminals). Because the class of context-free languages is closed under *gsm* mappings, the language  $M(L(G'))$  is still context-free.

Clearly, if  $path(G) \cap L(G')$  is an infinite set, then  $L_{\exists}(\gamma)$  is infinite as well. The converse does not hold, since one may have an arbitrarily long path described by a word in  $path(G)$  such that a prefix of this word is the prefix of a word in  $path(G) \cap L(G')$ . Therefore, if  $path(G) \cap L(G')$  is finite but non-empty, we construct the two sets  $\langle A \rangle_N$  and  $\langle A \rangle_T$  for each nonterminal  $A \in N$  as follows:

- $\langle A \rangle_N$  contains all nonterminals  $C$  such that there exists a rule  $A \rightarrow \alpha \in P$  satisfying the following condition: There is a nonterminal  $B$  (which may be  $C$  as well) such that both  $B$  and  $C$  appear in  $\alpha$  (if  $B = C$ , then  $\alpha$  contains at least two occurrences of  $B$ ) and the language accepted by the automaton  $\mathcal{A}_G$  with the initial state  $[B]$  is infinite.

- $\langle A \rangle_T$  contains all terminals under the same circumstances as above.

Now it is easy to see that  $L_{\exists}(\gamma)$  is infinite if and only if either  $path(G) \cap L(G')$  is infinite or

$$\bigcup_{A \in N} ((path(G) \cap L(G')) \cap (N^* \{A\} \langle A \rangle_N N^* T \cup N^* \{A\} \langle A \rangle_T)) \neq \emptyset.$$

By the closure and decision properties of the class of context-free languages this condition can be algorithmically checked.

We now present an undecidable problem which, in its turn, is based on the undecidability of the universality problem for CFGs.

**Proposition 4** *One cannot algorithmically decide whether or not the language generated by a given path-controlled grammar is context-free.*

**Proof** Let  $L \subseteq V^*$  be an arbitrary context-free language and  $L' \subseteq U^*$  be a non-context-free language in  $PC_{\exists}(CF, CF)$ . Assume that  $V \cap U = \emptyset$ . By the closure properties in Proposition 1 the language  $LU^* \cup V^*L'$  can be generated by a path-controlled grammar  $\gamma$ . If  $L = V^*$ , then  $L_{\exists}(\gamma) = V^*U^*$ , hence it is context-free. If  $L \neq V^*$ , then we consider  $w \in V^* \setminus L$  and observe that  $L_{\exists}(\gamma) \cap \{w\}U^* = \{w\}L'$ , hence  $L_{\exists}(\gamma)$  cannot be context-free. Indeed, if  $L_{\exists}(\gamma)$  were context-free, then  $\{w\}L'$  would be context-free, hence  $L'$  would be context-free, a contradiction.

Consequently,  $L_{\exists}(\gamma)$  is context-free if and only if  $L = V^*$  which is undecidable.  $\square$

### 19.5 Polynomial Recognition

The most important decision property for any class of grammars which aims to be considered as a model of syntax is the membership (recognition) problem, and especially the complexity of this problem provided that it is decidable. In this section we present two polynomial algorithms for recognizing languages generated by path-controlled grammars. We stress that such an algorithm might not be directly inferred from the work of Palis and Shende (1992) since it is likely that no efficient algorithm for simulating a path-controlled grammar with a control grammar in the sense of Palis and Shende (1992) exists.

We first give an algorithm arising from Propositions 1 and 2. Let  $\gamma = (G, G')$  be a path-controlled grammar and  $w$  be a word of length  $n$  over the terminal alphabet of  $G$ . Assume that  $\mathcal{A}$  is a finite automaton recognizing  $w$ ; grammars  $G_1$  and  $G'_1$  from the first part of the proof of Proposition 1 can be constructed in polynomial time. According to Proposition 2,

$$w \in L_{\exists}(\gamma) \text{ iff } \text{path}(G_1) \cap L(G'_1) \neq \emptyset.$$

By Lemma 1 the automaton recognizing  $\text{path}(G_1)$  as well as the CFG generating  $\text{path}(G_1) \cap L(G'_1)$  can be constructed in polynomial time. Now, the emptiness problem for a language generated by a given CFG can be algorithmically solved in polynomial time. The worst case complexity of this algorithm, which is  $O(n^{12})$ , can be inferred from the proof of the next statement which proposes a bit more efficient algorithm.

**Proposition 5** *The language generated by a given path-controlled grammar can be recognized in  $O(n^{10})$  time.*

**Proof** The proof is mainly based on the well-known Cocke-Younger-Kasami (CYK for short) algorithm for the context-free languages recognition. This algorithm appears in many places, it may be found in the pioneering works of Kasami (1965) and Younger (1967). Let  $\gamma = (G, G')$  be a path-controlled grammar with  $G = (N, T, S, P)$  a reduced CFG in the Chomsky Normal Form (CNF) and  $L(G') \subseteq SN^*T$ . This supposition does not decrease the generality of the reasoning since if  $G$  is not in the CNF, then we can transform it by the classic algorithm and  $G'$  accordingly. More precisely, in the first step we allow only terminal rules of the form  $X_a \rightarrow a$ , where  $X_a$  is a new nonterminal for each  $a \in T$ . Each word in  $L(G')$  is modified by inserting  $X_a$  before its last  $a$ . In the second step we eliminate all the chain rules and trans-

form  $G'$  as shown in the finiteness algorithm. In the last step, each rule  $r : A \rightarrow B_1 B_2 \dots B_k \in P$  with  $k \geq 3$  is transformed equivalently in the set of rules  $\{A \rightarrow B_1 X_1^{(r)}, X_1^{(r)} \rightarrow B_2 X_2^{(r)}, \dots, X_{k-2}^{(r)} \rightarrow B_{k-1} B_k\}$ . Note that the new nonterminals identify exactly the rule for which they have been used. We construct a *gsm* which nondeterministically inserts between any two consecutive nonterminal symbols it scans either nothing or a sequence  $X_1^{(p)} X_2^{(p)} \dots X_q^{(p)}$ , where  $p$  is the label of a rule in  $P$  and  $1 \leq q \leq m - 2$ ,  $m \geq 3$  being the length of the right-hand side of the rule  $p$ .

For a given word  $w = a_1 a_2 \dots a_n \in T^+$  we construct the deterministic finite automaton  $\mathcal{A}_w = (Q, N \cup T, \delta, q_0, \{q_f\})$ , where

$$Q = \{q_0, q_f\} \cup \{[A, i, j] \mid A \in N, 1 \leq i \leq j \leq n\}.$$

For any pair  $1 \leq i \leq j \leq n$  we denote by  $N(i, j) = \{A \in N \mid A \Rightarrow^* a_i a_{i+1} \dots a_j\}$ ; these sets can be computed by the CYK algorithm. Then we set

- (i)  $\delta([A, i, i], a_i) = q_f$  for each  $A \in N(i, i)$ ,  $1 \leq i \leq n$ ,
- (ii)  $\delta([A, i, j], B) = [B, i, k]$  and  $\delta([A, i, j], C) = [C, k + 1, j]$   
for each  $A \in N(i, j)$ ,  $1 \leq i \leq k < j \leq n$ , such that  
 $B \in N(i, k)$ ,  $C \in N(k + 1, j)$ ,  $A \rightarrow BC \in P$ ,
- (iii)  $\delta(q_0, A) = [A, 1, n]$  for all  $A \in N(1, n)$ .

Clearly,  $w \in L_{\exists}(\gamma)$  iff  $L(\mathcal{A}_w) \cap L(G') \neq \emptyset$ . As far as the complexity of this algorithm is concerned, the automaton  $\mathcal{A}_w$  (which has  $O(n^2)$  states) can be constructed in time  $O(n^3)$ , the number of nonterminals and rules in the context-free grammar  $G''$  generating  $L(\mathcal{A}_w) \cap L(G')$  is  $O(n^4)$  and  $O(n^6)$ , respectively. Applying the well-known algorithm, see, e.g., Hopcroft and Ullman (1979), to solve the emptiness of  $L(G'')$  we are done.  $\square$

## 19.6 Conclusion

This note tries to complete the picture of path-controlled grammars started in Marcus et al. (2001) with some mathematical properties which are missing from the aforementioned work: closure and decidability properties, including a polynomial recognition algorithm.

There is an interesting difference between our formalism and that considered in Weir (1992) and a series of subsequent works in that the grammars studied here require only the existence of a path labeled with a word in the control set while Weir's formalism requires, in essence, all independence paths to be labeled with words in the control set. On the other hand, all of the properties reported here as well as in Marcus



et al. (2001) are quite similar to those of Weir's corresponding classes. In spite of this fact, we were not able to emphasize the substantial differences, if any, between the two formalisms.

**Acknowledgments.** Some critical suggestions and comments of the referees which improved the presentation are gratefully acknowledged.

## References

- Abraham, S. 1965. Some Questions of Phrase-Structure Grammars. *Computational Linguistics* 4:61–70.
- Bellert, I. 1965. Relational Phrase Structure Grammar and Its Tentative Applications. *Information and Control* 8:503–530.
- Culik, K. and H. A. Maurer. 1977. Tree Controlled Grammars. *Computing* 19:120–139.
- Dassow, J. and G. Păun. 1989. *Regulated Rewriting in Formal Language Theory*. Berlin: Springer-Verlag.
- Ginsburg, S. and E. H. Spanier. 1968. Control Sets on Grammars. *Math. Systems Theory* 2:159–177.
- Hopcroft, J. E. and J. D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley.
- Joshi, A. K. 1985. How Much Context-Sensitivity Is Required to Provide Reasonable Structural Descriptions? Tree Adjoining Grammars. In D. R. D. et al, ed., *Natural Language Processing: Psycholinguistic, Computational, and Theoretic Perspectives*. New York: Cambridge Univ. Press.
- Kasami, T. 1965. *An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages*. Bedford Mass.: Air Force Cambridge Research Laboratory.
- Khabbaz, N. A. 1974. A Geometric Hierarchy of Languages. *J. Comput. System Sci.* 8:142–157.
- Kosaraju, S. R. 1982. Decidability of Reachability in Vector Addition Systems. In *14th Symposium on Theory of Computing*.
- Kudlek, M., C. Martín-Vide, A. Mateescu, and V. Mitrană. 2003. Contexts and the Concept of Mild Context-Sensitivity. *Linguistics and Philosophy* 26:703–725.
- Marcus, S., C. Martín-Vide, V. Mitrană, and G. Păun. 2001. A New-Old Class of Linguistically Motivated Regulated Grammars. In W. Daelemans, K. Sima'an, J. Veenstra, and J. Zavrel, eds., *7th International Conference*

*on Computational Linguistics in Netherlands*, Language and Computers-Studies in Practical Linguistics. Amsterdam: Rodopi.

Palis, M. A. and S. M. Shende. 1992. Upper Bounds on Recognition of a Hierarchy of Non-Context-Free Languages. *Theoret. Comput. Sci.* 98:289–319.

Palis, M. A. and S. M. Shende. 1995. Pumping Lemmas for the Control Language Hierarchy. *Math. Systems Theory* 28:199–213.

Thatcher, J. W. 1967. Characterizing Derivation Trees of Context-Free Grammars Through a Generalization of Finite Automata Theory. *J. Comput. System Sci.* 1:317–322.

Weir, D. J. 1992. A Geometric Hierarchy Beyond Context-Free Languages. *Theoret. Comput. Sci.* 104:235–261.

Younger, D. H. 1967. Recognition and Parsing of Context-Free Languages in Time  $n^3$ . *Information and Control* 10:189–208.

---

# Scope-marking constructions in type-logical grammar

WILLEMIJN VERMAAT

## Abstract

Scope marking constructions only appear in some languages and are characterized by the use of a specially reserved interrogative pronoun. The interrogative pronoun occurs in a main declarative clause while the sentence has an embedded interrogative clause. The whole construction is interpreted as an interrogative clause where the embedded interrogative pronoun indicates what kind of answer is being requested. In this paper, we show that we can account for these constructions along similar lines as for standard wh-question formation (Vermaat, Forthcoming). Our analysis is formulated in the multimodal variant of type-logical grammar (Moortgat, 1997).

**Keywords** GERMAN, HINDI, SCOPE-MARKING CONSTRUCTIONS, SEMANTIC UNIFORMITY, STRUCTURAL VARIATION, TYPE-LOGICAL GRAMMAR, WH-QUESTION FORMATION

## 20.1 Introduction

While languages may differ syntactically in the structural build of interrogative clauses, semantically interrogatives have a similar interpretation. In Vermaat (Forthcoming), we present a uniform account of wh-question formation by introducing a wh-type schema. The wh-type schema can be written out in the usual connectives of the base logic of multimodal categorial. To account for the syntactic differences, the grammatical reasoning system is extended with a structural module. The structural module consists of meaning preserving non-logical axioms that under feature control alter the dominance and precedence relations in a structure. With the uniform type schema proposed for

wh-phrases along with the restricted set of structural postulates, we account for the syntactic and semantic aspects of scope marking constructions in German and Hindi .

The term, *scope marking* is used by Dayal (2000). The phenomenon is sometimes referred to as *partial wh-movement* (McDaniel, 1989) or *was-constructions* (Van Riemsdijk, 1982). The following example illustrates a scope marking construction in German:

(20.57) German scope marking construction

*Was<sub>i</sub> glaubt Miro welches Bild<sub>i</sub> Picasso t<sub>i</sub> gemalt hatte?*  
 what believes Miro which picture Picasso painted had

“Which picture does Miro believe that Picasso had painted?”

Syntactically, the role of the scope marker in German and Hindi is close to the grammatical role of a relative complementizer phrase; like a complementizer it connects the subordinate clause to the matrix clause. Semantically, however, the scope marker not only connects the embedded interrogative clause to the main clause, but it also acts like a standard wh-phrase and associates to the gap in the embedded interrogative clause.

In this paper, we inspect scope marking constructions and show that scope markers fit the type schema that Vermaat (Forthcoming) proposes for wh-phrases. The syntactic and semantic aspects of scope-marking constructions follow directly. In section 20.2, due to limited space, we list the main points of the MMCG framework. For a complete overview of the deductive system, we refer to the handbook article of Moortgat (1997). In section 20.3, we present the type schema that we use to type interrogative pronouns along with an inference rule which merges a wh-phrase to the body of a question. In section 20.4, we give a small set of data which illustrates the basic construction of interrogative clauses with scope markers in German and Hindi. In section 20.5, we step-by-step construct the syntactic type of the German scope marker. We start with a syntactic analysis of the scope marker as an instance of the wh-type schema proposed for interrogative pronouns. The proposal is extended by unfolding the type for wh-questions which reveals the similarity between object wh-phrases and scope markers. The semantic representation of the scope marking construction on the basis of the Curry-Howard isomorphism shows a similar interpretation of scope marking constructions and direct questions. Additionally, in section 20.6 we present some further empirical support for our account. In section 20.7, we briefly discuss our analysis in comparison with current generative syntactic accounts. We summarize the main points of our proposal in the conclusion.

## 20.2 Theoretical background

Multimodal categorial grammar (MMCG), a version of type-logical grammar, is a lexicalized grammar system. Due to space constraints, we leave an overview of the theoretical background and only summarize the main points of the grammar framework:

- the deductive system distinguishes two parts which form the basis of the grammatical reasoning system:
  - an invariant core: the logical deductive system
  - a flexible extension: the structural module
- meaning assembly is accounted for in the deductive system via the Curry-Howard isomorphism
- structural manipulations are lexically controlled in terms of feature decorations encoded as unary operators  $\diamond$ ,  $\square$  and delimited by mode distinctions
- the structural module is severely restricted and limited to the following four instances of displacement postulates which we assume to be universally encoded

**left displacement postulates** move a feature decorated element on a left branch to a left branch one node higher:

$$\frac{\Gamma[(\diamond\Delta_1 \circ \Delta_2) \circ \Delta_3] \vdash C}{\Gamma[\diamond\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C} [Pl1] \quad \frac{\Gamma[\Delta_2 \circ (\diamond\Delta_1 \circ \Delta_3)] \vdash C}{\Gamma[\diamond\Delta_1 \circ (\Delta_2 \circ \Delta_3)] \vdash C} [Pl2]$$

**right displacement postulates** move a feature decorated element on a right branch to a right branch one node higher:

$$\frac{\Gamma[\Delta_1 \circ (\Delta_2 \circ \diamond\Delta_3)] \vdash C}{\Gamma[(\Delta_1 \circ \Delta_2) \circ \diamond\Delta_3] \vdash C} [Pr1] \quad \frac{\Gamma[(\Delta_1 \circ \diamond\Delta_3) \circ \Delta_2] \vdash C}{\Gamma(\Delta_1 \circ \Delta_2) \circ \diamond\Delta_3 \vdash C} [Pr2]$$

## 20.3 Wh-type schema

For the type assignment of interrogative pronouns, we propose a type schema  $\text{wh}(A, B, C)$  (Vermaat, Forthcoming). The wh-type schema is similar to the  $q$ -type schema,  $q(A, B, C)$ , which was proposed by Moortgat (1991) to account for in-situ binding of generalized quantified phrases. The three place type schema WH ranges over three arguments:  $B$  is the type of the body of the wh-question;  $A$  is the type of the gap hypothesis contained in the body;  $C$  is the type of the result of merging the body of the wh-question with the wh-phrase. Schematically, the following inference rule defines the merging of an arbitrary wh-phrase ( $= \Gamma$ ) with a body of a wh-question ( $= \Delta$ ) which contains a gap hypothesis ( $= \Delta[A]$ ). The result of merging the wh-phrase to the

body is a structure where the wh-phrase replaces the gap hypothesis in the structure (=  $\Delta[\Gamma]$ ).

The inference rule for the type schema along with its semantic decomposition is the following:

$$\frac{\Gamma \vdash wh : WH(A, B, C) \quad \Delta[x : A] \vdash BODY : B}{\Delta[\Gamma] \vdash wh \lambda x. BODY : C} \text{ [WH]}$$

Cross-linguistically, we recognize different instances of interrogative pronouns. In some languages the wh-phrase occurs fronted, (ex-situ, abbr. *ex*), while in other languages the wh-phrases stays in-situ (abbr. *in*). Another variation is the structural position of the gap hypothesis that the wh-phrase associates with. The gap hypothesis may reside either on a left or on a right branch. Whether the gap hypothesis occurs on a left (*l*) or on a right (*r*) branch influences the application of the structural rules and in turn effects the merging of the interrogative pronoun to the final structure.

We distinguish the different structural variants of type schemata by adding a subscript for the final position of the wh-phrase (*ex* or *in*) and a superscript for the underlying structural position of the gap hypothesis (*l* or *r*). For example,  $WH_{ex}^r(np, s, wh)$  is assigned to object wh-phrases in an SVO wh-fronting language. The wh-phrase associates to *np*-type argument hypotheses which occur on a right branch in the sentential body typed *s* and yields a wh-question of type *wh*.

## 20.4 Data

Scope marking constructions in German and Hindi are sentences with an embedded interrogative clause and a scope marker fronted (German) or cliticized to the verb (Hindi) in the main clause. The matrix verb phrase in scope marking constructions is a bridge verb, such as *glauben* (= ‘believe’) in German, which normally allows for long-distance displacement. To show the distinction between direct questions and scope marking constructions, we first present the basic formation of wh-questions in German and Hindi.

**Interrogative clauses** The standard construction of a direct question in German and Hindi are illustrated in examples 20.58 and 20.59. In German, the interrogative pronoun appears in fronted position in the main clause. In Hindi, the interrogative pronoun may appear either fronted in the matrix clause or cliticized to the verb.

(20.58) German direct question

*Welches Bild<sub>i</sub> glaubt Miro dass Picasso t<sub>i</sub> gemalt hatte?*  
 which picture believes Miro that Picasso painted had

“Which picture does Miro believe that Picasso had painted?”

(20.59) Hindi direct question (Mahajan, 2000, ex.4, p.318)

*Kis-ko<sub>i</sub> siitaa-ne socaa ki ravii-ne t<sub>i</sub> dekhaa?*  
 who Sita[erg] thought that Ravi[erg] saw

“Who did Sita think that Ravi saw?”

**Scope marking constructions** An illustration of a scope marking construction in German has been presented in example 20.57 and is reprinted below as example 20.60. The scope marker ‘*was*’ appears in fronted position while the actual interrogative pronoun ‘*welches Bild*’ is partially moved to the front of the embedded interrogative clause. Example 20.61 illustrates a scope marking construction in Hindi. The scope marker ‘*kyaa*’ may occur either fronted or preverbally in the matrix clause. The interrogative pronoun ‘*kis-se*’ occurs in-situ in the embedded interrogative clause.

(20.60) German scope marking construction

*Was<sub>i</sub> glaubt Miro welches Bild<sub>i</sub> Picasso t<sub>i</sub> gemalt hatte?*  
 what believes Miro which picture Picasso painted had

“Which picture does Miro believe that Picasso had painted?”

(20.61) Hindi scope marking construction (Mahajan, 2000, ex.1, p.317)

*(kyaa) siitaa-ne kyaa socaa ki ravii-ne kis-ko dekhaa?*  
 KYAA Sita[erg] KYAA thought that Ravi[erg] who saw

“Who did Sita think that Ravi saw?”

Although scope marking constructions differ structurally from the standard way of wh-question constructions, the overall interpretation of the scope marking construction is the same. In the coming section, we present an analysis for German scope markers which accounts for the syntactic differences and at the same time derives a similar semantic interpretation.

## 20.5 Analysis

For the analysis of interrogative clauses and scope marking constructions we assume basic categories *s* for main declarative clauses, *s'* for subordinate clauses headed by ‘*dass*’ and *s<sub>s</sub>* for declarative subordinate clauses. Wh-questions are typed as declarative clauses which are incomplete for a certain constituent, the answer to the wh-question: *s/?**A* where *A* ∈ {*np*, *iv* \ *iv*, ...}. The wh-phrase determines which kind of answer category is required<sup>1</sup>. The index *·?* is added to the binary con-

<sup>1</sup>In this paper, we only regard argument wh-phrases, but the same line of reasoning applies to adjunct wh-phrases.

nective to capture the compositional difference between predicates and arguments on a sentential level and between questions and answers on a discourse level. The most salient category for argument wh-questions is  $s/_{?}gq$  where  $gq$  is an abbreviation for generalized quantifiers:  $s/(np \setminus s)$ . To simplify the derivations, we abbreviate the categories assigned to interrogative clauses to  $wh$  ( $= s/_{?}np$ ) for main interrogative clauses and  $wh'$  ( $= s'/_{?}np$ ) for embedded interrogative clauses.

In German, main clauses appear in SVO word-order, while subordinate clauses have a head final SOV word-order. The general accepted mechanism behind this word-order difference is verb movement. Because we want to concentrate on the analysis of interrogative clauses, we fix the underlying word-order of main clauses and subordinate clauses in the lexicon. The type-assignments to syntactic objects such as transitive verbs, intransitive verbs, determiners and noun phrases are the common types assigned to lexical elements in categorial grammar fragments.

**German scope marker ‘was’** Before we discuss how the wh-scope marker finds its position at the front of the matrix clause, we first discuss the grammatical role of the scope marker. As we mentioned in the introduction, the scope marker acts like a complementizer. The complementizer ‘*dass*’ changes the type of the subordinate clause such that it can be selected by the matrix verb. In MMCG, *dass* ( $=$  ‘that’) is categorized as  $s'/s_s$ . In scope marking constructions, the matrix verb is a bridge verb which selects for an embedded declarative clause ( $\text{glaub} \vdash \text{IV}/s'$ ). The embedded clause in a scope marking construction, however, is an interrogative clause. The scope marker changes the category of the embedded interrogative ( $wh'$ ) into a category which can be selected by the bridge verb ( $s'$ ). Intuitively, the scope marker operates like a lever between the embedded interrogative clause and the matrix clause. As the following derivation illustrates, the scope marker selects the embedded interrogative and changes the type to an embedded interrogative clause ( $s'/wh'$ ). *wbpgh* is an abbreviation for an embedded interrogative clause, ‘*welches Bild Picasso gemalt hatte*’ ( $= wh'$ )

$$\frac{\text{glaub} \vdash \text{IV}/s' \quad \frac{\text{was} \vdash s'/wh' \quad \text{wbpgh} \vdash wh'}{\text{was} \circ \text{wbpgh} \vdash s'} \quad [E]}{\text{glaub} \circ (\text{was} \circ \text{wbpgh}) \vdash \text{IV}} \quad [E]$$

Structurally, however, the scope marker must appear at the front of the main clause and causes the sentence to be interpreted as a wh-question. The assignment of category  $s'/wh'$  to scope marker does not



account for the structural position of the scope marker at the front of the main clause. The scope marker must have a type-assignment which accounts for 1) the connection of the embedded interrogative to the main clause and 2) the structural position of the scope marker at the front of main clause. To account for these two characteristic, we use the *wh*-type schema to assign an appropriate category to scope marker. The scope marker is an *ex-situ* type which an associates to a gap hypothesis of category  $s'/wh'$  on a left branch. The scope marker merges to a question body of type  $s$  and yields a question of type  $wh$ . The instantiation of the type schema for the German scope marker ‘*was*’ becomes:

$$\text{was} : \text{WH}_{ex}^l(\diamond\Box(s'/wh'), s, wh)$$

On the basis of this type, the scope marking construction is derived as follows<sup>2</sup>. We only display the last steps in the derivation where ‘*wbpgh*’ is an abbreviation for an embedded interrogative clause, ‘*welches Bild Picasso gemalt hatte*’ (=  $wh'$ ).

$$\frac{\frac{(\text{glaubt} \circ \text{Miro}) \circ (\diamond\Box(s'/wh') \circ \text{wbpgh}) \vdash s}{\diamond\Box(s'/wh') \circ ((\text{glaubt} \circ \text{Miro}) \circ \text{wbpgh}) \vdash s} [Pl2]}{\vdots \text{was} \vdash \text{WH}_{ex}^l(\diamond\Box(s'/wh'), s, wh)} [\text{WH}_{ex}^l E]$$

$$\text{was} \circ ((\text{glaubt} \circ \text{Miro}) \circ \text{wbpgh}) \vdash wh$$

The derivation of the scope marker construction can be paraphrased in prose as follows. After the embedded interrogative is built up as usual, the sub formula of the scope marker ( $s'/wh'$ ) functions as an hypothesis that resides at the structural position of the relative clause marker. The hypothesized scope marker changes the type of the embedded interrogative into a embedded declarative clause ( $s'$ ). The embedded clause is selected by the main clause *glaubt Miro*. After the main clause is merged to the subordinate clause, the hypothesis is displaced to the front of the matrix clause via displacement postulate *Pl2*. When the hypothesis occurs in fronted position, the scope marker is merged to the structure replacing the its hypothesis. The complete expression becomes of type  $wh$ .

**Wh-phrases and scope markers** So far, we have concentrated on the similarity between scope markers and complementizer phrases. However, the *wh*-scope marker is semantically more similar to object

<sup>2</sup>We use an abbreviated natural deduction rule to merge the scope marker to the body of the question.

wh-phrases. Semantically, it associates to the gap hypothesis in the embedded interrogative. However, due to the occurrence of another wh-phrase, it cannot bind the hypothesized argument position directly. The gap hypothesis is already bound by the embedded wh-phrase. We show that the meaning assembly of the scope marker causes the  $\lambda$ -operator of the embedded interrogative to be bound to the scope marker. The semantic similarity between scope markers and object wh-phrases accomplishes this interpretation.

To inspect the similarity between object interrogatives and scope markers, we need to unfold the *wh*-type inside the scope marker type. We derive the following type in an unabbreviated format.

$$\frac{\text{WH}_{ex}^l(\diamond \square(s'/wh'), s, wh)}{\text{WH}_{ex}^l(\diamond \square(s'/(s'/\text{?}np)), s, (s/\text{?}np))} \text{ [unfold]}$$

Syntactically, the type for object wh-phrases is similar to the type assigned scope markers. The difference is that the object wh-phrase associates to *np* gap hypotheses and the scope marker associates to a *s'/(s'/np)* gap. Mapping the syntactic type to a semantic type, the scope marker is actually reasoning over “lifted” types:  $(A \rightarrow B) \rightarrow B$ .

$$\begin{aligned} \text{object wh-phrase 'was'} &: \text{WH}_{ex}^l(\diamond \square np, s, s/\text{?}np) \\ \text{scope marker 'was'} &: \text{WH}_{ex}^l(\diamond \square(s'/(s'/np)), s, s/\text{?}np) \end{aligned}$$

The semantic term assigned to scope markers reflects this similarity to object wh-phrases. Instead of applying a predicate  $P$  to the argument variable  $x$ , the predicate  $P$  is applied to the lifted argument variable:  $\lambda Q.(Q x)$ .

$$\begin{aligned} \text{object wh-phrase 'was'} &: \lambda Q^{et}.\lambda x^e.(Q x) \\ \text{scope marker 'was'} &: \lambda P^{((et)t)t}.\lambda x^e.(P \lambda Q^{et}.(Q x)) \end{aligned}$$

The semantic representation computed for scope marking constructions is the same for direct questions. We present the meaning assembly for the last steps of the previous derivation in figure 1. For easiness of reading, the type for wh-questions is abbreviated back to *wh* ( $= s/\text{?}np$ ).

Before merging the scope marker, the  $\lambda$ -operator,  $\lambda y$ , binding the gap variable in the embedded interrogative has been subsumed by variable  $R$  of the hypothesized scope marker. At the point where the scope marker is merged to the structure, the  $\lambda$ -operator  $\lambda Q$  pulls out the  $\lambda$ -operator of the embedded interrogative. The result is a  $\lambda$ -term where the gap variable is bound to a  $\lambda$ -operator which takes scope over the

$$\begin{array}{c}
((\mathbf{believe} (R \lambda y.((\mathbf{picture} y) \wedge ((\mathbf{painted} y) \mathbf{p})))) \mathbf{m}) \\
\frac{(\diamond \square (s'/wh') \circ ((\mathbf{glaubt} \circ \mathbf{miro}) \circ \mathbf{wbpgh}) \vdash s}{[\mathbf{WH}_{ex}^I]} \\
\vdots \text{ was} : \lambda P. \lambda x. (P \lambda Q. (Q x)) \\
\text{was} \circ ((\mathbf{glaubt} \circ \mathbf{miro}) \circ \mathbf{wbpgh}) \vdash wh \\
\rightsquigarrow_{\beta^*} \lambda x. ((\mathbf{believe} ((\mathbf{picture} x) \wedge ((\mathbf{painted} x) \mathbf{p}))) \mathbf{m})
\end{array}$$

FIGURE 1: Meaning assembly of a scope marking construction

main clause. Hence, interpreting the whole sentence as an interrogative.

The same approach applies to Hindi scope markers. The difference between German scope markers and Hindi, is the structural realization of the scope marker. The type-assignment of Hindi scope markers differs with respect to the category of the body of the question. Hindi scope markers occur optionally fronted or preverbally. For an analysis of Hindi scope marking constructions, we refer to Vermaat (Forthcoming).

## 20.6 Further empirical support

**Multiple scope marker construction** Constructions with multiple scope markers are derived by recursively binding the embedded question. Each embedded clause that intervenes the embedded interrogative and the main clause must contain another scope marker. These scope markers pass the semantic representation of the embedded argument position on to the main clause by binding the lambda abstraction of the embedded interrogative.

An example of a multiple scope marker construction is the sentence: ‘*Was glaubte Miro was Hans meint welches Bild Picasso gemalt hatte?*’. The intervening embedded clause is marked with another scope marker and thus binds the embedded argument position.

When the embedded wh-phrase does not occur in the final embedded clause, but occurs in an embedded clause higher up, the scope marker only appears in the clauses preceding the partially moved wh-phrase, for instance ‘*Was glaubte Miro welches Bild Hans meint dass Picasso gemalt hatte?*’. Due to the selectional requirements of the verb clause that follow the wh-phrase ( $\mathbf{Hans\ meint} \vdash s/s'$ ) the subordinate clause that follows the embedded interrogative are merged to the intervening clause using the relative complementizer phrase ‘*dass*’. If ‘*was*’ would be applied, the derivation fails.

**Multiple wh-phrases** Another scope marking phenomenon in Hindi falls out when scope markers are typed as proposed in this paper. In

Hindi, a scope marking construction is also used when the matrix verb selects for an embedded interrogative and the embedded interrogative contains multiple wh-phrases. For instance the sentence:

(20.62) (Lahiri, 2002, ex.73, p.522)

*Jaun kyaa puucha ki kis-ne kis-se baat kii?*  
 John what asked that who who-with talk do-[pres]  
 “Who did John ask who talked to?”

The embedded clause ‘*ki kis-ne kis-se baat kii*’ has two wh-phrases embedded and therefore becomes of type:  $(s/\text{?np})/\text{?np}$ . The matrix verb *puucha* (= ‘ask’), however, only selects for a single wh-question type. The hypothesis of the scope marker selects for a multiple wh-question type and yields a single wh-question type:  $((s'/\text{np})/(\text{wh}'/\text{np}))$ . The whole phrase is interpreted as presented in the glosses of the example; one argument is bound at the main clause level the other is bound in the embedded clause.

## 20.7 Discussion

Syntactic accounts in the syntactic framework have proposed different mechanisms to derive the right interpretation of scope marking constructions. Two leading approaches are the direct dependency approach and the indirect dependency approach. The proposed analysis type-logical grammar shows, however, that the mechanism used to derive a syntactic account of the scope marking constructions results in a semantic representation similar to direct questions. The syntactic account of the two languages differs only on the structural position of the scope marker. The semantic representations of the two scope marker indicate that the two grammatical constructions are much closer to each other than is suggested in generative syntactic literature. Furthermore, the scope marker show a clear relation to the standard account of interrogative pronouns. The type-assignment of scope marker are semantically derivable from the types assigned to object wh-phrases.

## 20.8 Conclusion

In this paper, we have shown that we can account for scope marking constructions using the proposed analysis of wh-question formation. The analysis is an example of how the deductive system of MMCG can be used to analyze natural language phenomena and indicate differences and similarities between and across languages. Because the deductive system of MMCG is completely lexicalized, any phenomena and any grammar system is determined by assigning the right type of type-assignment to individual expressions. We have shown how the restricted

set of structural postulates along with derived inference rules for the base logic realizes scope marking constructions in German. We have illustrated that the similarity between object *wh*-phrases and scope markers results in a similar semantic representation of direct questions and scope marking constructions.

## References

- Dayal, Veneeta. 2000. Scope marking: Cross-linguistic variation in indirect dependency. In U. Lutz, G. Müller, and A. von Stechow, eds., *Wh-Scope marking*, vol. 37 of *Linguistik Aktuell*, pages 157–193. Benjamins.
- Lahiri, Utpal. 2002. On the proper treatment of “expletive *wh*” in hindi. *Lingua* 112:501–540.
- Mahajan, Anoop. 2000. Towards a unified treatment of *wh*-expletives in Hindi and German. In *Wh-Scope marking*, pages 317–332. Benjamins.
- McDaniel, Dana. 1989. Partial and multiple *wh*-movement. *Natural language and Linguistic Theory* 7:565–604.
- Moortgat, Michael. 1991. Generalized quantifiers and discontinuous type constructors. In W. Sijtsma and A. van Horck, eds., *Discontinuous constituency*. De Gruyter.
- Moortgat, Michael. 1997. Categorical type logics. In J. van Benthem and A. ter Meulen, eds., *Handbook of Logic and Language*, chap. 2, pages 93–177. Amsterdam and Cambridge MA: Elsevier and MIT Press.
- Van Riemsdijk, Henk. 1982. Correspondence effects and the empty category principle. *Tilburg Papers in Language and Literature* 12.
- Vermaat, Willemijn. Forthcoming. *Logic of variation: a cross-linguistic account of wh-question formation in type-logical grammar*. Ph.D. thesis, Utrecht University, UiL-OTS.

