# $7^{\text{th}}$ International Workshop on Constraint Solving and Language Processing (CSLP'12)

**Orléans, France – 13 & 14 September 2012**

http://www.univ-orleans.fr/lifo/evenements/CSLP2012/

# Proceedings

**Denys Duchier and Yannick Parmentier (Eds.)**

**Sponsors:**

# Preface

These are the proceedings of the seventh International Workshop on Constraint Solving and Language Processing (CSLP). The first CSLP workshop took place in 2004 in Roskilde, Denmark. It opened a decade of research on the role of *constraints* in the representation and processing of language.

While constraints are widely used in many fields, including linguistics, computational linguistics, psycholinguistics, etc., their use and interpretation differs according to the research domain. In this context, the CSLP workshops aim at gathering researchers from various fields to allow them to exchange ideas about their conception of the role of constraints in language understanding and processing. Contributions to the CSLP workshops include (but are not limited to):

- Constraints in human language comprehension and production ;
- Context modelling and discourse interpretation ;
- Acquisition of constraints ;
- Constraints and learning ;
- Cross-theoretical view of the notion of constraint ;
- New advances in constraint-based linguistic theories ;
- Constraint satisfaction (CS) technologies for NLP ;
- Linguistic analysis and linguistic theories biased towards CS or constraint logic programming (CLP) ;
- Application of CS or CLP for NLP ;
- CS and CLP for other than textual or spoken languages, e.g., sign languages and biological, multimodal human-computer interaction, visual languages ;
- Probabilistic constraint-based reasoning for NLP ...

This year, 11 papers have been selected via the rigorous efforts of a Program Committee composed of renowned researchers. Each paper has been reviewed by three independent committee members. For the first time, the CSLP workshop welcomed both long and short papers, the latter being devoted to the report of ongoing work and partial results.

CSLP 2012 would not have been possible without the precious help of the program committee, the organizing committee, the local organizing committee and our sponsors. Thanks to them all.

Wishing you a fruitful reading,
yours faithfully

September 2012

Denys Duchier & Yannick Parmentier
Local Chairs
CSLP'12

# Organization

CSLP'12 is organized by the Laboratory for Fundamental Computer Science (LIFO), University of Orléans, France.

## Executive Committee

| | |
|---|---|
| Workshop Chairs: | Denys Duchier and Yannick Parmentier (Université d'Orléans, FR) |

## Steering Committee

| | |
|---|---|
| Philippe Blache | CNRS & Universit de Provence, France (FR) |
| Henning Christiansen | Roskilde University, Denmark (DK) |
| Vernica Dahl | Simon Fraser University, Canada (CA) |
| Denys Duchier | Université d'Orléans (FR) |
| Jørgen Villadsen | Technical University of Denmark (DK) |

## Program Committee

Philippe Blache, CNRS - Université de Provence, France
Adriane Boyd, Universität Tübingen, Germany
Aoife Cahill, ETS Princeton, USA
Henning Christiansen, Roskilde University, Denmark
Berthold Crysmann, CNRS - Paris 7, France
Verónica Dahl, Simon Fraser University, Canada
Helen de Hoop, Radboud University Nijmegen, Netherlands
Eric De La Clergerie, INRIA - Paris 7, France
Denys Duchier, Université d'Orléans, France
Claire Gardent, CNRS - LORIA, France
Barbara Hemforth, CNRS - Université Paris 7, France
Maria Dolores Jiménez-López, Universitat Rovira i Virgili, Spain
Laura Kallmeyer, Heinrich Heine Universität, Düsseldorf, Germany
Ruth Kempson, King's College London, UK
Stephan Kepser, Codecentric AG Düsseldorf, Germany
Patrick McCrae, Langtec Hamburg, Germany
Wolfgang Menzel, Universitt Hamburg, Germany
Detmar Meurer, Universitt Tbingen, Germany
Véronique Moriceau, Université Paris XI, France
Yannick Parmentier, Université d'Orléans, France
Jean-Philippe Prost, Université de Montpellier, France
Adam Przepiórkowski, IPIPAN, Warsaw, Poland

Christian Rétoré, Université de Bordeaux, France
Frank Richter, Universitt Tbingen, Germany
Sylvain Salvati, INRIA - Université de Bordeaux, France
Sylvain Schmitz, ENS Cachan, France
Kiril Simov, Bulgarian Academy of Sciences, Bulgaria
Jesse Tseng, CNRS - Université de Toulouse, France
Jørgen Villadsen, Technical University of Denmark

## Sponsoring Institutions

Laboratoire d'Informatique Fondamentale d'Orléans.
Université d'Orléans.
Conseil Régional du Centre.
Conseil Général du Loiret.
Mairie d'Orléans.

## Invited Speakers

**Ruth Kempson** King's College London

**Stergios Chatzikyriakidis** Royal Holloway University of London and Open University of Cyprus

*Grammars as Mechanisms for Real-Time Tree-Growth: Explaining Clitic Pronouns*

**Abstract:** In this talk, we argue for a shift of perspective into defining grammars as mechanisms for incremental growth of interpretation reflecting the time-line of processing (Dynamic Syntax: Kempson et al. 2001, Cann et al. 2005, Chatzikyriakidis & Kempson 2011). The core syntactic notion of this framework is that of monotonic context-relative tree growth (following Blackburn & Meyer-Viol 1994), with both content and structural parameters of underspecification and update. Our case study is the puzzle of clitic pronoun clusters of Modern Greek dialects, which illustrate both the low level morphological idiosyncracy of such clusterings, and yet the broad cross-linguistic constraints to which they are subject: these dialects display variants of the so-called Person Case Constraint, a constraint whose generality continues to provide a major challenge for current theoretical frameworks (Adger & Harbour 2007, Heap 2005, among others). We show first how the limits on variation displayed in such clusters are explicable in terms of a constraint debarring more than one underspecified tree relation of a type at a time, a constraint only expressible in a dynamical grammar framework; and we give an analysis of Greek dialectal variation in these terms. Then we explore the consequences of this theoretical perspective, viz. the domain-generality of the system of growth underpinning natural-language syntax; and we will suggest that metrical ambiguities and metrical dissonance displayed in music (Vazan & Schober 2004, London 2012) are subject to the same restriction on real-time structural processing.

# Table of contents

## Long papers



## Short papers

# An Account of Natural Language Coordination in Type Theory with Coercive Subtyping*

Stergios Chatzikyriakidis[1] and Zhaohui Luo[2]

[1] Dept of Computer Science, Royal Holloway, Univ of London
Egham, Surrey TW20 0EX, U.K; Open University of Cyprus
stergios.chatzikyriakidis@cs.rhul.ac.uk,
[2] Dept of Computer Science, Royal Holloway, Univ of London
Egham, Surrey TW20 0EX, U.K; Open University of Cyprus
zhaohui@cs.rhul.ac.uk

**Abstract.** We discuss the semantics of NL coordination in modern type theories (MTTs) with coercive subtyping. The issue of conjoinable types is handled by means of a type universe of linguistic types. We discuss quantifier coordination, arguing that they should be allowed in principle and that the semantic infelicity of some cases of quantifier coordination is due to the incompatible semantics of the relevant quantifiers. Non-Boolean collective readings of conjunction are also discussed and, in particular, treated as involving the vectors of type $Vec(A, n)$, an inductive family of types in an MTT. Lastly, the interaction between coordination and copredication is briefly discussed, showing that the proposed account of coordination and that of copredication by means of dot-types combine consistently as expected.

## 1 Introduction

The literature on NL coordination dates back to [22] and a number of proposals have been put forth within the Montagovian tradition since then. However, a number of central issues as regards NL coordination have not been clarified yet. In this paper we depart from single-sorted versions of type theory found in Montague's work (as well as in most of the subsequent work within the same tradition) and employ a many-sorted modern type theory (MTT)[3], as proposed and studied for NL semantics in [30, 17, 18], to deal with two central issues in NL coordination. These issues concern the notion of conjoinable types, in effect the question of which NL elements can be coordinated, and non-Boolean conjunction, where a collective rather than the expected Boolean distributive reading of *and* arises. The difference between collective and distributive readings is exemplified

* This work is supported by the research grant F/07-537/AJ of the Leverhulme Trust in the U.K.

[3] Examples of modern type theories include Martin-Löf's type theory [21, 26], the Unifying Theory of dependent Types (UTT) [15] and the type theory implemented in the Coq proof assistant (pCIC) [7].

in the examples below, where the same conjoined NP is interpreted distributively in (1) but collectively in (2):

(1) John and Mary came to the Party.

(2) John and Mary met at the Party.

We shall investigate how collective readings can be interpreted by means of the inductive family of types of vectors in an MTT.

We further discuss the interaction between dot-types for coordinated NPs. Dot-types have been proposed by Pustejovsky [28, 29] for lexical interpretations of inherently polysemous words in phenomena such as co-predication (see, for example, [2]).[4] For example, *book* according to [28] can be represented with the dot-type PHY • INFO, a type whose objects have both a physical and an informational aspect. Dot-types have been formally introduced into MTTs with coercive subtyping [17, 18] and a computational implementation of this account in Plastic[5] has also been done [35]. What we want to look at in this paper is the interaction between these types and coordination, i.e. examples of the following sort:

(3) The book and my lunch were sent by mistake to someone else.

(4) John picked up the newspaper and the book from the floor.

Given that the dot-types of the coordinated phrases are different and assuming that the NL coordination operate on the same types, we will have to explain how coordination is possible in these cases. The problem that arises in examples like (3) and (4) is that the individual NPs of the conjunction (e.g. *the book* and *my lunch* in (3) have different types (PHY • INFO for book and EVENT • PHY for lunch). The challenge is to account for the possibility of coordination in these cases by, at the same time, retaining the assumption that coordination operates on elements of the same type. As we shall see, the coercive subtyping mechanism actually allows us to combine the proposed typing for NL coordinations and the account with dot-types in a rather straightforward way.

## 2 Type Theory with Coercive Subtyping

In this paper, we employ modern type theories (MTTs) as the language for formal semantics. A brief introduction to the relevant features of MTTs are briefly given here.

An MTT has a number of differences when compared to Church's simple type theory as employed in Montague semantics [6, 23]. One of the most important

[4] See also [3] for a critique of the flaws in the various formalizations of dot-types in their original formulation as well as in much of the later work based on that.

[5] Plastic [5] is a *proof assistant*, an implementation of the modern types theory UTT [15] on the computer for formalised proof development. In the context of linguistic semantics, type theory based proof assistants such as Agda [1], Coq [7] and Plastic can be used to formalise and reason about the formal semantics based on MTTs.

differences between an MTT and the simple type theory, is that the former can be regarded as *many-sorted* while the latter single-sorted. MTTs use many types to interpret Common Nouns (CN) such as *man* and *table*, while single-sorted type theories use only one type ($e$) for the type of all entities (and another type $t$ for logical truth values), with CNs being interpreted as predicates of type $e \to t$.

In Montague semantics, an Intransitive Verb (IV) is interpreted as a function from entities to truth values ($e \to t$), a type which is shared with CNs and intersective adjectives, and a quantified NP as of the type from properties to truth values ($(e \to t) \to t$).

In an MTT, types ('sorts') are used to interpret the domains to be represented. Some of them are:

- the *propositional types* (or logical propositions),
- the *inductive types* such as the type of natural numbers and $\Sigma$-types of dependent pairs,
- the inductive families of types such as the types $Vec(A, n)$ of vectors (or lists) of length $n$, and
- other more advanced type constructions such as *type universes*.

For example, within such a many-sorted logical system CNs are not interpreted as predicates as in the Montagovian tradition but rather as Types. Theoretical motivation behind such a proposal has been provided by the second author based on the notion of identity criteria that CNs have according to [9]. (See [19] for the exact details of this proposal.) Then given the interpretation of CNs as types, adjectives are interpreted as a predicate over the type interpreting the domain of the adjective. For example, the adjective *handsome* is interpreted as $[\![handsome]\!] : [\![man]\!] \to Prop$, with $Prop$ being the type of logical propositions.[6] Modified CNs are then interpreted as $\Sigma$-types, the types that intuitively represent subset types but contain explicit proof objects.[7]

One of the important features of MTTs is the use of dependent types. Two examples of basic constructors for dependent types are $\Pi$ and $\Sigma$. The $\Pi$-type corresponds to universal quantification in the dependent case and implication in the non-dependent case. In more detail, when $A$ is a type and $P$ is a predicate over $A$, $\Pi x{:}A.P(x)$ is the dependent function type that, in the embedded logic, stands for the universally quantified proposition $\forall x{:}A.P(x)$. A $\Pi$-type degenerates to the function type $A \to B$ in the non-dependent case. In the case of $\Sigma$, if $A$ is a type and $B$ is an $A$-indexed family of types, then $\Sigma(A, B)$, or sometimes written as $\Sigma x{:}A.B(x)$, is a type, consisting of pairs $(a, b)$ such that $a$ is of type

[6] MTTs have consistent internal logics based on the propositions-as-types principle [8, 14]. For example, in a predicative type theory such as Martin-Löf's type theory, the logical proposition $A\&B$ corresponds to the product type $A \times B$ (a special case of $\Sigma$-type – see below) and a pair of a proof of $A$ and a proof of $B$ corresponds to an object of the product type. In an impredicative types theory such as UTT, logical propositions are similarly constructed as types but, furthermore, there is the type $Prop$ – a totality of logical propositions.

[7] See [30, 17] for more details on this.

$A$ and $b$ is of type $B(a)$. When $B(x)$ is a constant type (i.e., always the same type no matter what $x$ is), the $\Sigma$-type degenerates into product type $A \times B$ of non-dependent pairs. $\Sigma$-types (and product types) are associated projection operations $\pi_1$ and $\pi_2$ so that $\pi_1(a,b) = a$ and $\pi_2(a,b) = b$, for every $(a,b)$ of type $\Sigma(A,B)$ or $A \times B$.

Coercive subtyping is an adequate subtyping mechanism for MTTs [16, 20] and, in particular, it avoids a problem associated with the ordinary notion of subtyping (subsumptive subtyping), namely violation of canonicity [17].[8] Basically, coercive subtyping is an an abbreviation mechanism: $A$ is a (proper) subtype of $B$ ($A < B$) if there is a unique implicit coercion $c$ from type $A$ to type $B$ and, if so, an object $a$ of type $A$ can be used in any context $\mathfrak{C}_B[\_]$ that expects an object of type $B$: $\mathfrak{C}_B[a]$ is legal (well-typed) and equal to $\mathfrak{C}_B[c(a)]$. For instance, one may introduce $[\![man]\!] < [\![human]\!]$. Then, if we assume that $[\![John]\!] : [\![man]\!]$ and $[\![shout]\!] : [\![human]\!] \to Prop$, the interpretation (6) of (5) is well-typed, thanks to the coercive subtyping relation between $[\![man]\!]$ and $[\![human]\!]$:

(5) John shouts.

(6) $[\![shout]\!]([\![John]\!])$

Ending our discussion on the preliminaries of TTCS, we mention one further more advanced feature of the theory, that of universes. A universe is a collection of (the names of) types into a type [21]. This can be seen as a reflection principle where the universe basically reflects the types whose names are its objects. Universes are extremely useful in accounts of lexical semantics using MTTs. Specifically, universes can help semantic representations. To give an example, one may use the universe CN : $Type$ of all common noun interpretations and, for each type $A$ that interprets a common noun, there is a name $\overline{A}$ in CN. For example,

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

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

Summarizing, we can say that the use of TTCS in interpreting NL semantics has given a number of interesting results and insights. These include an increased type granularity when compared to Montague Semantics given its type richness as well as an adequate subtyping mechanism.[9] Furthermore the interpretation of CNs as Types rather than predicates seems to be closer to the idea according to which the distinguishing feature of CNs, when compared to other parts of speech, is that only the former have what Geach called, criteria of identity [9]. The work presented in [19] provides strong arguments for supporting the non-predicate view on CNs based on Geach's identity criteria. The successful formalization [17] and subsequent implementation in Plastic [35] of *dot.types* is another achievement of this line of research given that no proper formalization

[8] See [17] for the notion of canonicity.

[9] This subtyping mechanism is however in line with canonicity and as such computationally more attractive [17].

of *dot.types* existed up to that point. The use of universes has been also proven fruitful in looking at alternative ways for defining the types for quantifiers and adverbs among others. Lastly, parts of the various proposals made in the aforementioned papers have been tested using the Coq interactive theorem prover. Some first results can be seen in [18] as well as in this paper. Current work of the first author concentrates on the use of Coq to prove valid NL theorems[10] as well as building universes relevant to NL semantics (e.g. CN, LType) in Plastic.[11]

## 3 Conjoinable Types

The issue of defining which NL types are conjoinable is of very high importance to all accounts of coordination proposed so far. Most of the accounts that have been proposed in the Montagovian tradition argue that conjoinable types are either of type $t$ or of a function type that ends with $t$. The formalization might be different in individual cases but the core of the proposal is pretty much the same. The definition as given by Winter [34] is given below (using the term t-reducible):[12]

(7) $\tau$ is a t-reducible type iff $\tau = t$ or $\tau = \tau_1\tau_2$, where $\tau_1$ is any type and $\tau_2$ is a t-reducible type.

Such type of formulation allows coordination of categories ending in type $t$ only, with type $e$ conjunction not being possible. Thus, in these accounts proper name coordination is either assumed to involve type-lifting to quantifier type or proper names are assumed to be quantifiers in all cases. However, Partee & Rooth [27] propose a definition of $e$ conjoinable types to deal with collective reading cases. Similar proposals have been made by Hoeksema [11]. Of course, an inductive definition of an e-conjoinable type does not make much sense given that at least in standard Montagovian semantics, the only $e$ conjoinable types are the type of individual concepts, of type $s \rightarrow e$, i.e the type from indices to individuals, so the definition basically covers just one case.

Moving away from the simple type theory in Montague Grammar and using many-sorted MTTs, the first question to ask ourselves is how conjoinable categories can be defined. Well, the first question to be asked is which linguistic types can be conjoined? Surprisingly (or not) it seems that all linguistic categories can be conjoined. We first note the obvious cases of sentence and predicate coordination (8 and 9) to CN coordination (10):

(8) John walks and Mary talks.

(9) John walks and talks.

[10] An example of this type is the following: if John and Mary met then John met Mary. Such theorems can be proved if the correct semantics are given in each case.

[11] This is not possible in Coq.

[12] We follow the notation as this is given in [34]. As such, $\tau_1\tau_2$ should be taken to mean $\tau_1 \rightarrow \tau_2$.

(10) A friend and colleague came.

Then, quantified NP coordination (11), quantifier coordination (12) and proper name (PN) coordination are possible (13):

(11) Every student and every professor arrived.

(12) Some but not all students got an A.

(13) John and Mary went to Italy.

Adverb conjunction(14), preposition conjunction(15), PP conjunction (16)

(14) I watered the plant in my bedroom but it still died slowly and agonizingly.

(15) I can do with or without you.

(16) The book is on the table and next to the chair.

Lastly, coordination of subordinate connectives is also possible (17):

(17) When and if he comes, you can ask him.

### 3.1 Universe of Linguistic Types

In this section we will propose a way to handle the flexibility NL coordination exhibits by using a MTT. The key idea behind the account we are going to propose is the notion of a universe.

A universe, as we have already mentioned at the end of §2, is a collection of (the names of) types into a type [21]. In the case of coordination, the universe CN of the types that we have used to interpret common nouns is far too small to capture the generality of the phenomenon. Given that all linguistic categories can be coordinated, the universe we need, has to be far more general than CN.

The idea is to introduce a type universe $LType$ of *Linguistic Types*. Intuitively, $LType$ contains (the names of) all types that are employed in linguistic semantics. Of course, in doing so, we will have to specifically say what we consider a linguistic type to be. Even though a thorough discussion of meticulously constructing the universe of linguistic types is out of the scope of this paper, we shall indicate positively what types may have names in the universe $LType$.[13] Figure 1 contain some of the introduction rules for $LType$, where we have used the so-called Russell-style formulation of a universe to omit the names of its objects. The informal explanations of the rules in Figure 1 are given below.

– The type $Prop$ of logical propositions is a linguistic type. (It is of type $PType$[14] by the first rule and hence of type $LType$ by the last rule.)

[13] We leave this most thorough and complete discussion of the universe $LType$ for future work.

[14] $Ptype$ can be thought of as the universe of predicates. It is an intermediate universe used to build $LType$.

$$\frac{}{PType : Type} \qquad \frac{}{Prop : PType} \qquad \frac{A : LType \quad P(x) : PType\ [x{:}A]}{\Pi x{:}A.P(x) : PType}$$

$$\frac{}{LType : Type} \qquad \frac{}{\text{CN} : LType} \qquad \frac{A : \text{CN}}{A : LType} \qquad \frac{A : PType}{A : LType}$$

**Fig. 1.** Some (not all) introduction rules for $LType$.

- If $A$ are linguistic types and $P$ is an $A$-index family of types in $PType$, so is the $\Pi$-type $\Pi x{:}A.P(x)$. In particular, in the non-dependent case, if $A_i$ are linguistic types, so is the arrow type $A_1 \to ... \to A_n \to Prop$. (It is of type $PType$ by repeated uses of the third rule and hence of type $LType$ by the last rule.)
- The universe $CN$ (of types that interpret common nouns) is an object of type $LType$.
- If $A$ interprets a common noun, then $A$ is a linguistic type in $LType$. For example, the $\Sigma$-types that interpret modified CNs are in $LType$.

Other example types in $LType$ include the type of VP adverbs and that of quantifiers, shown in the following examples:

(18) $\Pi A : \text{CN}.\ (A \to Prop) \to (A \to Prop)$

(19) $\Pi A : \text{CN}.\ (A \to Prop) \to Prop$

Please note that we have only listed some of the introduction rules for $LType$. For example we have not yet included the type for PP-modifiers. At the moment, we shall leave the universe $LType$ to be *open* in the sense that we may introduce new types into it in the future.[15]

Having described the universe of linguistic types, we can now use it to describe the type of coordinators: every (binary) coordinator is of the following type:

(20) $\Pi A : LType.\ A \to A \to A$

For instance, the coordinator *and* is of the above type.

To give an example of how this type works, let us imagine three cases of coordination: PN coordination (John and George), propositional coordination (John runs and Mary drives) and VP coordination (John cycles and drives). In the first case, *John* and *George* are of type $[\![Man]\!]$, so the $A$ in this case is of type $[\![Man]\!]$ which is in $LType$ given that it is of type $CN$. In the case of propositional coordination, our $A$ is of type $Prop$, which being a $PType$ is also an $LType$. In the third case our $A$ is of type $[\![Man]\!] \to Prop$ which is also in $LType$. Similar considerations apply to all the cases from (8) to (17). Thus, this type captures the flexibility associated with coordination.[16] It is not difficult to

[15] Formally, openness of a universe would imply that we do not impose an elimination rule for it. We omit the technical details here.

[16] Of course, there are cases discussed in the literature where coordination of different categories seems to be possible. One such example is discussed in [24], where an

see that all examples of coordination from (8) to (17) are predicted via the type given for coordination above.[17] However, what we need to discuss is examples where the rule proposed in (20) might seem to overgenerate or departs from the standard assumptions as these are made in the formal semantic literature.

### 3.2 Quantifier Coordination

The type for coordination we have proposed might be argued to overgenerate for cases involving coordination of two quantifiers like the ones shown below:

(21) # Some and every man came

(22) # No and some boy read

The above sentences seem to be generated via the rule we have proposed for coordination. Note, that this problem applies to all coordination accounts proposed. Given that quantifiers involve a function type ending in $t$, they should be conjoinable according to the accounts proposed in the Montagovian literature. No explicit discussion has been made of how cases like these are disallowed, so it would be good to see in more detail what is going on in these cases.

The basic problem is that some quantifiers seem to be able to be coordinated and some others do not. Between the cases of quantifiers that cannot be coordinated with a coordinator there are cases where adding a modal adverb between the coordinator and the second conjunct make a difference in acceptability. For example, adding the modal adverb *possibly* in (21) but not in (24) makes the sentence semantically felicitous:

(23) Some and possibly every man came

(24) # No and possibly some boy read

For the rest of the cases, whether such sentences will be semantically felicitous depends on the type of coordination in each case (cf. the two examples below):

(25) # One and two of my students came to the party.

(26) One or two of my students came to the party.

So, it seems that in principle, we should allow coordination of quantifiers, since there are clear cases where this is possible. However, allowing coordination of quantifiers to be in principle possible, we will have to explain semantically infelicitous cases like (25). A way that can help us rule out a number of infelicitous semantic cases is to look at the semantics of the individual quantifiers in

adjective is coordinated with a NP: *John is either stupid or a liar*. We will not pursue an account here but we could note that an account in a similar vein to the one proposed by [25] where coordination even in this case operates on like and not on unlike categories is possible.

[17] All the examples have been checked using the Coq theorem prover [7]. The code can be found in the Appendix.

combination with the coordinator in each case. Let us take the example of the following NP:

(27) # Some and no men arrived.

The quantifiers in the above example can be coordinated via the rule we have proposed. However, the semantics we get for the coordinated NP *some and no man* are the following, in effect a contradiction:

(28) $\exists x : [\![man]\!] .P(x) \wedge \sim \exists x : [\![man]\!] .P(x)$

We can quite plausibly assume that the contradictory semantics is the reason the conjunction is infelicitous in (31), especially when uttered out of the blue without any context. Now imagine the following situation: someone is lying and has stated that *no men arrived* on one occasion and that *some men arrived* on another. Then, the hearer might spot this contradiction and utter the following *'some and no men arrived?'*. In this case, *some and no men* is perfectly felicitous.[18] Disjunction of the same quantifiers is possible without the aid of some special context. Examples of this quantifier combination are quite frequently found in NL:

(29) People with some or no academic training.

(30) This license may grant the customer the ability to configure some or no parts of the software themselves.

The semantics of *some or no x* in contrast to the semantics of *some and no x* do not give rise to a contradiction. To the contrary, they are always true under any interpretation. The example below depicts the semantics of *some or no men*:[19]

(31) $\exists x : [\![man]\!] \, P(x) \vee \sim \exists x : [\![man]\!] .P(x)$

Further examples of quantifiers that do not need a special context are *some but not all*, *more than three and less than five*. It might then be the case, that quantifier combinations that are always false need a special context in order to be felicitous while quantifier combinations that do not fall into this category do not. Of course, there are obvious counterexamples to such a proposal, for example cases like *some and all* or *most and all*, which are of course infelicitous in the absence of any special context contrary to what we expect in case what we say is true. However, quantifiers like *some* and *most* in NL carry a quantity implicature (see e.g. [13], [12] and [10] among others). The idea is that a speaker uttering *some* and not the stronger *all*, does that because he believes that substitution for the stronger value cannot be done salva veritate. For if the latter was true, he would have uttered the stronger *all*. A quantifier combination like *some and all* cancels out this implicature, so this might be the reason for the infelicitousness of

[18] The same kind of example can be devised for cases like (23).

[19] This is the case assuming a logical interpretation of *some*. If the quantity implicature is taken into consideration, the quantifier combination is not always true.

this quantifier combination when uttered out of context. The same can be argued for the case of *most and all*. The issue requires more careful examination in order to see whether what we have argued is true or not. In particular, one has to check whether cases of quantifier combinations that are always false need the aid of some special context in order to be felicitous. Then, cases where infelicitousness arises unexpectedly must be shown to arise from other independent factors (like the quantity implicature for example). We believe that what we have proposed can produce a fruitful line of research as regards quantifier coordination but at least for this paper, we will not examine the issue any further. What is rather uncontroversial, no matter the assumptions we make as regards the interplay of quantifier coordination and the use of context, is that we need a rule for coordination that will in principle allow quantifier coordination. The rule we have proposed in (24) suffices for this reason.

Recapitulating, we propose a general rule for coordination which extends over a universe that contains all linguistic types, the universe *LType*. This rule is general enough to allow all types of coordination we find in NL. The rule might seem to overgenerate in the case of quantifier coordination, but as we have seen, in principle quantifier coordination should be allowed. The infelicitous cases (when uttered out of the blue) are attributed to the semantics of the individual quantifiers under the coordinator involved in each case.

### 3.3 Non-Boolean Conjunction

The first thing we have to see is what is the prediction our typing rule proposed for coordination makes for these cases. But before doing this, we first have to discuss the typing of predicates like *meet* in their collective interpretation. Such predicates can be seen as one place predicates that take a plural argument and return a logical proposition (something in *Prop* in our case), an assumption already made in a number of the accounts within the Montagovian tradition (e.g. [34, 33]). The plausible question is how plural arguments are going to be represented.

An interesting account of plurality within an MTT is presented by [4], where Martin-Löf's type theory is used for linguistic semantics. In this account, plural count nouns are interpreted using the type $List(A)$. Such an account is shown to give a uniform treatment of both singular and plural anaphora, being compatible with the classical type-theoretic treatment of anaphora, as this is given by [32].[20] In MLTT, $List(A)$ corresponds to the set of lists of elements of a set $A$. We will keep the intuition regarding the need to represent lists of objects but instead of using the inductive type $List(A)$, we will use the inductive family of types *Vec(A,n)*. $Vec(A, n)$ and $List(A)$ are very much alike, with the difference being mainly that $Vec(A, n)$ involves an additional argument $n$ of type *Nat*, which

[20] Another interesting account of plurals is given by [31] using Girard's system F. It is shown that the basic properties of plurals can be effectively accounted for by using a second-order system like Girard's system F.

counts the number of the elements in a list (that is why they are called vectors):[21]

(32) $Vec : (A : Type)(n : Nat)Type$

Now, collective predicates can be given types in a more appropriate way. For example, the collective predicate *meet* can be given the following type:

(33) $\Pi n{:}Nat.\ Vec([\![human]\!], n+2) \rightarrow Prop$

Please note that, as $n \geq 0$, an object of type $Vec([\![human]\!], n+2)$ has length of at least 2 or longer – this means that *meet* can only be applied to at least two people, but not less. Such more exact requirements are captured in typing by means of the inductive families like $Vec(A, n)$.

Now, let us explain how to interpret sentences like (34):

(34) John and Mary met.

The above typing of *meet* assumes that the typing for humans can distinguish the number for the plural cases. In other words, this assumes that, collective *and* should be given the following type:

(35) $\Pi A : CN.\ \Pi n, m{:}Nat.\ Vec(A, n) \rightarrow Vec(A, m) \rightarrow Vec(A, n+m)$

Therefore, for example, assuming that $J : Vec([\![human]\!], 1)$ and $M : Vec([\![human]\!], 1)$, then $J\ and\ M$ is of type $Vec([\![human]\!], 2)$. In order to type phrases like $John\ and\ Mary$, we need to introduce the following coercions, for every type $A$:

$$A <_c Vec(A, 1)$$

where the coercion $c$ maps any $a$ to $[a]$, the vector with only element $a$. With $John : [\![man]\!] < [\![human]\!]$ and $Mary : [\![woman]\!] < [\![human]\!]$, we have that $John\ and\ Mary$ is interpreted as of type $Vec([\![human]\!], 2)$ and therefore, the above sentence (34) gets interpreted as intended.

However, we are not done yet with collective predication, given that we have not yet discussed cases involving quantifiers. Such a case is shown below:

(36) Three men and five women met.

Given the type associated with quantifiers, the rule for collective coordination as this was given in (35) will not work. What we propose is to use a unit type which will encode both typings for collective *and*. There is no space here to

[21] See Chapter 9 of [15] for the formal definition of $Vec(A, n)$. We omit the formal details here. Furthermore, and as suggested by an anonymous reviewer, one might consider using finite types (see for example Appendix B of [19]) instead of vector type. This seems to be a good suggestion and will be considered for future refinements of the proposals found in this paper.

explain the notion of a unit type but let us say that such a type will encode both the typings in (37) via the coercions in (38):[22]

(37) $[\![and_1]\!] : \Pi A : CN.\ \Pi n, m{:}Nat.\ Vec(A, n) \rightarrow Vec(A, m) \rightarrow Vec(A, n + m)$
$[\![and_2]\!] : \Pi A : \text{CN}.\ ((Vec(A, n) \rightarrow Prop) \rightarrow Prop) \rightarrow ((Vec(A, m) \rightarrow Prop) \rightarrow Prop) \rightarrow ((Vec(A, n + m) \rightarrow Prop) \rightarrow Prop)$

(38) $c_1(and) = [\![and_1]\!]$ and $c_2(and) = [\![and_2]\!]$

Now, given $[\![man]\!], [\![woman]\!] < [\![human]\!]$, we have:

(39) $and([\![three\ men]\!])([\![five\ women]\!]) : ((Vec([\![human]\!], 3 + 5) \rightarrow Prop) \rightarrow Prop)$

*Meet* is applied to the above type and the sentence is well-typed.

*Remark 1.* Another way of dealing with collective predication is to assume a type for collective and that extends over the universe $LType$ rather than $CN$. This rule will produce the following typing in case of quantifier coordination $Vec(((A \rightarrow Prop) \rightarrow Prop), n + m)$. However, given such a type since *meet* will not be able to apply assuming the type in (40). The solution in this case will be to have a unit type for *meet*, where one of the two types of the unit type is type lifted and turned into a functor taking a $GQ$ as an argument. We leave the discussion on the consequences and formalization of such proposal open due to space limitations. □

One further welcoming extension of the account proposed is a straightforward explanation of the way the reciprocal *each other* functions in English. Verbs like *meet* are reciprocal predicates in the sense that they do not need an overt reciprocal to give rise to a reciprocal reading (basically what we have been calling the collective reading so far). For non-reciprocal predicates, there is the possibility of getting these readings via the use of *each other*. The idea is that each other in English turns a transitive predicate into an intransitive one whose sole argument is a vector $A : CN$ with $n$ of at least 2:[23]

(40) $[\![each_o ther]\!] : \Pi A : \text{CN}.(A \rightarrow A \rightarrow Prop) \rightarrow (Vec(A, n + 2)Prop)$

[22] Another possibility will be to assume that only the type relevant for the collective interpretation of GQs is needed. In this case, proper names can be interpreted collectively only in their GQ guise.

[23] If we want to generalize the rule to verbs with arity of more than two, we can use the following: $[\![each_o ther]\!] : \Pi A : \text{CN}.(A \rightarrow A \rightarrow A^* \rightarrow Prop) \rightarrow (Vec(A, 2) \rightarrow A^* Prop)$, where $A^*$ stands for 0 or more $A$ arguments.

## 4 Interaction of Coordination and Copredication

Dot-types have been successfully formalized in MTTs with coercive subtyping [17, 18], and an implementation of them in the proof assistant Plastic also exists [35]. We first summarize the account proposed for dot-types and then proceed and discuss the interaction between dot-types and coordination. We will use *book* as our prototypical example in presenting the account.

Book is assumed to be a dot-type having both a physical and an informational aspect. The type-theoretic formalization of this intuition proceeds as follows. Let PHY and INFO be the types of physical objects and informational objects, respectively. One may consider the dot-type PHY•INFO as the type of the objects with both physical and informational aspects. A *dot-type* is then a subtype of its constituent types: PHY • INFO < PHY and PHY • INFO < INFO. A book may be considered as having both physical and informational aspects, reflected as:

$$(*) \qquad [\![book]\!] < \text{PHY} \bullet \text{INFO}.$$

Now, consider the following sentence:

(41) John picked up and mastered the book.

We assume the following typing for *pick-up* and *master* respectively:

$$[\![pick\ up]\!] : [\![human]\!] \to \text{PHY} \to Prop$$
$$[\![master]\!] : [\![human]\!] \to \text{INFO} \to Prop$$

Because of the above subtyping relationship $(*)$ (and contravariance of subtyping for the function types), we have

$$\begin{aligned} [\![pick\ up]\!] \ &: \ [\![human]\!] \to \text{PHY} \to Prop \\ &< \ [\![human]\!] \to \text{PHY} \bullet \text{INFO} \to Prop \\ &< \ [\![human]\!] \to [\![book]\!] \to Prop \end{aligned}$$

$$\begin{aligned} [\![master]\!] \ &: \ [\![human]\!] \to \text{INFO} \to Prop \\ &< \ [\![human]\!] \to \text{PHY} \bullet \text{INFO} \to Prop \\ &< \ [\![human]\!] \to [\![book]\!] \to Prop \end{aligned}$$

Therefore, $[\![pick\ up]\!]$ and $[\![master]\!]$ can both be used in a context where terms of type $[\![human]\!] \to [\![book]\!] \to Prop$ are required and the interpretation of the sentence (41) can proceed as intended.

The first case of interaction has already been introduced and involves examples like (41). It is to see how this is going to be predicted given what we have said.[24]

[24] See [17, 18] for an account of this.

The next step is to take a look at examples where two words with dot-types are coordinated. Such an example is shown below:

(42) The book and my lunch were sent by mistake to someone else.

In the above example we have two dot-types involved, PHY • INFO and PHY • EVENT, representing the types for *book* and *lunch* respectively. Let us see whether the rule for coordination we have along with the treatment of dot-types will give us the correct results.

We need to coordinate the two NPs:

$$[\![the\ book]\!] : [\![book]\!] \quad \text{and} \quad [\![my\ lunch]\!] : [\![lunch]\!] .$$

Furthermore, the passive *send* is of the following type:

$$[\![send_{pass}]\!] : \text{HUMAN} \rightarrow \text{PHY} \rightarrow Prop.$$

Now, because

$$[\![book]\!] < \text{PHY} \bullet \text{INFO} < \text{PHY}$$
$$[\![lunch]\!] < \text{PHY} \bullet \text{EVENT} < \text{PHY}$$

the above sentence (42) can be interpreted as intended. In other words, the coercive subtyping mechanism interacts with that for coordination correctly.

## 5 Conclusions

In this paper we presented an account of NL coordination using Type Theory with Coercive Subtyping. The issue of conjoinable types was taken care of by proposing an inductive type for coordination which extends over the universe of Linguistic Types, called *LType*. This type has been shown to be sufficient to explain the flexibility of NL coordination. We argued that a rule for NL coordination should in principle allow quantifier coordination and showed that the infelicitous quantifier combination cases are due to the inherent semantics of the quantifier combination under the coordinator in each case, along with general pragmatic implicatures associated with quantifiers (e.g. the quantity implicature for quantifiers *some* and *most*). Non-Boolean conjunction was accounted for, assuming that collective predicates take one vector argument representing plurality. A second rule for collective *and* was proposed which takes two vector arguments of $n$ and $m$ length and produces a vector type of length $n+m$. Lastly, the interaction of *dot.types* with coordination was briefly discussed. It was shown that the coordination account proposed in combination with the co-predication account as this was given in [18] gives the correct predictions.

## References

1. The Agda proof assistant (version 2). Available from the web page: `http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php` (2008)

2. Asher, N.: Lexical Meaning in Context: a Web of Words. Cambridge University Press (2012)
3. Bassac, C., M.B., Retor, C.: Towards a type-theoretical account of lexical semantics. Journal of Logic Language and Information 19, 229–245 (2010)
4. Boldini, P.: The reference of mass terms from a type-theoretical point of view. Paper from the 4th International Workshop on Computational Semantics (2001)
5. Callaghan, P., Luo, Z.: An implementation of LF with coercive subtyping and universes. Journal of Automated Reasoning 27(1), 3–27 (2001)
6. Church, A.: A formulation of the simple theory of types. J. Symbolic Logic 5(1) (1940)
7. The Coq Development Team: The Coq Proof Assistant Reference Manual (Version 8.3), INRIA (2010)
8. Curry, H., Feys, R.: Combinatory Logic, vol. 1. North Holland (1958)
9. Geach, P.: Reference and Generality: An examination of some Medieval and Modern Theories. Cornell University Press (1962)
10. Geurts, B.: Quantity Implicatures. Cambridge University Press (2010)
11. Hoeksema, J.: The semantics of non-boolean and. Journal of Semantics 6, 19–40 (1998)
12. Horn, L.: The border wars: a neo-gricean perspective. In: von Heusinger, K., Turner, K. (eds.) Where semantics meets pragmatics,s, pp. 21–46. Amsterdam: Elsevier. (2006)
13. Horn, L., R.: A Natural History of Negation. University of Chicago Press (1989)
14. Howard, W.A.: The formulae-as-types notion of construction. In: Hindley, J., Seldin, J. (eds.) To H. B. Curry: Essays on Combinatory Logic. Academic Press (1980)
15. Luo, Z.: Computation and Reasoning: A Type Theory for Computer Science. Oxford Univ Press (1994)
16. Luo, Z.: Coercive subtyping. Journal of Logic and Computation 9(1), 105–130 (1999)
17. Luo, Z.: Type-theoretical semantics with coercive subtyping. Semantics and Linguistic Theory 20 (SALT20), Vancouver (2010)
18. Luo, Z.: Contextual analysis of word meanings in type-theoretical semantics. In: Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736 (2011)
19. Luo, Z.: Common nouns as types. In: Bechet, D., Dikovsky, A. (eds.) Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351 (2012)
20. Luo, Z., Soloviev, S., Xue, T.: Coercive subtyping: theory and implementation. Submitted manuscript (2012)
21. Martin-Löf, P.: Intuitionistic Type Theory. Bibliopolis (1984)
22. Montague, R.: The proper treatment of quantification in ordinary English. In: Hintikka, J., Moravcsik, J., Suppes, P. (eds.) Approaches to Natural Languages (1973)
23. Montague, R.: Formal Philosophy. Yale University Press (1974)
24. Moortgat, M.: Categorial type logics. In: van Benthem, J., ter Meulen, A. (eds.) Handbook of Logic and Language. Elsevier/Mit press (1997)
25. Morril, G.: Type Logical Grammar: Categorial Logic of Signs. Kluwer Academic Publishers (1994)
26. Nordström, B., Petersson, K., Smith, J.: Programming in Martin-Löf's Type Theory: An Introduction. Oxford University Press (1990)
27. Partee, B., Rooth, M.: Generalized conjunction and type ambiguity. In: Bauerle, R., S.C., von Stechow, A. (eds.) Meaning, use, and interpretation of language. Mouton De Gruyter (1983)

28. Pustejovsky, J.: The Generative Lexicon. MIT (1995)
29. Pustejovsky, J.: Meaning in Context: Mechanisms of Selection in Language. Cambridge Press (2005)
30. Ranta, A.: Type-Theoretical Grammar. Oxford University Press (1994)
31. Retoré, C.: Variable types for meaning assembly: a logical syntax for generic noun phrases introduced by 'most'. Recherches linguistiques de Vincennes 41, 83–102 (2012)
32. Sundholm, G.: Proof theory and meaning. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic III: Alternatives to Classical Logic. Reidel (1986)
33. Winter, Y.: Flexibility Principles in Boolean Semantics. MIT Press, New York (2002)
34. Winter, Y.: A unified semantic treatment of singular np coordination. Linguistics and Philosophy 19, 337–391
35. Xue, T., Luo, Z.: Dot-types and their implementation. Logical Aspects of Computational Linguistics (LACL 2012). LNCS 7351 (2012)

# A Implementations in Coq

We use Coq's predifined Type Universe instead of LType. Bvector is needed for vectors (Require Import Bvector). The coercion $A <_c Vec(A, 1)$ for proper nouns is not possible in Coq, so we have to introduce the coercions as separate entries.

## A.1 Conjoinable types

```
(* Categories*)
Definition CN := Set.
Parameters Bank Institution Human Man Woman Object Animal OObject: CN.
Parameter John Stergios Zhaohui : Man.
Parameter Mary: Woman.
Axiom mh : Man->Human.  Coercion mh : Man >-> Human.
Axiom wh : Woman->Human.  Coercion wh : Woman >-> Human.
Axiom ha: Human-> Animal. Coercion ha: Human>-> Animal.
Axiom ao: Animal->Object. Coercion ao:  Animal>-> Object.
Axiom ooo: OObject-> Object. Coercion ooo: OObject>->Object.
Parameter walk: Animal ->Prop.
Parameter talk cycle drive: Human->Prop.
Parameter attack killed: Animal -> Animal -> Prop.
Parameter If when: Prop-> Prop-> Prop.
Parameter the some most all: forall A:CN, (A->Prop)->Prop.
Parameter die: OObject-> Prop.
Parameter slowly agonizingly: forall A:CN, (A->Prop)->(A->Prop).
Parameter And: forall A:Type, A->A->A. (*Predefined Type universe
 instead of LType*)
(*Cases to check*)
Check And Man (Stergios)(Zhaohui)
Check And Man (Stergios)(Mary) (*does not go through because Mary:Woman*)
Check And Human (Stergios)(Mary) (*this is fine given Woman Man<Human*)
```

```
Check And ((Human->Prop)->Prop) (some Man)(some Woman) (*Quantifier NP coordination*)
Check And (forall A: CN, (A->Prop)->Prop) (some)(all).(*Quantifier coordination*)
Check And (Human->Prop) (cycle)(drive) (*VP coordination*)
Check And (forall A:CN, (A->Prop)->(A->Prop))(slowly)(agonizingly). (*VP adverb coordination*)
Check And (Prop->Prop->Prop) (If)(when) (*subordinate conjunction coordination*)
```

### A.2 Non-Boolean Conjunction

```
Require Import Bvector.
Variables n m: nat.
Parameter meetc:forall n:nat, vector Human(n+2)->Prop.    (*collective meet*).
Parameter John1 George1: vector Human 1.(*coercions do not work with vectors so we use Human
instead of Man here*)
(*Unit type for collective And*)
Inductive OneAndc : Set := Andc.
Definition AndSem1 :=  forall A: CN,forall n:nat,forall m:nat, vector (A)(n)
->vector(A)(m)->vector(A)(n+m).
Definition AndSem2 :=forall A: CN,forall n:nat,forall m:nat, ((vector A n)->Prop)->Prop->
((vector A m)->Prop)->Prop->((vector A (n+m))->Prop).
Parameter Andc1 : AndSem1.
Parameter Andc2 : AndSem2.
Definition a1 (a:OneAndc) : AndSem1 := Andc1.  Coercion a1 : OneAndc >-> AndSem1.
Definition a2 (a:OneAndc) : AndSem2 := Andc2.  Coercion a2 : OneAndc >-> AndSem2.
*Some interesting cases to check*
Check meetc 0 ((Andc:AndSem1 (Human)(1)(1)(John1)(George1)) (*John and George met, with both George
 and John of lower type*)
```

### A.3 Co-predication

```
(* Phy dot Info *)
Parameter Phy Phy1 Info : CN.             (*Phy1 should be taken to be the same as Phy*)
Record PhyInfo : CN := mkPhyInfo { phy :> Phy; info :> Info }.
Parameter Book: PhyInfo.
Parameter Event : CN.
Record EventPhy : CN := mkEventPhy { event :> Event; phy1 :> Phy1}. (*Phy1 is used because Phy
 cannot be used twice*)
Parameter lunch: EventPhy.
Axiom po: Phy->Object. Coercion po:Phy>->Object.
Axiom pp: Phy1->Phy. Coercion pp: Phy1>->Phy.(*We introduce this coercion to mean that the two
 Phy and Phy1 are the same.*)
Parameter was_given_to_someone_else: Object->Prop.
*Interesting case to check*)
Check was_given_to_someone_else (And(Object)(Book)(lunch)).
```

# A Predicative Operator and Underspecification by the Type Theory of Acyclic Recursion

Roussanka Loukanova

Sweden

**Abstract.** We introduce a generalized predicative operator in the type theory of acyclic recursion $L^{\lambda}_{ar}$. We are investigating the potential use of such an operator for computational semantics and algorithmic classification of predicative expressions. The paper is an investigation of several kinds of predicative expressions that are headed by the copula verb be, by using the language and theory $L^{\lambda}_{ar}$ of typed acyclic recursion and the introduced predicative operator.

**Keywords:** intension, denotation, underspecification, type theory, acyclic recursion

## 1 Background of $L^{\lambda}_{ar}$

The work Moschovakis [3] initiated development of a new approach to the mathematical notion of algorithm. A great prospect for applications of the new approach is to computational semantics of artificial and natural languages[1] (NLs). In particular, the theory of acyclic recursion $L^{\lambda}_{ar}$, see Moschovakis [4], models the concepts of meaning and synonymy in typed models. For initial applications of $L^{\lambda}_{ar}$ to computational syntax-semantics interface in Constraint-Based Lexicalized Grammar (CBLG) of human language (HL), see Loukanova [2].

Moschovakis' formal system $L^{\lambda}_{ar}$, see Moschovakis [4], is a higher-order type theory, which is a proper extension of Gallin's $\mathrm{TY}_2$, see Gallin [1], and thus, of Montague's Intensional Logic (IL), see Montague [6]. The type theory $L^{\lambda}_{ar}$ and its calculi extend Gallin's $\mathrm{TY}_2$, at the level of the formal language and its semantics, by using several means: (1) two kinds of variables (*recursion variables*, called alternatively *locations*, and *pure variables*); (2) by formation of an additional set of *recursion terms*; (3) systems of rules that form various calculi, i.e., the reduction calculus and the calculus of referential synonymy. In the first part of the paper, we give a very brief, informal view of the syntax and denotational semantics of the language of $L^{\lambda}_{ar}$. Then, we introduce the intensional semantics of $L^{\lambda}_{ar}$. The major part of the paper is devoted to using the possibilities for underspecification at the object level of the type system $L^{\lambda}_{ar}$ for analysis of expressions with predicative VPs formed by a head copula like the verb be.

[1] *Natural Language (NL)* is a traditional way of address to human languages, which we follow in this work. However, we maintain the view that natural languages form a broader class of languages in nature.

## 2 Mini introduction to the type theory $L^{\lambda}_{ar}$

**Types of $L^{\lambda}_{ar}$:** The set *Types* is the smallest set defined recursively (using a wide-spread notation in computer science): $\tau :\equiv e \mid t \mid s \mid (\tau_1 \to \tau_2)$.

### 2.1 Syntax of $L^{\lambda}_{ar}$

The vocabulary of $L^{\lambda}_{ar}$ consists of pairwise disjoint sets, for each type $\tau$: $K_\tau = \{\mathsf{c}_0, \mathsf{c}_1, \ldots, \mathsf{c}_{k_\tau}\}$, a finite set of *constants* of type $\tau$; $PureVars_\tau = \{\mathsf{v}_0, \mathsf{v}_1, \ldots\}$, a set of *pure variables* of type $\tau$; $RecVars_\tau = \{\mathsf{p}_0, \mathsf{p}_1, \ldots\}$, a set of *recursion variables*, called also *locations*, of type $\tau$.
**The Terms of $L^{\lambda}_{ar}$:** In addition to application and $\lambda$-abstraction terms, $L^{\lambda}_{ar}$ has recursion terms that are formed by using a designated recursion operator, which is denoted by the constant $\mathsf{where}$ and can be used in infix notation. The recursive rules[2] for the set of $L^{\lambda}_{ar}$ terms can be expressed by using a notational variant of "typed" BNF:

$$\begin{aligned} A :\equiv \mathsf{c}^\tau : \tau \mid x^\tau : \tau \mid B^{(\sigma\to\tau)}(C^\sigma) : \tau \mid \lambda v^\sigma(B^\tau) : (\sigma \to \tau) \\ \mid A_0^\sigma \ \mathsf{where}\, \{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\} : \sigma \end{aligned}$$

where $\{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}$ is a set of assignments that satisfies the *acyclicity condition* defined as follows: For any terms $A_1 : \sigma_1, \ldots, A_n : \sigma_n$, and locations $p_1 : \sigma_1, \ldots, p_n : \sigma_n$ (where $n \geq 0$, and $p_i \neq p_j$ for all $i, j$ such that $i \neq j$ and $1 \leq i, j \leq n$), the set $\{p_1 := A_1, \ldots, p_n := A_n\}$ is an *acyclic system of assignments* iff there is a function $\mathsf{rank} : \{p_1, \ldots, p_n\} \longrightarrow \mathbb{N}$ such that, for all $p_i, p_j \in \{p_1, \ldots, p_n\}$, if $p_j$ occurs free in $A_i$ then $\mathsf{rank}(p_j) < \mathsf{rank}(p_i)$.

Terms of the form $A_0^\sigma \ \mathsf{where}\, \{p_1^{\sigma_1} := A_1^{\sigma_1}, \ldots, p_n^{\sigma_n} := A_n^{\sigma_n}\}$ are called *recursion terms*. Intuitively, a system $\{p_1 := A_1, \ldots, p_n := A_n\}$ defines recursive computations of the values to be assigned to the locations $p_1, \ldots, p_n$. Requiring a ranking function $\mathsf{rank}$, such that $\mathsf{rank}(p_j) < \mathsf{rank}(p_i)$, means that the value of $A_i$, which is assigned to $p_i$, may depend on the values of the location $p_j$, as well as on the values of the locations $p_k$ with lower rank than $p_j$. An acyclic system guarantees that computations close-off after a finite number of steps. Omitting the acyclicity condition gives an extended type system $L^{\lambda}_{lr}$, which admits full recursion, but is not in the subject of this talk.

### 2.2 Two kinds of semantics of $L^{\lambda}_{ar}$

**Denotational Semantics of $L^{\lambda}_{ar}$:** The language $L^{\lambda}_{ar}$ has denotational semantics that is given by a definition of a denotational function for any semantic structure with typed domain frames. The denotational semantics of $L^{\lambda}_{ar}$ follows the structure of the $L^{\lambda}_{ar}$ terms, in a compositional way.

[2] In an explicit definition of the $L^{\lambda}_{ar}$ terms, the acyclicity condition is a proper part of the case of recursion terms, as the above notational variant of BNF is taken.

**Intensional Semantics of $L^{\lambda}_{ar}$:** The notion of intension in the languages of recursion covers the most essential, computational aspect of the concept of meaning. The *referential intension*, $\mathsf{Int}(A)$, of a meaningful term $A$ is the tuple of functions (a recursor) that is defined by the denotations $\mathsf{den}(A_i)$ ($i \in \{0, \dots n\}$) of the parts (i.e., the head sub-term $A_0$ and of the terms $A_1, \dots, A_n$ in the system of assignments) of its canonical form $\mathsf{cf}(A) \equiv A_0 \ \mathsf{where} \ \{p_1 := A_1, \dots, p_n := A_n\}$. Intuitively, for each meaningful term $A$, the intension of $A$, $\mathsf{Int}(A)$, is the *algorithm* for computing its denotation $\mathsf{den}(A)$. Two meaningful expressions are synonymous iff their referential intensions are naturally isomorphic, i.e., they are the same algorithms. Thus, the algorithmic meaning of a meaningful term (i.e., its sense) is the information about how to "compute" its denotation step-by-step: a meaningful term has sense by carrying instructions within its structure, which are revealed by its canonical form, for acquiring what they denote in a model. The canonical form $\mathsf{cf}(A)$ of a meaningful term $A$ encodes its intension, i.e., the algorithm for computing its denotation, via: (1) the basic instructions (facts), which consist of $\{p_1 := A_1, \dots, p_n := A_n\}$ and the head term $A_0$, that are needed for computing the denotation $\mathsf{den}(A)$, and (2) a terminating $\mathsf{rank}$ order of the recursive steps that compute each $\mathsf{den}(A_i)$, for $i \in \{0, \dots, n\}$, for incremental computation of the denotation $\mathsf{den}(A) = \mathsf{den}(A_0)$. Thus, the languages of recursion offer a formalisation of central computational aspects: denotation, with (at least) two semantic "levels": *referential intensions (algorithms)* and *denotations*. The terms in canonical form represent the algorithmic steps for computing semantic denotations by using all necessary basic components:

***Some notations and abbreviations:*** As a typical practice, we skip some "understood" parentheses, use different shapes and sizes of parentheses, and some extra parentheses for clarity.

- The symbol "$=$" is an identity predicate constant from the vocabulary of $L^{\lambda}_{ar}$. It is also the identity relation between objects from the semantic domains $\mathbb{T}$ of $L^{\lambda}_{ar}$ structures.
- The symbol "$\equiv$" is a meta-symbol (i.e., it is not in the vocabulary of the language $L^{\lambda}_{ar}$), which we use to specify orthographical identity between expressions and types of $L^{\lambda}_{ar}$. E.g., we use it to introduce abbreviations and aliases.
- The symbol "$:\equiv$" is a meta-symbol that we use in inductive definitions, e.g., of types and terms. We use it also for the replacement operation[3]:

**Definition 1.** *For any given terms $A : \sigma$, $X : \tau$ (typically, a variable or a constant), and $C : \tau$, the term $A\{\, X :\equiv C \,\} : \sigma$ is the result of the* simultaneous replacement *of all free (unless otherwise stated) occurrences of $X$ with $C$ in $A$.*

[3] The author is working on a forthcoming work on a formal introduction to the type theory of acyclic recursion, with more an detailed definitions.

## 3 Underspecified **be**

Let $P : \big((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}}\big)$ and $p : (\widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}))$ be recursion variables. We render the lexical entry be into an underspecified term (1):

$$\textsf{be} \xrightarrow{\textsf{render}} \lambda x P\big(\lambda y p(y)(x)\big) : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \tag{1}$$

Given typical nominal expressions, we use recursive rendering:

$$\begin{aligned} \textsf{good doctor} &\xrightarrow{\textsf{render}} good(doctor) : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) && && \text{(2a)}\\ &\Rightarrow_{\textsf{cf}} good(d) \text{ where } \{d := doctor\} && \text{by (ap)} && \text{(2b)} \end{aligned}$$

Then, in a compositional way:

$$\begin{aligned} \textsf{a good doctor} &\xrightarrow{\textsf{render}} a\big(good(doctor)\big) : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}} && && \text{(3a)}\\ &\Rightarrow a(g) \text{ where } \{g := good(doctor)\} && \text{by (ap)} && \text{(3b)}\\ &\Rightarrow_{\textsf{cf}} a(g) \text{ where } \{g := good(d),\, d := doctor\} && \text{by (rep3)} && \text{(3c)} \end{aligned}$$

Then, we can use (1) by suitable specifications depending on its complements:

$$\begin{aligned} \textsf{is a good doctor} &\xrightarrow{\textsf{render}} \lambda x P\big(\lambda y p(y)(x)\big) \text{ where} && && \text{(4a)}\\ &\qquad \{P := a\big[good(doctor)\big]\} : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) && && \text{(4b)}\\ &\Rightarrow_{\textsf{cf}} \lambda x P\big(\lambda y p(y)(x)\big) \text{ where} && && \text{(4c)}\\ &\qquad \{P := a(g),\, g := good(d),\, d := doctor\} && && \text{(4d)} \end{aligned}$$

Note that $p$ is a free recursion variable in (4a)–(4d). By this, we render the expression is a good doctor into an underspecified term, by virtue of the free recursion variable $p$. It is a reasonable to do so when there isn't sufficient information about how the verb be is used. In (5a)–(5g), $p$ is bound by the assignment $p := is$. Here we leave the question if the term $is$ is a constant or more complex term denoting the identity relation or some other relation, e.g., by adding time information by respecting the inflection of the verb be (or reflecting professional qualification of John over time). In what follows, the identity relation symbol "=" is a constant that denotes identity relation between objects of type $\widetilde{\mathsf{e}}$, i.e., $= \in K_{\widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})}$. We use curried terms for "=", e.g., $=(B)(A) \equiv A = B$, to avoid $\lambda$-abstracts where possible.

$$\begin{aligned} \textsf{John is a good doctor} &\xrightarrow{\textsf{render}} \big[\lambda x P\big(\lambda y p(y)(x)\big)\big](john) \text{ where} && && \text{(5a)}\\ &\qquad \{P := a\big[good(doctor)\big], && && \text{(5b)}\\ &\qquad\quad p := is\} && : \widetilde{\mathsf{t}} && \text{(5c)}\\ &\Rightarrow_{\textsf{cf}} \big[\lambda x P\big(\lambda y p(y)(x)\big)\big](j) \text{ where} && && \text{(5d)}\\ &\qquad \{P := a(g),\, g := good(d), && && \text{(5e)}\\ &\qquad\quad d := doctor, && && \text{(5f)}\\ &\qquad\quad p := is,\, j := john\} && && \text{(5g)} \end{aligned}$$

$$\mathsf{is\ Mary} \xrightarrow{\mathsf{render}} \lambda x P\big(\lambda y p(y)(x)\big)\ \mathsf{where} \tag{6a}$$
$$\{P := \Big(\lambda v\, v(m)\ \mathsf{where}\,\{m := mary\}\Big)\} \tag{6b}$$
$$: (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})$$
$$\Rightarrow_{\mathsf{cf}} \lambda x P\big(\lambda y p(y)(x)\big)\ \mathsf{where} \tag{6c}$$
$$\{P := \lambda v\, v(m), \tag{6d}$$
$$m := mary\} \qquad \text{by (B-S)} \tag{6e}$$

The sub-term $\big(\lambda v\, v(m)\ \mathsf{where}\,\{m := mary\}\big)$, in the assignment (6b), suggests the underspecified term $\big(\lambda x P\big(\lambda y p(y)(x)\big)\ \mathsf{where}\,\{P := \lambda v\, v(m)\}\big)$, which can be used for expressions that prompt a name, like "is ...", e.g., when it is clear that "..." has to be a name:

$$\mathsf{is}\ \ldots \xrightarrow{\mathsf{render}} \lambda x P\big(\lambda y p(y)(x)\big)\ \mathsf{where}\,\{P := \lambda v\, v(m)\} \tag{7}$$

In (6a)-(6e), $p$ is a free recursion variable. In (8a)–(8d), $p$ is bound to the equality constant, by the assignment $p := =$. (Depending on context, an utterance of this sentence may be interpreted differently, e.g., that the singer is named Mary.)

$$\mathsf{The\ singer\ is\ Mary} \xrightarrow{\mathsf{render}} \big[\lambda x P\big(\lambda y p(y)(x)\big)\big](t)\ \mathsf{where} \tag{8a}$$
$$\{P := \lambda v\, v(m),\ m := mary, \tag{8b}$$
$$t := the(r),\ r := singer, \tag{8c}$$
$$p := =\} \tag{8d}$$

## 4 Predicative operator

Throughout what follows, unless otherwise specified, we assume that $\mathsf{Pr}$ and $C$ are recursion variables of the following types:

$$\mathsf{Pr}, C \in RecVars, \tag{9a}$$
$$\mathsf{Pr} : \big((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})\big), \tag{9b}$$
$$C : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \tag{9c}$$

We render the base form of the lexeme be into the following underspecified $L^{\lambda}_{ar}$ term:

$$\mathsf{be} \xrightarrow{\mathsf{render}} \mathsf{Pr} : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}), \tag{10}$$

We can use (10) by suitable specifications of the recursion variable $\mathsf{Pr}$.

### 4.1 The predicative operator and time locations

In this paper, similarly to Moschovakis [4], we assume that a state is a quadruple $u = \langle i, j, k, A, \delta \rangle$ where $i$ is a possible world, $j$ is a period of time (which can collapse into a point), $k$ is a space location, $A$ is a speaker, and $\delta$ is a (speaker's reference) function, which assigns values to occurrences of speaker dependent expressions such as proper names.

In the rest of the paper[4], we use a *rigid reference operator* designated by the constant $\mathsf{dere}$ and defined similarly to Moschovakis [4]:

**Definition 2 (a-la-Carnap rigid reference operator $\mathsf{dere}$).**

$$\mathsf{dere}(X, a)(b) \overset{\mathsf{def}}{=} X(a), \quad \textit{for all } X : s \to \sigma, \textit{ and } a, b : s \tag{11}$$

Informally, for any term $X : s \to \sigma$, that may denote state dependent object[5], and any state $a : s$, $\mathsf{dere}(X, a) : s \to \sigma$ denotes a constant function such that $\mathsf{dere}(X, a)(b) = X(a)$, i.e., evaluated to the denotation of $X(a)$ in all states $b : s$.

We call the recursion variable $\mathsf{Pr}$ in (10) *underspecified predicative operator*. In this work, the recursion variable $\mathsf{Pr}$ may specify time (and aspect) information expressed by the inflection of the verb $\mathsf{be}$. In addition, it can be further specified with other information depending on phrasal combinations headed by the verb $\mathsf{be}$, e.g., with specific information carried by its complement.

$$\mathsf{was} \xrightarrow{\mathsf{render}} \mathsf{Pr} \text{ where} \tag{12a}$$

$$\{\mathsf{Pr} := \lambda X \big[ \mathsf{dere}(X)(u\{j :\equiv t'\}) \big] \}, \tag{12b}$$

$$\text{for } t' < \mathsf{time}(u) \tag{12c}$$

$$\mathsf{is} \xrightarrow{\mathsf{render}} \mathsf{Pr} \text{ where} \tag{13a}$$

$$\{\mathsf{Pr} := \lambda X \big[ \mathsf{dere}(X)(u\{j :\equiv t'\}) \big] \}, \tag{13b}$$

$$\text{for } t' \circ \mathsf{time}(u) \tag{13c}$$

In this work, the constant $<: \widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})$ denotes the precedence relation between times; the constant $\circ : \widetilde{\mathsf{e}} \to (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})$ denotes the overlapping relation between time periods. We use the infix relational notation for both.

In the following sections, we demonstrate this role of the predicative operator $\mathsf{Pr}$ by considering 3rd-singular past tense form of $\mathsf{be}$. We consider predicative VPs formed with the copula verb $\mathsf{be}$ and complements of the syntactic categories AdjV, predicative PPs, and NPs that are proper names, referential definite descriptions, or predicative indefinite descriptions.

[4] There are other possibilities for context and agent dependent reference operator in our future work on the topic, which are not in the subject of this paper.

[5] For sake of space we do not express the details between terms and denotations.

## 5 Predicative nominal descriptions

In general, if $\mathsf{A}$ is a nominal phrase, of syntactic category that is often denoted by NOM, and $\mathsf{A} \xrightarrow{\text{render}} A : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})$, then, by circumventing the indefinite (existential) quantification, we set:

$$\begin{aligned}
&\text{is an } \mathsf{A} \xrightarrow{\text{render}} \mathsf{Pr}(C) \text{ where} && (14a)\\
&\qquad \{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], && (14b)\\
&\qquad\ C := A\}, && (14c)\\
&\text{for } t' \circ \mathsf{time}(u) && (14d)
\end{aligned}$$

$$\begin{aligned}
&\text{was an } \mathsf{A} \xrightarrow{\text{render}} \mathsf{Pr}(C) \text{ where} && (15a)\\
&\qquad \{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], && (15b)\\
&\qquad\ C := A\}, && (15c)\\
&\text{for } t' < \mathsf{time}(u) && (15d)
\end{aligned}$$

In (14a)–(14c) and (15a)–(15c), we assume that $A$ is a proper (i.e., not an immediate) term, otherwise it is simpler to take $C \equiv A$ and spare $C := A$.

Now the VP is a good doctor can be rendered as follows:

$$\begin{aligned}
&\text{is a good doctor} \xrightarrow{\text{render}} \mathsf{Pr}(C) \text{ where} && (16a)\\
&\qquad \{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], && (16b)\\
&\qquad\ C := \mathit{good}(\mathit{doctor})\} && (16c)\\
&\qquad : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})\\
&\Rightarrow_{\mathsf{cf}}\ \mathsf{Pr}(C) \text{ where} && (16d)\\
&\qquad \{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], && (16e)\\
&\qquad\ C := \mathit{good}(d),\ d := \mathit{doctor}\}, && (16f)\\
&\text{for } t' \circ \mathsf{time}(u) && (16g)
\end{aligned}$$

$$\begin{aligned}
&\text{John is a good doctor} \xrightarrow{\text{render}} T_1 && (17a)\\
&\qquad \equiv \mathsf{Pr}(C)(\mathit{john}) \text{ where} && (17b)\\
&\qquad\qquad \{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], && (17c)\\
&\qquad\qquad\ C := \mathit{good}(\mathit{doctor})\} && (17d)\\
&\qquad\qquad : \widetilde{\mathsf{t}}\\
&\qquad \Rightarrow_{\mathsf{cf}}\ \mathsf{Pr}(C)(j) \text{ where} && (17e)\\
&\qquad\qquad \{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], && (17f)\\
&\qquad\qquad\ C := \mathit{good}(d),\ d := \mathit{doctor}, && (17g)\\
&\qquad\qquad\ j := \mathit{john}\}, && (17h)\\
&\text{for } t' \circ \mathsf{time}(u) && (17i)
\end{aligned}$$

In essence, the terms (17b) and (17e) ascribe the property denoted by the term $good(doctor)$, in the present time, to the object denoted by $john$. I.e., they have the same denotation as $\big[\mathsf{dere}\,\big(good(doctor)\big)(u\{j :\equiv t'\})\big](john)$ without stating that John is one of the good doctors, which is more closely rendered by (5a) and (5d). Now we consider the term (18a):

$$T_2 \equiv \big[\mathsf{dere}\,\big(good(doctor)\big)(u\{j :\equiv t'\})\big](john) \tag{18a}$$
$$\Rightarrow_{\mathsf{cf}} \big[\mathsf{dere}(C)(u\{j :\equiv t'\})\big](j)\ \mathsf{where} \tag{18b}$$
$$\{C := good(d),\ d := doctor, \tag{18c}$$
$$j := john\}, \tag{18d}$$
$$\text{for } t' \circ \mathsf{time}(u) \tag{18e}$$

**Corollary 1.** *The terms $T_1$, in (17a), and $T_2$, in (18a), are denotationally equivalent, while not algorithmically (referentially) synonymous, e.g.,*

$$T_1 \not\approx T_2 \tag{19a}$$
$$T_1 \models\!\!\!\models T_2 \tag{19b}$$

*Proof.* (19a) follows from the Referential Synonymy Theorem (see Moschovakis [4]). (19b) follows from the definition of the denotation function $\mathsf{den}$ (for the definition see Moschovakis [4]).

The algorithmic (i.e. computational) difference between $T_1$ and $T_2$ is that in $T_1$ the temporal information is moved into the assignments. The temporal reference is computed and stored in the location $\mathsf{Pr}$ by the assignment $\mathsf{Pr} := \lambda X\big[\mathsf{dere}(X)(u\{j :\equiv t'\})\big]$. The term $T_1$ presents a computational pattern $\mathsf{Pr}(C)(j)$ for predicative statements with a copula presented by $\mathsf{Pr}$ (here we consider just the copula $\mathsf{be}$). The pattern $\mathsf{Pr}(C)(j)$ provides possibilities for varying the predicate term $C : (\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}})$ and the individual term $j : \widetilde{\mathsf{e}}$ by their assignments. Moving the term $\lambda X\big[\mathsf{dere}(X)(u\{j :\equiv t'\})\big]$ into the assignment part of $\mathsf{Pr}$ provides variant temporal information, which can be modified or combined with additional information.

## 6 Proper names in a predicative position

In general, if $\mathsf{M}$ is a proper name and $\mathsf{M} \xrightarrow{\mathsf{render}} M$, then:

$$\mathsf{is\ M} \xrightarrow{\mathsf{render}} \mathsf{Pr}(C)\ \mathsf{where} \tag{20a}$$
$$\{\mathsf{Pr} := \lambda X\big[\mathsf{dere}(X)(u\{j :\equiv t'\})\big], \tag{20b}$$
$$C := =(m) \tag{20c}$$
$$m := M\}, \qquad \text{for } t' \circ \mathsf{time}(u) \tag{20d}$$

$$\text{was M} \xrightarrow{\text{render}} \text{Pr}(C) \text{ where} \tag{21a}$$
$$\{\text{Pr} := \lambda X\big[\,\text{dere}(X)(u\{j :\equiv t'\})\big], \tag{21b}$$
$$C :== (m) \tag{21c}$$
$$m := M\}, \qquad \text{for } t' < \text{time}(u). \tag{21d}$$

Now, from (21a), when $\text{M} \equiv \text{Mary}$ and $M \equiv \mathit{mary}$, by the reduction rule (recap), we have:

$$\text{The singer was Mary} \xrightarrow{\text{render}} \text{Pr}(C)(\mathit{the}(\mathit{singer})) \text{ where} \tag{22a}$$
$$\{\text{Pr} := \lambda X\big[\,\text{dere}(X)(u\{j :\equiv t'\})\big], \tag{22b}$$
$$C :== (m), \tag{22c}$$
$$m := \mathit{mary}\} \tag{22d}$$
$$\Rightarrow_{\text{cf}} \text{Pr}(C)(t) \text{ where} \tag{22e}$$
$$\{\text{Pr} := \lambda X\big[\,\text{dere}(X)(u\{j :\equiv t'\})\big], \tag{22f}$$
$$C :== (m), \tag{22g}$$
$$t := \mathit{the}(r),\ r := \mathit{singer}, \tag{22h}$$
$$m := \mathit{mary}\}, \qquad \text{for } t' < \text{time}(u). \tag{22i}$$

Note that the term (22e)–(22i) is not referentially synonymous to the terms in a direct rendering of the sentence The singer was Mary such as in (23a)-(23e), while these terms are denotationally equal. The head and the first assignment parts in (22e)–(22i) present the additional computational steps needed by the general predicative pattern.

$$\text{The singer was Mary} \xrightarrow{\text{render}} \big[\,\text{dere}\big(:=(\mathit{mary})\big)(u\{j :\equiv t'\})\big](\mathit{the}(\mathit{singer})) \tag{23a}$$
$$\Rightarrow_{\text{cf}} \big[\,\text{dere}\,(C)(u\{j :\equiv t'\})\big](t) \text{ where} \tag{23b}$$
$$\{C :== (m), \tag{23c}$$
$$t := \mathit{the}(r),\ r := \mathit{singer}, \tag{23d}$$
$$m := \mathit{mary}\}, \qquad \text{for } t' < \text{time}(u). \tag{23e}$$

## 7 Adjective phrases (AdjPs) in a predicative position

If A is a predicative[6] AdjP and $\text{A} \xrightarrow{\text{render}} A : (\widetilde{\text{e}} \to \widetilde{\text{t}})$, then:

$$\text{is A} \xrightarrow{\text{render}} \text{Pr}(C) \text{ where} \tag{24a}$$
$$\{\text{Pr} := \lambda X\big[\,\text{dere}(X)(u\{j :\equiv t'\})\big], \tag{24b}$$
$$C := A\}, \tag{24c}$$
$$\text{for } t' \circ \text{time}(u) \tag{24d}$$

[6] Which adjectives and which AdjPs are predicative is determined by specific grammars of NL.

$$\textsf{was A} \xrightarrow{\textsf{render}} \textsf{Pr}(C)\ \textsf{where} \tag{25a}$$
$$\{\textsf{Pr} := \lambda X\big[\,\textsf{dere}(X)(u\{j :\equiv t'\})\big], \tag{25b}$$
$$C := A\}, \tag{25c}$$
$$\text{for } t' < \textsf{time}(u). \tag{25d}$$

Then:

$$\textsf{Peter was happy} \xrightarrow{\textsf{render}} \textsf{Pr}(C)(p)\ \textsf{where} \tag{26a}$$
$$\{\textsf{Pr} := \lambda X\big[\,\textsf{dere}(X)(u\{j :\equiv t'\})\big], \tag{26b}$$
$$C := \mathit{happy}, \tag{26c}$$
$$p := \mathit{peter}\}, \tag{26d}$$
$$\text{for } t' < \textsf{time}(u). \tag{26e}$$

A direct rendering of the same sentence is in (28a)–(28b). The difference in the terms is in their heads and the additional assignment part (26b). The term (26b) determines the additional computational step needed by the general predicative pattern (26a).

$$\textsf{was happy} \xrightarrow{\textsf{render}} \textsf{dere}(\mathit{happy})(u\{j :\equiv t'\}) \tag{27a}$$

Then:

$$\textsf{Peter was happy.} \xrightarrow{\textsf{render}} \big[\,\textsf{dere}(\mathit{happy})(u\{j :\equiv t'\})\big](\mathit{peter}) \tag{28a}$$
$$\Rightarrow_{\textsf{cf}} \big[\,\textsf{dere}(C)(u\{j :\equiv t'\})\big](p)\ \textsf{where} \tag{28b}$$
$$\{C := \mathit{happy},\ p := \mathit{peter}\}, \tag{28c}$$
$$\text{for } t' < \textsf{time}(u). \tag{28d}$$

## 8 Prepositional phrases in a predicative position

For any predicative prepositional phrase (PP) A, in which the head preposition is relational, and such that $\textsf{A} \xrightarrow{\textsf{render}} A : (\widetilde{\textsf{e}} \to \widetilde{\textsf{t}})$, we set:

$$\textsf{is A} \xrightarrow{\textsf{render}} \textsf{Pr}(C)\ \textsf{where} \tag{29a}$$
$$\{\textsf{Pr} := \lambda X\big[\,\textsf{dere}(X)(u\{j :\equiv t'\})\big], \tag{29b}$$
$$C := A\}, \qquad \text{for } t' \circ \textsf{time}(u) \tag{29c}$$

$$\textsf{was A} \xrightarrow{\textsf{render}} \textsf{Pr}(C)\ \textsf{where} \tag{30a}$$
$$\{\textsf{Pr} := \lambda X\big[\,\textsf{dere}(X)(u\{j :\equiv t'\})\big], \tag{30b}$$
$$C := A\}, \qquad \text{for } t' < \textsf{time}(u). \tag{30c}$$

For example:

$$\text{on the table} \xrightarrow{\text{render}} on\big(the(table)\big) \tag{31a}$$
$$\Rightarrow_{\mathsf{cf}} on(t) \text{ where } \{t := the(b),\ b := table\} \tag{31b}$$

$$\text{the book} \xrightarrow{\text{render}} the(book) \tag{32a}$$
$$\Rightarrow_{\mathsf{cf}} the(b_1) \text{ where } \{b_1 := book\} \tag{32b}$$

From (30a) and (31b) by (recap), (rep3), (B-S), we have:

$$\text{was on the table} \xrightarrow{\text{render}} \mathsf{Pr}(C) \text{ where} \tag{33a}$$
$$\{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], \tag{33b}$$
$$C := on\big(the(table)\big)\} \tag{33c}$$
$$\Rightarrow_{\mathsf{cf}} \mathsf{Pr}(C) \text{ where} \tag{33d}$$
$$\{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], \tag{33e}$$
$$C := on(t),\ t := the(b),\ b := table\}, \tag{33f}$$
$$\text{for } t' < \mathsf{time}(u). \tag{33g}$$

Now, from (33d) and (32b) by (rep3), ..., (B-S), etc., we have:

$$\text{the book was on the table} \xrightarrow{\text{render}} \mathsf{Pr}(C)(the(book)) \text{ where} \tag{34a}$$
$$\{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], \tag{34b}$$
$$C := on\big(the(table)\big)\big(the(book)\big)\} \tag{34c}$$
$$\Rightarrow_{\mathsf{cf}} \mathsf{Pr}(C)(t_1) \text{ where} \tag{34d}$$
$$\{\mathsf{Pr} := \lambda X\big[\,\mathsf{dere}(X)(u\{j :\equiv t'\})\big], \tag{34e}$$
$$C := on(t), \tag{34f}$$
$$t := the(b),\ b := table, \tag{34g}$$
$$t_1 := the(b_1),\ b_1 := book\}, \tag{34h}$$
$$\text{for } t' < \mathsf{time}(u). \tag{34i}$$

## 9 Future work

We plan to extend the work in this paper by investigating the predictive operator for other applications. In particular, a perspective for other applications is to analyse other kinds of predicative expressions, other copula verbs, and other uses of the verb be, e.g., the existential be and passive aspects.

Another direction of work is computational syntax-semantics interface of predicative VPs. The analysis of the verb be, and other related semantic phenomena of NL, would be better by using some computational grammar for rendering NL expressions into $L^{\lambda}_{ar}$ terms. This can be done by using a generalized

CBLG covering variety of grammar systems. It is very interesting to see such rendering defined with the new grammatical framework GF, see Ranta [5].

Elaboration of the semantic domain frames of $L^{\lambda}_{ar}$ is very important for adequate coverage of semantic phenomena related to context dependency, e.g, by using resource situations (states) in semantic representations of sub-expressions, and in resolution of underspecification by using context information. A primary prospect in this direction is development of enriched semantic structures of versions of $L^{\lambda}_{ar}$ (and $L^{\lambda}_{lr}$ with full recursion) having enriched domain of the states, e.g., by using suitable versions of situation-theoretical structures.

## References


1. D. Gallin. *Intensional and Higher-Order Modal Logic*. North-Holland, 1975.
2. R. Loukanova. Semantics with the language of acyclic recursion in constraint-based grammar. In G. Bel-Enguix and M. D. Jiménez-López, editors, *Bio-Inspired Models for Natural and Formal Languages*, pages 103–134. Cambridge Scholars Publishing, 2011.
3. Y. N. Moschovakis. Sense and denotation as algorithm and value. In J. Oikkonen and J. Vaananen, editors, *Lecture Notes in Logic*, number 2 in Lecture Notes in Logic, pages 210–249. Springer, 1994.
4. Y. N. Moschovakis. A logical calculus of meaning and synonymy. *Linguistics and Philosophy*, 29:27–89, 2006.
5. A. Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford, 2011.
6. R. H. Thomason, editor. *Formal Philosophy: Selected Papers of Richard Montague*, ed. Richmond Thomason. Yale University Press, New Haven, Connecticut, 1974. Edited, with an introduction, by Richmond H. Thomason.

# A Speaker-Referring OT Pragmatics of Quantity Expressions

Chris Cummins

SFB 673 – Alignment in Communication, Universität Bielefeld, Germany
c.r.cummins@gmail.com

## Abstract

Constraint-based approaches to pragmatics have customarily focused on the hearer, and aim to identify the optimal interpretation of utterances. Blutner (2006 i.a.) has argued that it is necessary also to consider the role of the speaker, and thus motivates a bidirectional Optimality Theory (OT) pragmatics. However, as he notes, this may have limited explanatory potential from a processing standpoint. A recent account, focusing on expressions of quantity, proposes instead to model the speaker's choice of expression by unidirectional OT. In this paper I discuss the merits of this versus the bidirectional account, and in particular explore the implications for the hearer, who is required to solve the problem of utterance interpretation by different means in the two models. I consider the predictions arising from the unidirectional account, with reference to interpretation and processing, and discuss its theoretical implications for pragmatics in general.

## 1. Introduction

Perhaps the most general problem in linguistic pragmatics is to establish how enrichments to the meaning of an utterance are computed by hearers, given that – in principle – virtually any utterance may (under appropriate contextual conditions) be used to convey virtually any meaning.[1] Hendriks and de Hoop (2001) explore the possibility of treating this as a problem of constraint satisfaction on the part of the hearer. They analyse anaphora resolution as a case in which a highly ambiguous discourse entity, such as the "one" of "Who wants one?", acquires an interpretation that is optimal with respect to a set of contextual, intonational and syntactic constraints. Blutner (2000) argues, on the basis of examples of blocking and pronoun binding, that it is preferable to adopt a bidirectional OT account, making reference to the speaker as well as the hearer. Within such an account, one can explain the impermissibility of a given interpretation on the basis that such an interpretation would have been expressed in a different way. This provides a mechanism for generating the markedness implicatures discussed by Horn (1984), Levinson (2000) and others, in which the use of an indirect expression (such as "cause to die") implicates that the corresponding direct expression ("kill") would have been inappropriate.

However, as discussed by Blutner (2006), the implications of a bidirectional mechanism for processing are not generally straightforward. He observes that there are two methods for obtaining the Hornian markedness implicatures: either you posit linking constraints explicitly specifying that unmarked forms should carry unmarked meanings and marked forms should carry marked meanings, or you adopt an approach of 'weak bidirection' (Blutner 1998) which is more permissive than the strong form. Briefly, in the strongly bidirectional model, a form-meaning pair <f, m> is optimal if there exists no alternative meaning m' such that <f, m'> satisfies the constraints better than <f, m>, and there exists no alternative form f such that <f', m> satisfies the constraints better than <f, m>. In the weakly bidirectional model, a form-meaning pair <f, m> is 'super-optimal' if there exists no alternative meaning m' such that <f, m'> satisfies the constraints better than <f, m> *and* <f, m'> is itself super-optimal, and there exists no alternative form f' such that <f', m> satisfies the constraints better than <f, m> *and* <f', m> is itself super-optimal.

[1] For example, if S asks a yes/no question, an irrelevant response on the part of H may convey an affirmative or negative response, irrespective of the semantics of the response itself. This is traditionally analysed as an implicature arising from the flouting of relevance (Grice 1989).

In the absence of a linking constraint, neither marked forms nor marked meanings can participate in strongly optimal pairs. If we consider the form "cause to die", as a marked alternative to "kill" (Levinson 2000: 142), and pair this with the meaning $m_1$, the pair <cause to die, $m_1$> will satisfy the constraints less well than <kill, $m_1$>, and therefore not be optimal. However, <cause to die, $m_1$> can still be super-optimal, if <kill, $m_1$> is not itself super-optimal – in other words, if there is some $m_2$ such that <kill, $m_2$> satisfies the constraints better than <kill, $m_1$>.

Thus, as Blutner (2006: 16) puts it, weak bidirection "typically allows marked expressions to have an optimal interpretation, although both the expressions and the situation they describe have a more efficient counterpart". However, as he acknowledges, this weak bidirectional account requires global solution and consequently "[does] not even fit the simplest requirements of a psychologically realistic model of online, incremental interpretation" (ibid.). Meanwhile, the strong bidirectional approach is essentially *ad hoc* (Blutner 2000: 10).

An alternative approach, apparently under-explored within this line of research, is to consider the role of the speaker as primary. If we assume that the role of the hearer is to attempt to reconstruct the communicative intention of the speaker, then the authorship of the utterance is primary, in that the speaker determines which meanings are present in the utterance and therefore which meanings the hearer should attempt to reconstruct. In the above example, the 'correct' resolution of the meaning of "one" seems uncontroversially to be 'whatever the speaker meant by it'. From this perspective, positing a hearer-referring OT system seems unreasonable: for it to work, the speaker would have to encode intentions in a way that was decipherable by this system. However, hearer-referring accounts do not typically explain how the speaker is able to accomplish this non-trivial task, instead considering the utterance to be 'given', and implicitly assuming that it is related to the speaker's intention in some appropriate way. It could be more effective to posit instead that the speaker is engaging in a constraint-governed process of production and that the hearer attempts to 'undo' this to reconstruct the intended meaning. Such an approach resembles the Dual Optimization of Smolensky (1996), in which production and comprehension are distinct processes, rather than the simultaneous optimisation of bidirectional OT.

In this paper I discuss a recent proposal concerning the pragmatics of numerically-quantified expressions, which adopts the speaker-referring unidirectional stance. I consider its relation to bidirectional accounts of numerical quantification, and in particular I examine the obligations it places on the hearer to unpack the various aspects of communicated meaning. I will argue that this model has some explanatory advantages with respect to recent data, and that it also invites the drawing of testable hypotheses about the hearer's reasoning processes. I conclude by looking at the potential of this model to capture other aspects of meaning that have been considered in the OT semantics/pragmatics literature.

## 2. An OT Model of Speaker Behaviour

Cummins (2011) proposes an OT model of the production of numerically-quantified expressions. This domain has been studied from a constraint-based perspective (e.g. Krifka 2002, 2009), and is particularly amenable to such an approach for several reasons. Firstly, there are frequently numerous semantically truthful candidate expressions for a given communicative intention, because of the rich entailment relations that exist among numerical expressions: if "more than 100" is true, so is "more than 99/98/97…", and so on. There is therefore a non-trivial mapping problem to address in this domain. Secondly, the numerical domain tends to offer a convenient means for quantifying constraint violations: for instance, we could think of "more than 99" as including one extra possibility (in the cardinal case) by comparison with "more than 100". Thirdly, the use of numerical quantifiers has been argued to depend upon a range of considerations traditionally connected with different fields of enquiry, ranging from philosophical semantics through linguistic pragmatics to numerical psychology, and a constraint-based account provides a convenient way to capture all these competing factors within a single coherent model.

Drawing upon various sources of theoretical and experimental work, Cummins (2011) proposes six constraints on the use of numerically-quantified expressions: informativeness, granularity, numerical salience, quantifier simplicity, numeral priming, and quantifier priming. Two of these, numerical salience and quantifier simplicity, are treated as markedness constraints, as they govern the form of the utterance irrespective of context. The

others are treated as faithfulness constraints, as they govern the relationship between the 'situation' (broadly construed) and the utterance. Numeral and quantifier priming require the reuse of contextually activated material, granularity requires the use of the appropriate level of specificity in the choice of numeral (see Krifka 2009), and informativeness requires the use of a maximally informative expression given the communicative intention of the speaker (which in Cummins 2011 is considered initially in terms of excluding possibilities that the speaker knows to be false).

Cummins (2011) discusses how these constraints combine to yield novel predictions, as well as matching existing predictions under less stipulative assumptions. This work offers an alternative pragmatically-based account of the differences between "more than" and "at least" (Geurts and Nouwen 2007), also extending to related classes of expressions (Nouwen 2010). It also sketches an alternative explanation for the preferences for approximate interpretation of round number words discussed by Krifka (2009). Moreover, it draws novel predictions about the range of usage and interpretation associated with modified round numerals ("more than 100", etc.), which are borne out experimentally by Cummins, Sauerland and Solt (2012).

The above predictions are made both with respect to the choice of expression and to its interpretation. Clearly in order to draw predictions about interpretation it is necessary to take a position on the actions of the hearer within such a model. In the following section I discuss how the hearer's role differs within this account to that posited in bidirectional and hearer-referring OT approaches, before going on to consider the evidence that hearers do indeed behave in the way the speaker-referring account predicts.

## 3. The Hearer's Task in a Speaker-Referring Model

Under the assumptions of the model discussed above, the speaker's choice of expression is ultimately dependent on various facets of the situation (specifically, those referred to by the constraints). These include the speaker's communicative intention, but also factors pertaining to the discourse context (i.e. which linguistic entities are activated or salient within it) and the structure of the number and quantifier systems (i.e. which numbers and quantifiers are generally easier to use). To the speaker, all these considerations can be treated as part of the input, and the linguistic expression produced is the output.

The hearer's task, however, is not simply to reverse this process and attempt to map the utterance to the appropriate situation. In fact, the hearer also typically shares some of this information about the context, such as which discourse entities are activated and which numerals are generally salient. The hearer's role is to infer the information to which s/he is not privy, crucially including the speaker's intention.

In this respect, the hearer's task within this speaker-referring model is unlike the hearer's task within either a bidirectional model or a hearer-referring model. In the bidirectional case, the system specifies optimal form-meaning pairings, and if the hearer is able to identify these in the same way as the speaker, the hearer can simply read off the speaker's meaning. In the hearer-referring case, the hearer computes a preferred interpretation given the utterance and the hearer's own interpretative preferences (about which s/he may be presumed fully knowledgeable).

As a consequence, the unidirectional speaker-referring account is able to address aspects of pragmatics that do not appear to admit straightforward treatments within the other formalisms. Consider the case of "more than *n*", for round values of *n*, and its preferred interpretations. If the hearer wishes to describe a quantity in excess of 73, say, then (1a-c) are candidate utterances (among many others which I ignore here for expository purposes).

(1a) more than 73
(1b) more than 70
(1c) more than 60

Assuming no prior contextual commitments, the relevant constraints here are informativeness (INFO) and numeral salience (NSAL). Schematically, assuming that 60 and 70 are equally more salient than 73, the tableau would be as follows.

| | INFO | NSAL |
|---|---|---|
| more than 73 | | * |
| more than 70 | * | |
| more than 60 | ** | |

With INFO ranked above NSAL, "more than 73" would be preferred; with NSAL above INFO, "more than 70" would be preferred. Now by contrast, if the numeral "60" is activated in the preceding discourse, the numeral priming constraint (NPRI) also becomes relevant, and is violated by those forms that do not (re)use this number, resulting in the following tableau.

| | INFO | NSAL | NPRI |
|---|---|---|---|
| more than 73 | | * | * |
| more than 70 | * | | * |
| more than 60 | ** | | |

Now if INFO > {NSAL, NPRI}, "more than 73" is again preferred. However, if NPRI > INFO, "more than 60" is preferred. Only if NSAL > INFO > NPRI is "more than 70" still the preferred option.

Let us take the hearer's perspective and consider the rational interpretation that could be placed on the utterance arising under these circumstances. The hearer may assume that the utterance is optimal, given the constraints placed upon the speaker, if the speaker is behaving cooperatively. Within this model, 'optimal' specifically means that the speaker is producing the preferred utterance according to the situation (as they see it), including their intention, given their constraint ranking.

Ignoring contextual factors, the hearer might rationally interpret the utterances as follows. "More than 73" would arise only if INFO was the top-ranked constraint, and therefore would signal that the speaker's knowledge extends to the quantity under discussion being "greater than 73". "More than 70" would arise if NSAL > INFO, if the speaker's intention was "more than 70", or "more than 71", and so on up to "more than 79". It would also arise if INFO > NSAL and the speaker's knowledge extended to "more than 70". In either case, it would signal that the speaker could not commit to "more than 80", as otherwise "more than 80" would be a preferable output. Similarly, "more than 60" could arise only if the speaker was not able to commit to "more than 70".

Note that, in both cases, there is a many-to-one mapping between the speaker's intention and the optimal utterance: it is therefore not possible for the hearer to recapture the details of that intention. This again contrasts with the bidirectional model, in which (typically) a unique preferred interpretation is associated with each form, and vice versa. This permits the unidirectional model to generalise to a case in which distinct intentions are associated with identical preferred forms. Such a case would naturally arise if we considered a speaker's knowledge about a quantity as a probability distribution over possible values, as this would seem to permit greater nuance in the speaker's knowledge representation than could plausibly be represented in their choice of linguistic form.

If we consider contextual factors, the hearer's task becomes more complex. Suppose first that the hearer knows "60" to be contextually activated (e.g. because they have just asked whether the quantity under discussion is over 60). In this case, the interpretation of "more than 73" or "more than 70" proceeds as above. However, if the utterance is "more than 60", the hearer should be aware that this might reflect adherence to NPRI on the part

of the speaker, and thus be entirely compatible with an intention of "more than 70" and even higher values. So in this case the implicature that "more than 60" signals the speaker's inability to commit to "more than 70" should not arise.

Moreover, it is possible that a numeral might be primed without the hearer being aware of it. Therefore, any utterance should be interpreted (according to this model) as a potential case of priming. The utterance of "more than 73" might in fact reflect the action of a highly-ranked priming constraint in a situation in which the speaker has more precise information but considers 73 a significant threshold worthy of mention. The same applies to "more than 70", and thus the implicature that "not more than 80" should not be exceptionless. The hearer should, however, be able to infer that either the speaker is not willing to commit to "not more than 80", or the numeral mentioned has especial significance (or both).

It may appear that this model places unreasonable burdens on the hearer. Nevertheless, recent experimental data suggests that hearers not only can, but actually do interpret expressions of quantity in approximately this way. Cummins, Sauerland and Solt (2012) demonstrated that the preferred interpretation of an expression such as "more than 70" is one in which a pragmatic upper bound is inferred, i.e. the hearer infers that the speaker is unwilling to commit to the truth of "more than 80". They further showed that when the numeral is mentioned as a threshold value in the preceding conversational turn, this implicature is attenuated: i.e. the hearer more often accepts the idea that the speaker may know that "more than 80" holds but has deliberately chosen to make a weaker statement.

In both conditions, these inferences are frequent but not universal, and thus also accord with the prediction that hearers should factor in possible intentional use of a specific numeral. However, more direct evidence in support of this prediction comes from examples such as (2a-c), in which the selected numeral is not understood to give rise to a range implicature but rather to make reference to a critical threshold.

(2a) Samaras insists his party can still get more than 150 seats in Parliament to form a government on his own.[2]

(2b) [Sachin Tendulkar] has now scored more than 11,953 runs in the five-day game.[3]

(2c) The only painting to sell for more than $135 million was, perhaps unsurprisingly, one of Pollock's.[4]

The ostensibly correct interpretation of each of these sentences requires the inference that the choice of numeral carries additional information. In the case of (2a), it is that you need 151 seats to command a majority in the Greek Parliament; in the case of (2b), it is that the record Tendulkar broke stood at 11,953 runs; in the case of (2c), it is that the next-most-costly painting ever sold (at that time) cost $135 million (and was not by Pollock). While the precise inferences require encyclopaedic knowledge in order to be drawn, the fact that some kind of inference of this type is available appears to be a pragmatic property of the utterance, and can only be explained under the assumptions that (i) the speaker may choose to use a numeral that is contextually salient and (ii) the hearer is aware of this and interprets the utterance accordingly.

It could therefore be argued that this type of approach captures useful generalisations about the interpretation of utterances. However, the above discussion does not consider how the hearer is able to perform this task, which is presumed to be a complex matter of abductive inference. In the following section I turn to the question of heuristics with a view to outlining some practical hypotheses about the interpretation of quantity expressions.

[2] http://www.ekathimerini.com/4dcgi/_w_articles_wsite1_31219_22/04/2012_438798, retrieved 12 May 2012

[3] http://www.eatsleepsport.com/cricket/tendulkar-breaks-laras-test-record-800854.html#.T67HduVPQms, retrieved 12 May 2012.

[4] http://www.dailyiowan.com/2011/02/15/Opinions/21376.html, retrieved 12 May 2012.

## 4. Constraining the Hearer's Reasoning

Given the above constraint-based model, the hearer is entitled to assume that any alternative utterance would have been non-optimal, and to draw pragmatic inferences based on this observation. However, in principle this task requires infinitely many reasoning steps. Moreover, the inferences drawn would include many that were not useful. For instance, an utterance of "more than 70" indicates that "more than 1 million" and "more than 50" are both less appropriate as alternative utterances, but the corresponding inferences are hardly worth calculating. In the former case, "not more than a million" is unlikely to be an interesting conclusion (and is in any case entailed by stronger available inferences such as "not more than 100"). In the latter case, "more than 70" already entails "more than 50", and the inference that the latter is not maximally informative is redundant – the hearer already knows this on the basis of the semantics of the expression uttered.

Thus, to make this account psychologically plausible, it appears necessary also to specify some conditions governing the hearer's reasoning. In particular, we wish to constrain the possible alternatives that are considered – or, more precisely, to account for how the hearer manages to locate the alternatives that would give rise to pragmatically useful implicatures.

One possible technique is to appeal to scale granularity, in the sense of Krifka (2009). It is evident that many quantity scales have especially salient divisions and units: in number in general, the set (10, 20, 30, …) seems to constitute a natural coarse-grained subscale within the scale of integers, as does (50, 100, 150, …), while for instance (7, 17, 27, …) does not. These scales vary across domains: for instance, in time, (15, 30, 45, 60) is a scale for minutes, but (3, 6, 9, 12) is one for months, and so on. A hearer could exploit this by considering the next relevant scale point and using this to derive an alternative utterance that gives rise to pragmatic enrichments. In the case of expressions such as "more than", or existential contexts, the relevant direction is upwards, as in (3a) and (3b); in the case of "fewer than", it is downwards, as in (3c).

(3a) The child is more than 3 months old.
*Next scale point:* 6 months. *Alternative:* The child is more than 6 months old.
*Inference*: The child is not more than 6 months old.

(3b) We can comfortably accommodate 250 guests.
*Next scale point:* 300. *Alternative:* We can comfortably accommodate 300 guests.
*Inference:* It is not the case that we can comfortably accommodate 300 guests.

(3c) There are fewer than 70 places remaining.
*Next scale point*: 60. *Alternative*: There are fewer than 60 places remaining.
*Inference*: There are not fewer than 60 places remaining.

This type of reasoning is somewhat appealing intuitively, and matches the self-reported behaviour of some participants in Cummins et al. (2012). Within the constraint-based account discussed here, it is also a very efficient approach. The numeral salience and granularity constraints require the use of a number with a particular level of prominence, and disfavour alternatives with less prominent numbers. For instance, in a case such as (3a), it would potentially violate granularity to say "more than 4 months". Consequently, the inference that such an expression was actually untrue of the situation is not justified – the decision to use another expression might reflect adherence to a granularity constraint rather than any uncertainty on the speaker's part as to the validity of the semantic content. Thus, the inference based on the next scale point is the first safe inference. It is also the strongest safe inference, in that it entails all the inferences that could be drawn by considering more distant scale points ("not more than 9 months", "not more than 12 months", and so on).

In the case of numerals which have been previously mentioned in the context, the constraint-based approach predicts that no implicatures of the above kind are robust. In this case, the heuristic is very simple – the hearer should not consider expressions with any alternative numerals, on the basis that all such alternatives are potentially dispreferred for reasons that have nothing to do with their semantic content. However, this leaves open the question of how the hearer should respond if the numeral *might*, but might not, be being used because

of prior activation (which is arguably the typical case in real life, and is crucial to the interpretation of (2a-c) above). One possible account is that the hearer proceeds to consider alternatives, along the lines sketched above, but draws inferences with varying degrees of confidence. For instance, a hearer encountering (3c) and believing 70 not to be a salient number should draw the inference "not fewer than 60" with high confidence, while a hearer encountering (3c) when 70 might be a salient number should draw the inference with low confidence. The descriptive accuracy of this account is an empirical question that has not yet been explored.

It has also been observed in the literature that expressions such as "more than four" fail to implicate "not more than five", and "at least four" fails to implicate "not at least five" (Krifka 1999, Fox and Hackl 2006). In both cases, the semantic meaning and the implicature would, taken together, entail a precise value ("five" and "four" in the above examples). By contrast, Cummins et al. (2012) provide evidence that, for instance, "more than 70" does implicate "not more than 80". Moreover, it appears introspectively that "more than 4 metres" does similarly implicate "not more than 5 metres". This suggests that the granularity-based heuristic does not operate when scale points are adjacent, for the practical reason that it would generate enriched meanings that could be better expressed by bare numerals. However, an alternative explanation is that the implicatures discussed by Krifka (1999) and Fox and Hackl (2006) are sometimes available, but that the examples they consider involve the repetition of contextually salient values, thus invoking priming constraints and causing the failure of inference. For instance, (4) is especially felicitous in a context in which having three children is a critical threshold of some kind, e.g. for the receipt of benefits, that has already been established in the discourse. Again, empirical work could establish whether either or both of these explanations could be tenable.

(4) John has more than three children.

Perhaps a more interesting question concerns the treatment of different quantifiers. When we consider alternative expressions that involve the same number, clearly the number-referring constraints cannot adjudicate between them. In the model discussed here, the relevant constraints are quantifier simplicity and quantifier priming. Where no priming is in effect, the hearer should be entitled to conclude that the speaker could not use a simpler quantifier and the same number. Clearly it is crucial to establish what constitutes a 'simpler' quantifier: a bare numeral (i.e. null quantifier) could be argued to be the simplest, but precedence among other expressions is not straightforward. Cummins and Katsos (2010) argue from experimental data that the superlative quantifiers 'at least' and 'at most' are more complex than the corresponding comparative quantifiers 'more than' and 'fewer than'. This would entitle the hearer to draw the inferences specified in (5a) and (5b) from the use of the superlative quantifiers.

(5a) John has at least four children.
*Alternative 1:* John has four children.
*Inference 1:* The speaker is not certain that John has (exactly) four children.
*Alternative 2:* John has more than four children.
*Inference 2:* The speaker is not certain that John has more than four children.

(5b) John has at most four children.
*Alternative 1:* John has four children.
*Inference 1:* The speaker is not certain that John has (exactly) four children.
*Alternative 2:* John has fewer than four children.
*Inference 2:* The speaker is not certain that John has fewer than four children.

By contrast, if we hold the numeral constant, "John has more/fewer than four children" admits no such inferences. "At least/most" would be a more complex alternative and is therefore exempt from consideration. The bare numeral would give an alternative expression ("(exactly) four") that is already contradicted by the semantics of the original statement. Crucially, the inferences from "more than *n-1*" are thus systematically different from those arising from "at least *n*", even under the assumption that these expressions are semantically equivalent. Hence, this approach captures the difference observed by Geurts and Nouwen (2007) between the interpretation of comparative and superlative quantifiers. A similar account could be offered for any of the class B quantifiers of Nouwen (2010).

Generally, the major stipulation of this account is that certain specific alternatives are considered in place of actually uttered material. From a pragmatic perspective, this could be seen as occupying an intermediate position between default and contextual accounts of implicature. In common with default accounts, this approach suggests that certain expressions serve as triggers for inferences about the falsity of other expressions. However, in common with contextual accounts, this approach predicts that the implicatures do not arise in cases where their licensing conditions are not met. The justification for positing the partial element of defaultness in this model is practical, as discussed earlier – it is not feasible for a hearer to evaluate all the logically possible alternatives, and the search-space for pragmatically significant alternatives must be curtailed in some way. However, I propose that these potential alternatives are then evaluated on their merits, and that there is no rush to judgement. This accords with the experimental pragmatic literature, which has so far obtained limited evidence for any default inferences, while also respecting the intuitions of theorists such as Levinson (2000).

## 5. Conclusion

A unidirectional OT approach to numerical quantification yields novel predictions as to usage and interpretation. By comparison with alternative accounts, it places a considerable burden of reasoning on the hearer, and considers it generally impossible that speaker meaning is fully and accurately communicated. However, aspects of both hearers' behaviour and their introspective experience support the proposal. This account can be usefully enriched by positing particular heuristics as to how hearers reason, which aim to make the task of seeking pragmatic enrichments tractable by restricting it to a small finite search space. This approach in turn appears generally to cohere with an intermediate position between default and contextual accounts of pragmatic enrichment.

## References


Blutner, R. (1998): Lexical Pragmatics. Journal of Semantics 15, 115–162.

Blutner, R. (2000): Some Aspects of Optimality in Natural Language Interpretation. Journal of Semantics 17, 189–216.

Blutner, R. (2006): Embedded Implicatures and Optimality Theoretic Pragmatics. In: Solstad, T., Grønn, A. and Haug, D. (eds.) A Festschrift for Kjell Johan Sæbø: in partial fulfilment of the requirements for the celebration of his 50th birthday. Oslo.

Cummins, C. (2011): The Interpretation and Use of Numerically-Quantified Expressions. PhD thesis, available online at http://www.dspace.cam.ac.uk/handle/1810/241034.

Cummins, C. and Katsos, N. (2010): Comparative and Superlative Quantifiers: Pragmatic Effects of Comparison Type. Journal of Semantics 27, 271–305.

Cummins, C., Sauerland, U. and Solt, S. (2012): Granularity and Scalar Implicature in Numerical Expressions. Linguistics and Philosophy 35, 135-169.

Fox, D. and Hackl, M. (2006): The Universal Density of Measurement. Linguistics and Philosophy 29, 537–586.

Geurts, B. and Nouwen, R. (2007): "At least" et al.: the Semantics of Scalar Modifiers. Language 83, 533–559.

Grice, H. P. (1989): Studies in the Way of Words. Harvard University Press, Cambridge MA.

Hendriks, P. and de Hoop, H. (2001): Optimality Theoretic Semantics. Linguistics and Philosophy 24, 1–32.

Horn, L. R. (1984): Towards a New Taxonomy for Pragmatic Inference: Q-based and R-based Implicature. In: Schiffrin, D. (ed.) Meaning, Form and Use in Context (GURT '84), pp.11–42. Georgetown University Press, Washington DC.

Krifka, M. (1999): At least some Determiners aren't Determiners. In: Turner, K. (ed.), The Semantics/Pragmatics Interface from Different Points of View. Current Research in the Semantics/Pragmatics Interface Vol. 1, pp.257–292.

Krifka, M. (2002): Be Brief and Vague! And how Bidirectional Optimality Theory allows for Verbosity and Precision. In: Restle, D. and Zaefferer, D. (eds.) Sounds and Systems. Studies in Structure and Change. A Festschrift for Theo Vennemann, pp.439–458. Mouton de Gruyter, Berlin.

Krifka, M. (2009): Approximate Interpretations of Number Words: a Case for Strategic Communication. In: Hinrichs, E. and Nerbonne, J. (eds.) Theory and Evidence in Semantics, pp.109–132. CSLI Publications, Stanford.

Levinson, S. C. (2000): Presumptive Meanings. MIT Press, Cambridge MA.

Nouwen, R. (2010): Two Kinds of Modified Numerals. Semantics and Pragmatics 3, 1–41.

Smolensky, P. (1996): On the Comprehension/Production Dilemma in Child Language. Linguistic Inquiry 27, 720–731.

# Inducing lexical entries for an incremental semantic grammar

Arash Eshghi[1], Matthew Purver[1], Julian Hough[1], and Yo Sato[2]

[1] Interaction Media and Communication
School of Electronic Engineering and Computer Science
Queen Mary University of London
{arash,mpurver,jhough}@eecs.qmul.ac.uk
[2] Adaptive Systems Research Group
Science and Technology Research Institute
University of Hertfordshire
y.sato@herts.ac.uk

**Abstract.** We introduce a method for data-driven learning of lexical entries in an inherently incremental semantic grammar formalism, Dynamic Syntax (DS). *Lexical actions* in DS are constrained procedures for the incremental projection of compositional semantic structure. Here, we show how these can be induced directly from sentences paired with their complete propositional semantic structures. Checking induced entries over an artificial dataset generated using a known grammar demonstrates that the method learns lexical entries compatible with those defined by linguists, with different versions of the DS framework induced by varying only general tree manipulation rules. This is achieved without requiring annotation at the level of individual words, via a method compatible with work on linguistic change and routinisation.

## 1 Introduction

Dynamic Syntax (DS) is an inherently incremental semantic grammar formalism [1, 2] in which semantic representations are projected on a word-by-word basis. It recognises no intermediate layer of syntax (see below), and generation and parsing are interchangeable. Given these properties, it seems well suited for dialogue processing, and can in principle model common dialogue phenomena such as unfinished or co-constructed utterances, mid-utterance interruption and clarification etc. [3]. However, its definition in terms of semantics (rather than the more familiar syntactic phrase structure) makes it hard to define or extend broad-coverage grammars: expert linguists are required. On the other hand, as language resources are now available which pair sentences with semantic logical forms (LFs), the ability to automatically induce DS grammars could lead to a novel and useful resource for dialogue systems. Here, we investigate methods for inducing DS grammars, and present an initial method for inducing lexical entries from data paired with complete, compositionally structured, propositional LFs.

From a language acquisition perspective, this problem can be seen as one of constraint solving for a child: given (1) the constraints imposed through time by her understanding of the meaning of linguistic expressions (from evidence gathered from e.g.

her local, immediate cognitive environment, or interaction with an adult), and (2) innate cognitive constraints on how meaning representations can be manipulated, how does she go about separating out the contribution of each individual word to the overall meaning of a linguistic expression? And how does she choose among the many guesses she would have, the one that best satisfies these constraints?

This paper represents an initial investigation into the problem: the method presented is currently restricted to a sub-part of the general problem (see below). Future work will adapt it to a more general and less supervised setting where the input data contains less structure, where more words are unknown in each sentence, and applicable to real-world datasets – but the work here forms a first important step in a new problem area of learning explicitly incremental grammars in the form of constraints on semantic construction.

## 2 Previous work on grammar induction

Existing grammar induction methods can be divided into two major categories: supervised and unsupervised. Fully supervised methods, which use a parsed corpus as the training data and generalise over the the phrase structure rules to apply to a new set of data, has achieved significant success, particularly when coupled with statistical estimation of the probabilities for production rules that share the same LHS category (e.g. PCFGs [4]). However, such methods at best only capture part of the grammar learning problem, since they presuppose prior linguistic information and are not adequate as human grammar learning models. Unsupervised methods, on the other hand, which proceed with unannotated raw data and hence are closer to the human language acquisition setting, have seen less success. In its pure form —positive data only, without bias— unsupervised learning has been demonstrated to be computationally too complex ('unlearnable') in the worst case [5]. Successful cases have involved some prior learning or bias, e.g. a fixed set of known lexical categories, a probability distribution bias [6] or a hybrid, semi-supervised method with shallower (e.g. POS-tagging) annotation [7].

More recently another interesting line of work has emerged: supervised learning guided by *semantic* rather than syntactic annotation – more justifiably arguable to be 'available' to a human learner with some idea of what a string in an unknown language could mean. This has been successfully applied in Combinatorial Categorial Grammar [8], as it tightly couples compositional semantics with syntax [9, 10] and [11] also demonstrates a limited success of a similar approach with partially semantically annotated data that comes from a controlled experiment. Since these approaches adopt a lexicalist framework, the grammar learning involves inducing a lexicon assigning to each word its syntactic and semantic contribution.

Such approaches are only lightly supervised, using sentence-level propositional logical form rather than detailed word-level annotation. Also, grammar is learnt ground-up in an 'incremental' fashion, in the sense that the learner collects data and does the learning in parallel, sentence by sentence. Here we follow this spirit, inducing grammar from a propositional meaning representation and building a lexicon which specifies what each word contributes to the target semantics. However, taking advantage of the DS formalism, a distinctive feature of which is *word-by-word* processing of

semantic interpretation, we bring an added dimension of incrementality: not only is learning sentence-by-sentence incremental, but the grammar learned is word-by-word incremental, commensurate with psycholinguistic results showing incrementality to be a fundamental feature of human parsing and production [12, 13]. Incremental parsing algorithms have correspondingly been proposed [14–16], however, to the best of our knowledge, a learning system for an explicitly incremental grammar is yet to be presented – this work is a step towards such a system.

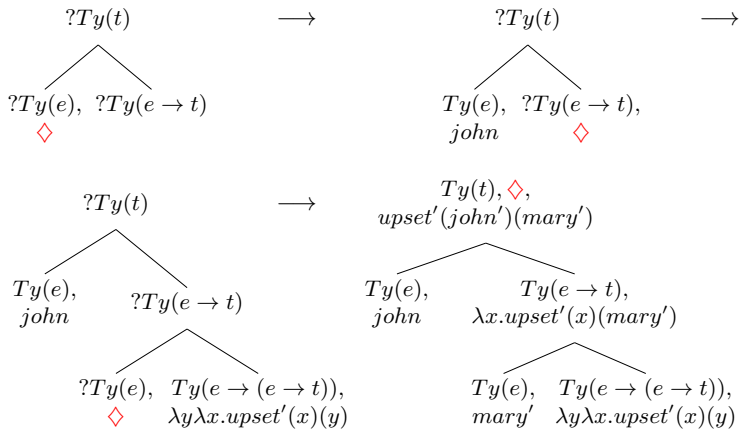



**Fig. 1.** Incremental parsing in DS producing semantic trees: *"John upset Mary"*

## 3 Dynamic Syntax

Dynamic Syntax is a parsing-directed grammar formalism, which models the word-by-word incremental processing of linguistic input. Unlike many other formalisms, DS models the incremental building up of *interpretations* without presupposing or indeed recognising an independent level of syntactic processing. Thus, the output for any given string of words is a purely *semantic* tree representing its predicate-argument structure; tree nodes correspond to terms in the lambda calculus, decorated with labels expressing their semantic type (e.g. $Ty(e)$) and formula, with beta-reduction determining the type and formula at a mother node from those at its daughters (Figure 1).

These trees can be *partial*, containing unsatisfied requirements for node labels (e.g. $?Ty(e)$ is a requirement for future development to $Ty(e)$), and contain a *pointer* ◇ labelling the node currently under development. Grammaticality is defined as parsability: the successful incremental construction of a tree with no outstanding requirements (a *complete* tree) using all information given by the words in a sentence. The input to our induction task here is therefore sentences paired with such complete, *semantic* trees, and what we try to learn are constrained lexical procedures for the incremental construction of such trees.

### 3.1 Actions in DS

The central tree-growth process is defined in terms of conditional *actions*: procedural specifications for monotonic tree growth. These take the form both of general structure-building principles (*computational actions*), putatively independent of any particular natural language, and of language-specific actions induced by parsing particular lexical items (*lexical actions*). The latter are what we here try to learn from data.

*Computational actions* These form a small, fixed set. Some merely encode the properties of the lambda calculus itself and the logical tree formalism (LoFT, [17]) – these we term *inferential* actions. Examples include THINNING (removal of satisfied requirements) and ELIMINATION (beta-reduction of daughter nodes at the mother). These actions are entirely language-general, cause no ambiguity, and add no new information to the tree; as such, they apply non-optionally whenever their preconditions are met.

Other computational actions reflect DS's predictivity and the dynamics of the framework. For example, replacing feature-passing concepts, e.g. for long-distance dependency, *ADJUNCTION introduces a single unfixed node with underspecified tree position; LINK-ADJUNCTION builds a paired ("linked") tree corresponding to semantic conjunction and licensing relative clauses, apposition and more. These actions represent possible parsing strategies and can apply optionally at any stage of a parse if their preconditions are met. While largely language-independent, some are specific to language type (e.g. INTRODUCTION-PREDICTION in the form used here applies only to SVO languages).

*Lexical actions* The lexicon associates words with lexical actions, which like computational actions, are each a sequence of tree-update actions in an IF..THEN..ELSE format, and composed of explicitly procedural *atomic actions* like `make`, `go`, `put` (and others). `make` creates a new daughter node. `go` moves the pointer to a daughter node, and `put` decorates the pointed node with a label. Fig. 2 shows a simple lexical action for *John*. The action says that if the pointed node (marked as $\Diamond$) has a requirement for type $e$, then decorate it with type $e$ (thus satisfying the requirement); decorate it with formula $John'$ and finally decorate it with the bottom restriction $\langle\downarrow\rangle\bot$ (meaning that the node cannot have any daughters). In case the IF condition $?Ty(e)$ is not satisfied, the action aborts, meaning that the word 'John' cannot be parsed in the context of the current tree.

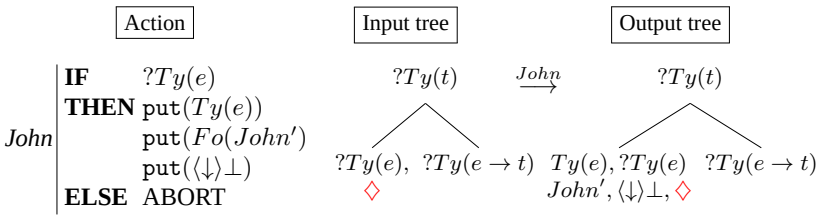
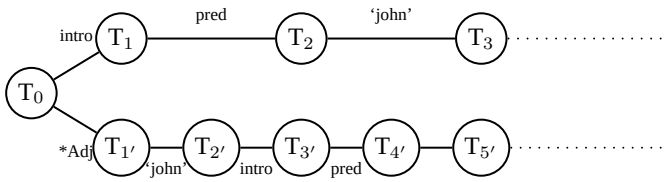



**Fig. 2.** Lexical action for the word 'John'

### 3.2 Graph Representation of DS Parsing

Given a sequence of words $(w_1, w_2, ..., w_n)$, the parser starts from the *axiom* tree $T_0$ (a requirement $?Ty(t)$ to construct a complete tree of propositional type), and applies the corresponding lexical actions $(a_1, a_2, \ldots, a_n)$, optionally interspersing computational actions – see Figure 1. [18] shows how this parsing process can be modelled on a *Directed Acyclic Graph* (DAG), rooted at $T_0$, with partial trees as nodes, and computational and lexical actions as edges (i.e. transitions between trees):




In this DAG, *intro*, *pred* and **Adj* correspond to the computational actions INTRODUCTION, PREDICTION and *-ADJUNCTION respectively; and 'john' is a lexical action. Different paths through the DAG represent different parsing strategies, which may succeed or fail depending on how the utterance is continued. Here, the path $T_0 - T_3$ will succeed if 'John' is the subject of an upcoming verb ("John upset Mary"); $T_0 - T_4$ will succeed if 'John' turns out to be a left-dislocated object ("John, Mary upset").

This DAG is taken to represent the *linguistic context* available during a parse, used for ellipsis and pronominal construal [19, 20]. It also provides us with a basis for implementing a best-first probabilistic parser, by taking the current DAG, plus a backtracking history, as the *parse state*. Given a conditional probability distribution $P(a_i|f(t))$ over possible actions $a_i$ given a set of features of the current partial tree $f(t)$, the DAG is then incrementally constructed and traversed such that at any node (partial tree), the most likely action (edge) is traversed first, with backtracking allowing other possibilities to be explored. Estimation of this probability distribution is not a problem we address here – we assume a known probability distribution for the known grammar fragment.

## 4 Learning lexical actions

### 4.1 Assumptions and Problem Statement

*Assumptions.* Our task here is data-driven learning of lexical actions for unknown words. Throughout, we will assume that the (language-independent) *computational* actions are known. To make the problem tractable at this initial stage, we further make the following simplifying assumptions: (1) The supervision information is structured: i.e. our dataset pairs sentences with the DS tree that expresses their predicate-argument structure – rather than just a less structured Logical Form as in e.g. [9] (Note that this does not provide word-level supervision: nodes do not correspond to words here) (2) The training data does not contain any pronouns or ellipsis (3) We have a seed lexicon such that there are no two adjacent words whose lexical actions are unknown in any given training example. As we will see this will help determine where unknown actions

begin and end. We will examine the elimination of this assumption below. Relaxing any of these assumptions means a larger hypothesis space.

*Input.* The input to the induction procedure to be described is now as follows:

- the set of computational actions in Dynamic Syntax, $G$.
- a seed lexicon $L_s$: a set of words with their associated lexical actions that are taken to be known in advance.
- a set of training examples of the form $\langle S_i, T_i \rangle$, where $S_i$ is a sentence of the language and $T_i$ – henceforth referred to as the *target tree* – is the complete semantic tree representing the compositional structure of the meaning of $S_i$.

*Target.* The output is the lexical actions associated with previously unknown words. We take these to be conditioned solely on the semantic type of the pointed node (i.e. their IF clause takes the form IF $?Ty(X)$). This is true of most lexical actions in DS (see examples above), but not all. This assumption will lead to some over-generation: inducing actions which can parse some ungrammatical strings.The main output of the induction algorithm is therefore the THEN clauses of the unknown actions: sequences of DS atomic actions such as *go*, *make*, and *put* (see Fig. 2). We refer to these sequences as *lexical hypotheses*. We first describe our method for constructing lexical hypotheses with a single training example (a sentence-tree pair). We then discuss how to generalise over and refine these outputs incrementally as we process more training examples.

### 4.2 Hypothesis construction

DS is *strictly monotonic*: actions can only *extend* the tree under construction, deleting nothing except satisfied requirements. Thus, hypothesising lexical actions consists in an incremental search through the space of all monotonic extensions of the current tree $T_{cur}$ that subsume (i.e. can be extended to) the target tree $T_t$. Not all possible trees and tree extensions are well-formed (meaningful) in DS, making the search constrained to a degree. The constraints are: (1) Lexical actions add lexical content —formula & type labels together— only to *leaf* nodes with the corresponding type requirement. Non-terminal nodes can thus only be given type *requirements* (later receiving their type and content via beta-reduction); (2) Leaf nodes can only be decorated by one lexical action, i.e. once a leaf node receives its semantic content, no lexical action will return to it (anaphora, excluded here, is an exception); (3) Once a new node is created, the pointer must move to it immediately and decorate it with the appropriate type requirement.

The process of hypothesis construction proceeds by locally and incrementally extending $T_{cur}$, using sequences of `make`, `go`, and `put` operations as appropriate and constrained as above, each time taking $T_{cur}$ one step closer to the target tree, $T_t$, at each stage checking for subsumption of $T_t$.This means that lexical actions are not hypothesised in one go, but left-to-right, word-by-word.

Hypothesis construction for unknown words is thus interleaved with parsing known words on the same DAG: Given a single training example, $\langle (w_1, \ldots, w_n), T_t \rangle$, we begin parsing from left to right. Known words are parsed as normal; when some unknown $w_i$ is encountered, $(w_i, \ldots, w_n)$ is scanned until the next known word $w_j$ is found or the end of the word sequence is reached (i.e. $j = n$). We then begin hypothesising for $w_i, \ldots, w_{j-1}$, incrementally extending the tree and expanding the DAG, using

both computational actions, and hypothesised lexical tree extensions until we reach a tree where we can parse $w_j$. This continues until the tree under development equals the target tree. All such possibilities are searched depth-first via backtracking until no backtracking is possible, resulting in a fully explored hypothesis DAG. Successful DAG paths (i.e. those that lead from the axiom tree to the target tree) thus provide the successful hypothesised lexical sub-sequences; these, once refined, become the THEN clauses of the induced lexical actions.

For unknown words, possible tree extensions are hypothesised as follows. Given the current tree under construction $T_{cur}$ and a target tree $T_t$, possible sequences of atomic actions (e.g. `go`, `put`, `make`) are conditioned on the node $N_t$ in $T_t$ which corresponds to – has the same address as – the pointed node $N_{cur}$ in $T_{cur}$. If $N_t$ is a leaf node, we hypothesise `put` operations which add each label on $N_t$ not present on $N_{cur}$, thus unifying them. Otherwise, we hypothesise adding an appropriate type requirement followed by `make`, `go` and `put` operations to add suitable daughters.

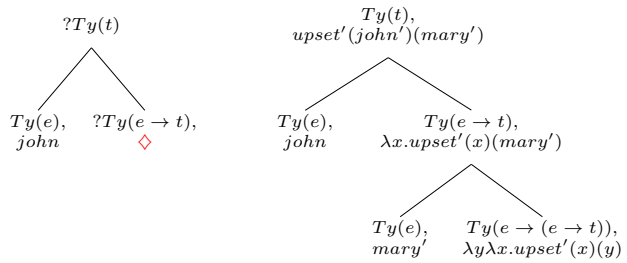


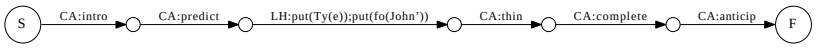

**Fig. 3.** The tree under development $T_{cur}$ (left) and the target tree $T_t$ (right)

Figure 3 shows an example. Here, $T_t$ is the complete tree on the right, and $T_{cur}$ the partial tree on the left. Since $T_{cur}$'s pointed node corresponds to a non-leaf node in $T_t$, we hypothesise two local action sequences: one which builds an argument daughter with appropriate type requirement (`make`$(\downarrow_0)$; `go`$(\downarrow_0)$; `put`$(?Ty(e))$), and another which does the same for a functor daughter (`make`$(\downarrow_1)$; `go`$(\downarrow_1)$; `put`$(?Ty(e \rightarrow (e \rightarrow t)))$).

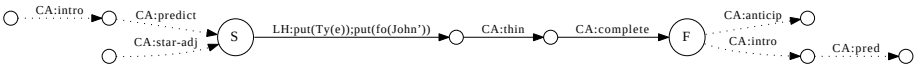
This method produces, for each training example $\langle S_i, T_i \rangle$, a hypothesis DAG representing all possible sequences of actions that lead from an Axiom tree to the associated target tree, $T_i$, using known lexical actions for known sub-strings of $S_i$, new hypothesised lexical actions for unknown sub-strings, and the known computational actions of DS. Such a DAG is in effect a mapping from the unknown word sub-strings of $S_i$ into sequences of local action hypotheses plus general computational actions that may have applied between words.

This method does not in principle require us to know any of the words in a given training example in advance if we employed some method of 'splitting' sequences associated with more than one adjacent word (a tactic employed in [10] as well as [11]). We will discuss this possibility in the next section, but currently, since producing a compact lexicon requires us to generalise over these action sequence hypotheses, we opt for the simpler alternative of assuming that in any pair of adjacent words one of them is known.

First Training Example: ‘john’ in subject position:




Second Training Example: ‘john’ on unfixed node, i.e. left-dislocated object:




Third training example: ‘john’ before parsing relative clause ‘who…’:

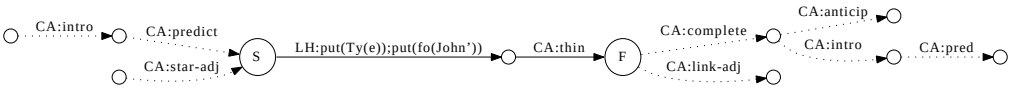



**Fig. 4.** Incremental intersection of candidate sequences

### 4.3 Hypothesis generalisation and refinement

Hypotheses produced from a single training example are unlikely to generalise well to other unseen examples: words occur at different syntactic/semantic positions in different training examples. We therefore require a method for the incremental, example-by-example refinement and generalisation of the action hypotheses produced for the same unknown word in processing different $\langle S, T_t \rangle$ pairs as above.

DS’s general computational actions can apply at any point before or after the application of a lexical action, thus providing strategies for adjusting the syntactic context in which a word is parsed. We can exploit this property to generalise over our lexical hypotheses: by partitioning a sequence into sub-sequences which can be achieved by computational actions, and sub-sequences which must be achieved lexically. Removing the former will leave more general lexical hypotheses.

However, we need a sequence generalisation method which not only allows computational action subsequences to be removed when this aids generalisation, but also allows them to become lexicalised when generalisation is not required – i.e. when all observed uses of a word involve them[3]. For example, the parsing of relative clauses in current DS grammars involves the computational action LINK-ADJUNCTION building a paired tree. Parse DAGs for sentences including relatives will therefore include LINK-ADJUNCTION in all successful paths. If every observed use of the relative pronoun *who* is now associated with a sequence containing LINK-ADJUNCTION, this computational action can become part of its lexical entry, thus increasing parsing efficiency.

**Generalisation through sequence intersection.** The hypothesis construction process above produces a set of parse/hypothesis DAGs, $D_i$, from a corresponding set of training examples, $\langle S_i = (w_1, \ldots, w_n), T_i \rangle$. Each of these provides a mapping, $CS_i(w)$, from any unknown word $w \notin L_s$ in $S_i$ (where $L_s$ is the seed lexicon), into a set of sequences of (partial) trees, connected by *candidate sequences* of actions (see Figure 4) made up of both computational actions and the local lexical hypotheses (marked as

[3] See e.g. [21] for an explanation of syntactic change via calcification or routinisation, whereby repeated use leads to parsing strategies becoming fixed within some lexical domain.

"LH" in Figure 4). Given our first simplifying assumption from Section 4.1 above, these candidate sequences must always be bounded on either side by *known* lexical actions. As we process more training examples, the set of candidate sequences for $w$ grows as per: $CS(w) = \bigcup_{n=1}^{i} CS_i(w)$. The problem now is one of generalisation over the candidate sequences in $CS(w)$.

Generalisation over these sequences proceeds by removing *computational* actions from the beginning or end of any sequence. We implement this via a single packed data-structure which we term the *generalisation DAG*, as shown in Figure 4: a representation of the full set of candidate sequences via their intersection (the central common path) and differences (the diverging paths at beginning and end), under the constraint that these differences consist only of computational actions. Nodes here therefore no longer represent single trees, but sets of trees. As new candidate sequences are added from new training examples, the intersection is reduced. Figure 4 shows this process over three training examples containing the unknown word 'john' in different syntactic positions. The 'S' and 'F' nodes here mark the start and finish of the current intersection subsequence –initially the entire sequence. As new training examples arrive, the intersection – the maximal common path – is reduced as appropriate. Lexical hypotheses thus remain as general as possible, with initial/final action sub-sequences which depend on syntactic context delegated to computational actions, but computational actions that *always* precede or follow a sequence of lexical hypotheses becomng lexicalised, as desired.

Eventually, the intersection is then taken to form the THEN clause of the new learned lexical entry. The IF clause is a type requirement, obtained from the pointed node on all partial trees in the 'S' node beginning the intersection sequence. As lexical hypotheses within the intersection are identical, and lexical hypotheses are constrained to add type information before formula information (see Section 4.2), any type information must be common across these partial trees. In Figure 4 for 'john', this is $?Ty(e)$, i.e. a requirement for type $e$, common to all three training examples.[4]

*Lexical Ambiguity.* Of course, it may well be that a new candidate sequence for $w$ cannot be intersected with the current generalisation DAG for $w$ (i.e. the intersection is the null sequence). Such cases indicate differences in lexical hypotheses rather than in syntactic context – either different formula/type decoration (polysemy) or different structure (multiple syntactic forms) – and thus give rise to *lexical ambiguity*, with a new generalisation DAG and lexical entry being created.

*Splitting Lexical Items.* Our assumption that no two adjacent words are unknown in any training example reduces the hypothesis space: candidate sequences correspond to single words and have known bounds. Relaxing this assumption can proceed in two ways: either by hypothesising candidate action sequences for multi-word sequences, and then hypothesising a set of possible word-boundary breaks (see e.g. [10, 11]); or by hypothesising a larger space of lexically distinct candidate sequences for each word. Due to the incremental nature of DS, hypotheses for one word will affect the possibilities for the next, so connections between lexical hypotheses for adjacent words must be maintained as the hypotheses are refined; we leave this issue to one side here.

[4] As we remove our simplifying assumptions, IF conditions must be derived by generalising over all features of the partial trees in the start node. We do not address this here.

## 5 Testing and Evaluation

We have tested a computational implementation of this method over a small, artificial data set: following [22] we use an existing grammar/lexicon to generate sentences with interpretations (complete DS trees), and test by removing lexical entries and comparing the induced results. As an initial proof of concept, we test on two unknown words: 'cook' (in both transitive and intransitive contexts) and 'John' (note that results generalise to all words of these types); the dataset consists of the following sentences paired with their semantic trees: (1) 'John likes Mary' (2) 'John, Mary likes' (3) 'Mary likes John' (4) 'Bill cooks steak' (5) 'Pie, Mary cooks' (6) 'Bill cooks'. (1), (2) and (3) have 'John' in subject, left-dislocated object, and object positions respectively. The structurally ambiguous verb 'cooks' was chosen to test the ability of the system to distinguish between its different senses.

| **Original** | |
|---|---|
| **IF** | $?Ty(e)$ |
| **THEN** | $put(Ty(e))$ |
| | $put(Fo(John'))$ |
| | $put(\langle\downarrow\rangle\ \bot)$ |
| **ELSE** | ABORT |

| **Induced** | |
|---|---|
| **IF** | $?Ty(e)$ |
| **THEN** | $put(Ty(e))$ |
| | $put(Fo(John'))$ |
| | $put(\langle\downarrow\rangle\ \bot)$ |
| | $delete(?Ty(e))$ |
| **ELSE** | ABORT |

**Fig. 5.** Original and Induced lexical actions for 'John'

The original and learned lexical actions for a proper noun ('John') are shown in Figure 5. The induced version matches the original with one addition: it deletes the satisfied $?Ty(e)$ requirement, i.e. it lexicalises the inferential computational action THINNING (see Figure 4: THINNING occurs in all observed contexts, hence its lexicalisation).

Verbs provide a stronger test case. In the original conception of DS [1], the computational actions INTRODUCTION and PREDICTION (together, INTRO-PRED) were taken to build argument and functor daughters of the root $Ty(t)$ node in English, accounting for the strict SVO word order and providing a node of $Ty(e \rightarrow t)$ as trigger for the verb. However, more recent variants [23] have abandoned INTRO-PRED in favour of a more language-general LOCAL*-ADJUNCTION rule, motivated independently for Japanese and all languages with NP clustering and scrambling (see [2], chapter 6).[5] Such variants require markedly different lexical actions for verbs, triggered by $?Ty(t)$ and building a complete propositional template while merging in argument nodes already constructed.

We therefore test verb induction given different sets of computational actions (i.e. one with LOCAL*-ADJUNCTION, one with INTRO-PRED). Fig. 6 shows the results for a transitive verb: the induced actions match the original manually defined actions for both variants, given only this change in the general computational actions available. With INTRO-PRED, the induced action is triggered by $?Ty(e \rightarrow t)$ and does not need to build the root node's daughters; with LOCAL*-ADJUNCTION, the action builds a complete propositional template triggered by $?Ty(t)$, and merges the unfixed node introduced by LOCAL*-ADJUNCTION into its appropriate subject position.

[5] This rule allows the addition of a second local *unfixed node* with its merge point restricted to any argument position. See [23, 2] for details.

Moreover, in the sequence intersection stage of our method, the action for the intransitive 'cook' (from training sentence (6), but not included here for reasons of space) was successfully distinguished from that of the transitive form in Fig. 6; the candidate sequences induced from sentences (4) and (5) were incompatible with those from (6), and thus resulted in a null intersection, giving rise to two separate lexical entries.

**Original**

| | |
|---|---|
| **IF** | $?Ty(e \rightarrow t)$ |
| **THEN** | $make(\downarrow_1); go(\downarrow_1)$ |
| | $put(Fo(\lambda y \lambda x.cook(x, y)))$ |
| | $put(Ty(e \rightarrow (e \rightarrow t)))$ |
| | $\texttt{put}(\langle\downarrow\rangle \perp)$ |
| | $go(\uparrow); make(\downarrow_0); go(\downarrow_0)$ |
| | $put(?Ty(e))$ |
| **ELSE** | ABORT |

**Induced with Intro-Pred**

| | |
|---|---|
| **IF** | $?Ty(e \rightarrow t)$ |
| **THEN** | $make(\downarrow_1); go(\downarrow_1)$ |
| | $put(?Ty(e \rightarrow (e \rightarrow t)))$ |
| | $put(Fo(\lambda y \lambda x.cook(x, y)))$ |
| | $put(Ty(e \rightarrow (e \rightarrow t)))$ |
| | $put(\langle\downarrow\rangle \perp)$ |
| | $delete(?Ty(e \rightarrow (e \rightarrow t)))$ |
| | $go(\uparrow); make(\downarrow_0); go(\downarrow_0)$ |
| | $put(?Ty(e))$ |
| **ELSE** | ABORT |

**Induced with Local*-Adj**

| | |
|---|---|
| **IF** | $?Ty(t)$ |
| **THEN** | $make(\downarrow_0); go(\downarrow_0)$ |
| | $put(?Ty(e))$ |
| | $merge$ |
| | $make(\downarrow_1); go(\downarrow_1)$ |
| | $put(?Ty(e \rightarrow t))$ |
| | $make(\downarrow_1); go(\downarrow_1)$ |
| | $put(?Ty(e \rightarrow (e \rightarrow t)))$ |
| | $put(Fo(\lambda y \lambda x.cook(x, y)))$ |
| | $put(Ty(e \rightarrow (e \rightarrow t)))$ |
| | $delete(?Ty(e \rightarrow (e \rightarrow t)))$ |
| | $go(\uparrow); make(\downarrow_0); go(\downarrow_0)$ |
| | $put(?Ty(e))$ |
| **ELSE** | ABORT |

**Fig. 6.** Original and Induced lexical actions for transitive 'cook'

## 6 Conclusions and Future work

In this paper we have outlined a novel method for the induction of new lexical entries in an inherently incremental and semantic grammar formalism, Dynamic Syntax, with no independent level of syntactic phrase structure. Methods developed for other non-incremental or phrase-structure-based formalisms could not be used here. Our method learns from sentences paired with semantic trees representing the sentences' predicate-argument structures: hypotheses for possible lexical action subsequences are formed under the constraints imposed by the known sentential semantics and by general facts about tree dynamics. Its success on an artificially generated dataset shows that it can learn new lexical entries compatible with those defined by linguists, with different variants of the DS framework inducible by varying only general tree manipulation rules.

Our research now focusses on relaxing our simplifying assumptions and applying to real data. Firstly, we are developing the method to remove the assumptions limiting the number of unknown words. Secondly, the induction method here is more supervised than we would like; work is under way to adapt the same method to learn from sentences paired not with trees but with less structured LFs using Type Theory with Records [24] and/or the lambda calculus, for which corpora are available. Other work planned includes integrating this method with the learning of conditional probability distributions over actions, to provide a coherent practical model of parsing and induction with incremental updates of both the lexical entries themselves and the parameters of the parsing model.

## References


1. Kempson, R., Meyer-Viol, W., Gabbay, D.: Dynamic Syntax: The Flow of Language Understanding. Blackwell (2001)
2. Cann, R., Kempson, R., Marten, L.: The Dynamics of Language. Elsevier, Oxford (2005)
3. Gargett, A., Gregoromichelaki, E., Kempson, R., Purver, M., Sato, Y.: Grammar resources for modelling dialogue dynamically. Cognitive Neurodynamics **3**(4) (2009) 347–363
4. Charniak, E.: Statistical Language Learning]. MIT Press (1996)
5. Gold, E.M.: Language identification in the limit. Information and Control **10**(5) (1967) 447–474
6. Klein, D., Manning, C.D.: Natural language grammar induction with a generative constituent-context mode. Pattern Recognition **38**(9) (2005) 1407–1419
7. Pereira, F., Schabes, Y.: Inside-outside reestimation from partially bracketed corpora. In: Proc. 30th Meeting of the Association for Computational Linguistics. (1992) 128–135
8. Steedman, M.: The Syntactic Process. MIT Press, Cambridge, MA (2000)
9. Zettlemoyer, L., Collins, M.: Online learning of relaxed CCG grammars for parsing to logical form. In: Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). (2007)
10. Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., Steedman, M.: Inducing probabilistic CCG grammars from logical form with higher-order unification. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. (2010) 1223–1233
11. Sato, Y., Tam, W.: Underspecified types and semantic bootstrapping of common nouns and adjectives. In: Proc. Language Engineering and Natural Language Semantics. (2012)
12. Lombardo, V., Sturt, P.: Incremental processing and infinite local ambiguity. In: Proceedings of the 1997 Cognitive Science Conference. (1997)
13. Ferreira, F., Swets, B.: How incremental is language production? evidence from the production of utterances requiring the computation of arithmetic sums. Journal of Memory and Language **46** (2002) 57–84
14. Hale, J.: A probabilistic Earley parser as a psycholinguistic model. In: Proc. 2nd Conference of the North American Chapter of the Association for Computational Linguistics. (2001)
15. Collins, M., Roark, B.: Incremental parsing with the perceptron algorithm. In: Proceedings of the 42nd Meeting of the ACL. (2004) 111–118
16. Clark, S., Curran, J.: Wide-coverage efficient statistical parsing with CCG and log-linear models. Computational Linguistics **33**(4) (2007) 493–552
17. Blackburn, P., Meyer-Viol, W.: Linguistics, logic and finite trees. Logic Journal of the Interest Group of Pure and Applied Logics **2**(1) (1994) 3–29
18. Sato, Y.: Local ambiguity, search strategies and parsing in Dynamic Syntax. In: The Dynamics of Lexical Interfaces. CSLI (2010) to appear.
19. Cann, R., Kempson, R., Purver, M.: Context and well-formedness: the dynamics of ellipsis. Research on Language and Computation **5**(3) (2007) 333–358
20. Purver, M., Eshghi, A., Hough, J.: Incremental semantic construction in a dialogue system. In: Proc. of the 9th International Conference on Computational Semantics. (2011) 365–369
21. Bouzouita, M.: At the syntax-pragmatics interface: clitics in the history of spanish. In: Language in Flux: Dialogue Coordination, Language Variation, Change and Evolution. College Publications, London (2008) 221–264
22. Pulman, S.G., Cussens, J.: Grammar learning using inductive logic programming. Oxford University Working Papers in Linguistics **6** (2001)
23. Cann, R.: Towards an account of the english auxiliary system: building interpretations incrementally. In: Dynamics of Lexical Interfaces. Chicago: CSLI Press (2011)
24. Cooper, R.: Records and record types in semantic theory. Journal of Logic and Computation **15**(2) (2005) 99–112

# Modelling language, action, and perception in Type Theory with Records

Simon Dobnik, Robin Cooper, and Staffan Larsson

Department of Philosophy, Linguistics and Theory of Science
University of Gothenburg, Box 200, 405 30 Göteborg, Sweden
{simon.dobnik@,staffan.larsson@ling,robin.cooper@ling}.gu.se
http://www.flov.gu.se

**Abstract.** We present a formal model for natural language semantics using Type Theory with Records (TTR) and argue that it is better suited for representing the meaning of spatial descriptions than traditional formal semantic models. Spatial descriptions include perceptual, conceptual and discourse knowledge which we represent all in a single framework. Being a computational framework TTR is suited for modelling language and cognition of conversational agents in robotics and virtual environments where interoperability between language, action and perception is required. The perceptual systems gain access to abstract conceptual meaning representations of language while the latter can be justified in action and perception.

**Keywords:** language, action, perception, formal semantics, spatial descriptions, learning and classification

## 1 Introduction

Classical formal model-theoretic semantics [2] does not deal with perception, action and dynamic meaning. It assumes that there is some fixed denotation function which mediates between natural language or semantic representations and the world but how this function is determined is left unaccounted for. Furthermore, there are many cases where the meaning of linguistic expressions is underspecified and can only be recovered from the context. Deictic pronouns are a well known example – they must be resolved against a contextually present entity which can typically be recovered from the discourse. Spatial descriptions such as "to the left of" or "near" rely on the physical context to an even greater extent as they must be evaluated against a particular pair of objects, their geometric representation and arrangement and the size of the scene. At the same time they also rely on our world knowledge about the interaction of objects.

Previously, there has not been a unified model of spatial language. Proposals include first order logic representations that we find in [30] and [23]. These only represent the conceptual knowledge but ignore how this knowledge maps to the environment. On the other hand there are approaches such as [13] and [21] that completely ignore the conceptual dimension of spatial descriptions and only consider perceptual parameters such as distances and angles in two or three dimensional space.

A need for an integrated model of language, action and perception is expressed most strongly in robotics ([27], [26], [15] and [22]). They are concerned

to represent and apply linguistic knowledge in robotic control (of a particular system) but they are less concerned with developing a general theory of meaning. [11] describes a system where mobile robots situated in a real room learn the meaning of perceptual descriptions as classifiers which they use to generate descriptions of new scenes or to answer questions about the scene and perform actions. In this paper we build on this experience by attempting to re-formulate it in a theoretical framework known as Type Theory with Records (TTR) [4, 6] which can represent perceptual, conceptual and discourse constraints that make up the meaning of words of natural language. This way we take an important step towards a more cognitive model of robotic perception and control.

## 2 TTR: a theory of natural language semantics

Types are intensional – that is, there can be distinct types which have identical extensions. They are semantic categories which are linked to perception. Perception events are assignments to types. The notion of truth is linked to *judgements* that an object $a$ is of type $T$ ($a : T$).

The theory defines a few basic types such as *Ind* and *Ev* which correspond to basic human conceptual categories such as individuals and events[1] but there are also complex types built from simple types and from other complex types. Most important among these are *record types* which are similar to feature structures which contain label-value pairs. Proof objects of record types are called records. Here is an example of what a record type *Apple* might be:

$$\begin{bmatrix} \mathrm{x} & : & \mathit{Ind} \\ \mathrm{c_{apple}} & : & \mathrm{apple(x)} \\ \mathrm{c_{ash}} & : & \mathrm{apple\text{-}shape(x)} \\ \mathrm{c_{ac}} & : & \mathrm{apple\text{-}colour(x)} \\ \mathrm{c_{at}} & : & \mathrm{apple\text{-}taste(x)} \\ \mathrm{c_{as}} & : & \mathrm{apple\text{-}smell(x)} \end{bmatrix}$$

The example introduces types which are constructed from predicates (like "apple") and objects which are arguments to these predicates like x. An object belonging to such *p-type* (also known as a *proof object* or a *witness*) can be considered to be an event or a situation [9, 1]. It follows that types can be used as propositions true just in case there is an object of that type. Types constructed with predicates may be dependent. An argument of a predicate may be represented by labels (on the left of ":") elsewhere in the record type. The type of $\mathrm{c_{ash}}$ is dependent on x (as is also $\mathrm{c_{ac}}$). The letter "c" in labels stands for constraints.

As demonstrated by this example the type representation combines both conceptual ($\mathrm{c_{apple}}$) and perceptual constraints ($\mathrm{c_{ash}}$) that must be verified against records containing individuals in order that such individuals may be judged as of type *Apple*. The model brings the semantic theory for natural language and models of perception and action in robotics close together. The verification may be modelled as classification where the classifier is learned through

[1] Later this set will be extended with *Real* and *Vector* for representing real numbers.

machine learning. [17] presents a model of a single-layer perceptron within the TTR framework. The important consequence of such a combined model is that natural language semantics does not only involve conceptual or categorial knowledge but also distributional knowledge from perception. This model has, among other things, advantages in the treatment of intensionality, vagueness and dialogue structure [5, 18, 7, 14]. The model also contributes to (artificial) perception since this gets access to conceptual knowledge represented in language which recently has been argued to be necessary for a more successful image recognition [12]. Overall, the TTR brings formal semantics closer to human cognition than classical model-theoretic approaches.

## 3 Spatial descriptions

By spatial descriptions we mean two or three place prepositional predicates that refer to the location of objects usually known as the *located object* and the *reference object* [16] or *referent* and *relatum* [23] or *figure* and *ground* [28]. Examples are: *on(spider,wall)* and *between(apple,vase,glass)*. Events may also be arguments of such predicates as shown by *in(eat(George,apple),garden)* or *in(put(book),bag)*. The preceding references provide a survey of semantics of spatial descriptions in English. Its constraints fall under four headings:

1. **Scene geometry:** the size of the scene and the angles and distances between objects as represented, for example, in the *attentional vector sum model* [25];
2. **View on the scene:** spatial descriptions do not refer to the actual objects but to their geometric representations which may be points, lines, areas and volumes. For example:"the crack in the vase" refers to a situation where the vase is viewed as a surface whereas "the water in the vase" refers to the vase as a volume enclosed by the object.
3. **World knowledge:** [8] show experimentally that human participants sometimes ignore geometry and consider descriptions such as "the umbrella is over a man" acceptable even in situations where the umbrella is held horizontally but is providing protection from the rain whereas for "above" the geometry is more important.
4. **Perspective:** directionals such as *to the left of* require a contextually defined perspective – something on may left may be on your right or may be behind relative to someone else. In our forthcoming work we discuss how perspective is aligned in conversation.

## 4 A TTR account of spatial descriptions

In this section we discuss how the constraints (1–3) can be expressed in TTR to represent the meaning for a robotic agent.

### 4.1 Objects and geometric spatial representations

In robotics the location of the robot and other objects in its environment is frequently determined using a method known as SLAM which stands for simultaneous localisation and map building [10]. A robot starts in an unknown

location and in an unknown environment and uses its relative observations to build an absolute map of the environment incrementally using probabilistic unsupervised machine learning. A laser scanner detects distances to points: a set of beams is emitted and a vector of distances is returned. The observation is made at a particular time and at a particular location on the estimated global map. An observation can therefore be represented as a record of the following type:

$$\mathit{Observation} = \begin{bmatrix} \text{scan} & : & \mathit{RealVector}_n \\ \text{time} & : & \mathit{Time} \\ \text{at} & : & \mathit{Real}_2 \\ \text{c}_{\text{ls}} & : & \text{laser-scan(scan)} \end{bmatrix}$$

where $\mathit{RealVector}_n$ is the type of vectors of $n$ real numbers, $\mathit{Time}$ is the type of the temporal record when the observation is made, $\mathit{Real}_2$ is the type of the location where the observation is made and there is a constraint specifying a dependency between the object scan and the p-type laser-scan. The observations from the sensors of some entity are records which form a probability distribution which corresponds to the type. These records of perceptual samples are not the actual objects in the world (if these indeed exist) – in other words one cannot really access the thing-in-itself.

During the learning stage, the SLAM algorithm, here simplified as a function $f$, takes iterative sensor scans of distances to points (scan : $\mathit{RealVector}_n$) and constructs an absolute map of the environment consisting of $\mathit{Point}$ objects (reification). When the map is initialised it contains no points. The value of the "at" label is a type $\mathit{RealVector}_2$, a vector containing the $x$ and $y$ coordinates of an individual $a$ on the global absolute map with some random origin.

$f$: $\mathit{Observation} \rightarrow (\mathit{PointMap} \rightarrow \mathit{PointMap})$

$$\mathit{Point} = \begin{bmatrix} \text{a} & : & \mathit{Ind} \\ \text{at} & : & \mathit{RealVector}_2 \\ \text{c}_{\text{point}} & : & \text{point(a)} \\ \text{c}_{\text{loc}} & : & \text{loc(a,at)} \end{bmatrix} \qquad \mathit{PointMap} = \begin{bmatrix} \text{p}_1 & : & \mathit{Point} \\ \vdots & & \vdots \\ \text{p}_\text{n} & : & \mathit{Point} \end{bmatrix}$$

Note that the robot is mobile but the points are static. After the learning step a mobile robot can use the map it has previously built and its current observation to find out its current location on that map. The function $g$ is applied every time the robot changes its location.

$$g\text{: } \mathit{Observation} \rightarrow \mathit{PointMap} \rightarrow \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{at} & : & \mathit{RealVector}_2 \\ \text{c}_{\text{robot}} & : & \text{robot(x)} \\ \text{c}_{\text{loc}} & : & \text{loc(x,at)} \end{bmatrix}$$

One could object to our treatment of individuals as points. However, we do refer to whole objects as if they were points, for example "This point here is the museum" (pointing at a map) or "The town is at the border" (a point at a line). Complex objects will be sets of simple point objects. Such cases are also attested in human cognition as exemplified by the well-known sorites paradox. In computational approaches to spatial cognition typically a bounding region or

a volume is defined which envelopes the object and which is known as convex hull. We need to specify at least three points for a region and at least four points for a volume.

$$f\text{: } \lambda r.\ \mathit{PointMap}\ (\begin{bmatrix} \mathrm{a} & : & \mathit{Ind} \\ \mathrm{p_1} & : & \mathrm{r}.\mathit{Point} \\ \vdots & & \\ \mathrm{p_n} & : & \mathrm{r}.\mathit{Point} \\ \mathrm{c_{reg}} & : & \mathrm{region(a)} \\ \mathrm{c_{inc}} & : & \mathrm{includes(a,\langle p_1,\ldots p_n\rangle)} \\ \text{conv-hull} & : & \langle \mathrm{p_i},\ \mathrm{p_j},\ \mathrm{p_k}\rangle \\ \mathrm{c_{hulled}} & : & \mathrm{hulled(\langle p_1,\ldots p_n\rangle,conv\text{-}hull)} \end{bmatrix})$$

Determining the function $f$ which picks out a list of convex points (conv-hull) and all the points within the region hulled by these points from the point map is not trivial. It could be learned from sets of annotated maps where objects of type *Point* may be enriched with other features such as colour beforehand. The constraint $\mathrm{c_{hulled}}$ ensures that all the points from the point map $\mathrm{p_1}\ldots\mathrm{p_n}$ fall within the convex region defined by the convex points. Note also that a region now represents a new a : *Ind*.

In [11] we use such maps as geometric spatial representations of scenes from which we learn classifiers for spatial descriptions. This is more suitable for the task than image data since it gives us very accurate representations of distances and angles in a scene. The identification of objects is done manually.

Being a semantic theory of natural language TTR provides an account of events which can be extended to the domain of robotic control. By representing the semantics of natural language and of robotic control within the same framework we facilitate the mediation of information between the two and give robotic control a more cognitive aspect. A system using such representations could exploit interaction in natural language with humans to learn new more complex actions from the primitive ones (explanation based learning [20]) or use knowledge expressed in types to reason about the outcomes of actions before committing to a particular one (mental simulations used for planning).

Robots are equipped with sensors and they are continuously perceptually aware. The frequency (sfreq) determines the temporal rate at which an agent wants to update its knowledge. The constraint $\mathrm{c_{sense}}$ ensures that the sensing event is initiated at that frequency. Robot has one sensor which is focused at some location. The sensor also has a reading. The constraint $\mathrm{c_{sloc}}$ ensures that the reading inherits the location at which it is made. Sensing is an event which involves applying the sfunc function which returns the reading.

$$\mathrm{s\text{: }sense(x,x.sensor_1)\ iff\ s\text{:}}\begin{bmatrix} \mathrm{x} & : & \mathit{Robot} \\ \mathrm{sfunc} & : & \mathit{RealVector}_2 \rightarrow \mathrm{Observation} \\ \mathrm{x.sensor_1.reading} & : & \mathrm{sfunc(x.sensor_1.focus)} \end{bmatrix}$$

$$Robot = \begin{bmatrix} \text{x} & : & \mathit{Ind} \\ \text{loc} & : & \mathit{RealVector}_2 \\ \text{sensor}_1 & : & \begin{bmatrix} \text{y} & : & \mathit{Ind} \\ \text{focus} & : & \mathit{RealVector}_2 \\ \text{reading} & : & \mathit{Observation} \\ \text{c}_{\text{sensor}} & : & \text{sensor(y)} \\ \text{c}_{\text{sloc}} & : & \text{equal(focus,reading.loc)} \end{bmatrix} \\ \text{sfreq} & : & \mathit{MSeconds} \\ \text{c}_{\text{robot}} & : & \text{robot(x)} \\ \text{c}_{\text{sense}} & : & \text{at(sense(x,x.sensor}_1\text{),sfreq)} \end{bmatrix}$$

The mapping between the perceptual and conceptual or language domains can only be learned through experience as a (probabilistic) classifier – it is unclear otherwise how one would specify a mapping from a continuous numeric domain to a symbolic domain with a sufficient level of complexity required to model human spatial cognition.[2] Having classifiers which are generalisations of distributional knowledge also allows us to account for the vagueness of spatial descriptions as we will discuss later. The mapping from *Observation* to *PointMap* in the SLAM procedure is an example of probabilistic classification. Subsequently, the mapping involves increasingly higher conceptual categories rather than individuals as points. TTR does not distinguish between symbolic and sub-symbolic domains that are common in the traditional AI approaches. Instead, we refer to low-level and high level types which contain conceptual or distributional information or both.

A perceptual classifier for spatial descriptions such as "to the left of" is a function from a pair of records of type *Region* to records of type *Left*. The mapping is further constrained by a classifier function $f$ which considers some values of labels on the input types and applies them to classifier knowledge $\pi$ which represents a theory of *Left* in geometric terms. Note also that we store the convex hulls of objects r.$\mathit{Region}_1$.c-hull and r.$\mathit{Region}_2$.c-hull as these represent proofs that that spatial relation holds for $\mathit{Region}_1$ and r.$\mathit{Region}_2$.

$$\lambda r\text{: } \langle \begin{bmatrix} \text{a}_1 & : & \mathit{Ind} \\ \text{c}_{\text{reg}} & : & \text{region(a}_1\text{)} \\ \vdots & & \\ \text{c-hull} & : & : \langle \text{p}_\text{i}, \text{p}_\text{j}, \text{p}_\text{k} \rangle \end{bmatrix}, \begin{bmatrix} \text{a}_2 & : & \mathit{Ind} \\ \text{c}_{\text{reg}} & : & \text{region(a}_2\text{)} \\ \vdots & & \\ \text{c-hull} & : & : \langle \text{p}_\text{i}, \text{p}_\text{j}, \text{p}_\text{k} \rangle \end{bmatrix} \rangle \ (\mathit{Left})$$

$$\mathit{Left} = \begin{bmatrix} \text{x}_1 & : & \text{r.Region}_1\text{.a} \\ \text{x}_2 & : & \text{r.}\mathit{Region}_2\text{.a} \\ \text{c-hull}_{\text{r1}} & : & \text{r.}\mathit{Region}_1\text{.c-hull} \\ \text{c-hull}_{\text{r2}} & : & \text{r.}\mathit{Region}_2\text{.c-hull} \\ \text{classifier} & : & \pi \\ \text{c}_{\text{left}} & : & f(\pi,\text{c-hull}_{\text{r1}},\text{c-hull}_{\text{r2}}) = \begin{cases} \text{left}(\text{x}_1,\text{x}_2) \vee \\ \neg\ \text{left}(\text{x}_1,\text{x}_2) \end{cases} \end{bmatrix}$$

[2] Spatial templates [21] capture such probabilistic knowledge as they represent for each cell the degree of applicability of a particular spatial expression.

Regions may also be sub-types of other record types: we need to assign them a name such as "chair" or "table" rather than just "region" using perceptual classification. Here we assume that this is accomplished only by considering the shape of the point cloud but typically in object identification and image recognition a richer sensory information is used.

*Region* $\sqsubseteq$ *Chair*

$$Chair = \begin{bmatrix} \mathrm{b} & : & Ind \\ \mathrm{p_1} & : & Point \\ \vdots & & \\ \mathrm{p_n} & : & Point \\ \mathrm{c_{reg}} & : & \mathrm{region(b)} \\ \mathrm{c_{chair}} & : & f(\pi, \langle\, \mathrm{p_1} \ldots \mathrm{p_n} \rangle) = \begin{cases} \mathrm{chair(b)} \vee \\ \neg\, \mathrm{chair(b)} \end{cases} \end{bmatrix}$$

The classifier knowledge $\pi$ represents a model that has been learned through observing a number of situations containing two regions matched by a linguistic description "left". In practical robotic applications any popular machine learning classifier could be used [31]. In [11] we used a decision tree learning method based on the ID3 algorithm and the Naïve Bayes method. Recently, the probabilistic Bayesian methods have become very popular in cognitive science as a model of human learning and cognitive development [3, 29]. Equally, for natural language semantics [19] argues that vague predicates like "tall" (and we may add spatial predicates like "left") are described by continuous probability functions.

We believe that Naïve Bayes classification demonstrates nicely how competition between different vague terms is resolved – when does something that is "tall" become "short" or "left" become "behind". In order to do that we need to change our estimates of probabilities slightly: rather than estimating $p(left)$ and $p(\neg left)$ we need to find $p(left)$, $p(right)$, $p(behind)\ldots$ for all spatial expressions in our language. The probability of each of these is estimated by the Bayesian rule.

$$\begin{aligned} p(v|a_1 \ldots a_i) = & \quad \frac{p(a_1, a_2 \ldots a_i|v)p(v)}{p(a_1, a_2 \ldots a_i)} \\ = & \quad p(v) \prod_i^n \frac{p(v|a_i)}{p(a_i)} \end{aligned}$$

where $v$ is the value ("left") and $a$ are classification attributes, for example distances and angles between the objects obtained from their convex hulls. In practice, it is assumed (naïvely) that the attributes are independent and hence the equation may be rewritten as in the second line. Crucially, in common Naïve Bayes classification we accept the maximum likelihood hypothesis ($v_{map}$): the $v$ with the highest $p(v|a_1 \ldots a_i)$ and ignore the other target classes. However, this

does not need to be the case. If the difference between two $p(v)$ is small and falls within some contextually determined interval, one could exploit the probabilistic knowledge to generate descriptions such as "the chair is to the left of the table but it's also behind it". In other words, the model can account for situations where more than one spatial description applies.

### 4.2 Objects as geometric conceptualisations

In a computational model the effect of different geometric conceptualisations of objects on the semantics of spatial descriptions becomes blatantly obvious. They are dependent on the sensors employed and the geometric models that can be constructed from such sensory information as demonstrated above. Different conceptualisations of objects are represented as different TTR types from which it follows that spatial relations also have multiple type representations. This is not only because different object representations feature as sub-types of spatial relations types but also because their classifier knowledge is different as it abstracts over different attributes: if objects are points, a classifier cannot take into account their shape and if they are regions, the classifier model ignores their height. In sum, a description such as "left" corresponds to different types each representing a different view or take on a situation.

### 4.3 World knowledge

The world knowledge about objects is expressed in types as propositional content. A situation denoted by "the umbrella is over the man" involves three individuals conceptualised as volumes: the umbrella ($x_1$), the man ($x_2$) and the rain ($x_3$). Note that the latter is not mentioned in the description. $x_1$ and $x_2$ have to be in a particular geometric arrangement determined by the classifier. There is an additional constraint that the individuals $x_3$, $x_2$ and $x_1$ are involved in a situation of type *Protects* = protects($x_1$, $x_2$, $x_3$). Since the geometric meaning constraint $c_{over}$ is determined by a probabilistic classifier, the acceptable deviations of the umbrella from the prototypical vertical upright position and their gradience are accounted for. Equally, the model predicts that a situation where a man holds an umbrella in the upright position and therefore the $c_{over}$ constraint is defined with high probability but the umbrella does not provide protection from the rain cannot have the denotation of the type *Over* since the constraint $c_{protects}$ is not satisfied.

$$\lambda r\text{: } \langle Vol_1,\ Vol_2,\ Vol_3 \rangle\text{:} \begin{bmatrix} x_1 & : & \text{r.}Vol_1\text{.a} \\ x_2 & : & \text{r.}Vol_2\text{.a} \\ x_3 & : & \text{r.}Vol_3\text{.a} \\ \text{c-hull}_{x_1} & : & \text{r.}Vol_1\text{.c-hull} \\ \text{c-hull}_{x_2} & : & \text{r.}Vol_2\text{.c-hull} \\ \text{classifier} & : & \pi \\ c_{over} & : & f(\pi,\text{c-hull}_{x_1},\text{c-hull}_{x_2}) = \begin{cases} \text{over}(x_1,x_2) \vee \\ \neg\ \text{over}(x_1,x_2) \end{cases} \\ c_{protects} & : & \text{protects}(x_3,x_2,x_1) \end{bmatrix}$$

We assume that the geometric constraints on the type *Protects* are defined conceptually but they could also be defined by a perceptual classifier or a combination of both. $x_1$ protects $x_2$ from $x_3$ if (i) we are able to draw a line through their centroids; (ii) we are able rotate them so that the line connecting their centroids becomes their positive vertical axis; and (iii) the 2-dimensional projections of the rotated objects (their *xy* dimensions or regions) overlap ($\subseteq$) in a particular way. Therefore, the conceptual world knowledge that we bring into the meaning representations of spatial descriptions may also depends on perceptual knowledge. Language is deeply embedded in perception. It is a way of expressing increasingly more complex types from simple perceptual types.

$$\lambda r{:}\ \langle Vol_3,\ Vol_2,\ Vol_1\rangle{:}\begin{bmatrix}
x_3 & : & r.Vol_3.a \\
x_2 & : & r.Vol_2.a \\
x_1 & : & r.Vol_1.a \\
c_{protects} & : & protects(x_3,x_2,x_1) \\
ce_{x_1} & : & centroid(x_1) \\
ce_{x_2} & : & centroid(x_2) \\
ce_{x_3} & : & centroid(x_3) \\
cline & : & :\ line(\langle ce_{x_1},ce_{x_2},ce_{x_3}\rangle) \\
rot_{x_1} & : & :\ rotate(x_1,x_1.z,cline) \\
rot_{x_2} & : & :\ rotate(x_2,x_2.z,cline) \\
rot_{x_3} & : & :\ rotate(x_3,x_3.z,cline) \\
reg_{x_1} & : & :\ xy_project(rot_{x_1}) \\
reg_{x_2} & : & :\ xy_project(rot_{x_2}) \\
reg_{x_3} & : & :\ xy_project(rot_{x_3}) \\
c_{overlap} & : & (reg_{x_3} \subseteq reg_{x_2} \subseteq reg_{x_1}) \vee \\
 & & (reg_{x_1} \subseteq reg_{x_2} \subseteq reg_{x_3})
\end{bmatrix}$$

An important question to address is how such world knowledge constraints are acquired in types such as *Over*. With humans they may be learned by corrective feedback and explicit definition through dialogue interaction (see [7] for learning ontological conceptual knowledge this way). This opens a possibility of dynamic types. In a computational framework one might explore domain theory learning described in [24]. For the remaining discussion we simply assume that we have a database of predicate assertions of such knowledge.

We discuss in Section 3 how world knowledge plays a greater role in the meaning representation of "over" than "above". This is accounted for in a probabilistic model as follows. In Section 4.1 we show that conceptual record types such as left($x_1$,$x_2$) and chair(x) are characterised by probabilistic knowledge. This is expressing a degree of belief that particular records of perceptual types are associated with a particular record of a conceptual type: or in other words that particular sensory observations map to a particular individual of the type *Chair*. The probabilistic knowledge is acquired by counting the observations events of records of appropriate type. The constraints that are a part of predicate types such as $c_{protects}$ in *Over* provide support for that type in the same way as perceptual observations do. The application of constraints $c_1$, $c_2$,... $c_n$ may be thought of as classification function that classifies for that type.

$$f(\pi,\mathrm{c}_1,\mathrm{c}_2,\ldots\mathrm{c}_\mathrm{n}) = \begin{cases} \mathit{Type} \vee \\ \neg\ \mathit{Type} \end{cases}$$

The domain of the function are records of the appropriate type and the target is a type. If we induce from the records that something is of some type then we can conclude that we have a record of this type. As a classifier the Bayes' rule may be applied.

$$T = \{Type, \neg Type\}$$

$$p(t_{map}) \equiv \arg\max_{t \in T} p(t) \prod_{i}^{n} \frac{p(c_i|t)}{p(c_i)}$$

For conceptual p-types $p(t)$ and $p(c)$ can be learned by counting the number of times a record of that type is encountered in our knowledge database over the total number of predicate assertions there. The arity of the p-type is important when examining the database but the nature of its arguments is not. The probability $p(c|t)$ is the number of times a particular p-type $c$ is encountered as a defining constraint for its super p-type $t$ in our knowledge base over the number of times that the p-type $c$ is used as a constraint in any type. The $\frac{p(c_i|t)}{p(c_i)}$ defines the support of $c$ for $t$. Its value will be maximum (1) if $c$ exclusively constrains $t$ and none of the other types.

For perceptual p-types $c$ the probability of a record belonging to this type is determined by the application of a probabilistic classifier as previously described. The probability $p(c|t)$ is determined by observing the number of times a positive classification ($c$ rather than $\neg c$) is used as a constraint for its super p-type $t$ over the number of times $c$ is used as a constraint in any type. Consider the case of the spatial descriptions "over" and "above". In the learning stage of the type *Over* one may observe assignments to this type where the geometric constraint is ignored in favour of the conceptual constraint. Consequently, the support of the geometric constraint $c_i$ for $t$ is lower and that of the conceptual constraint $c_j$ is higher. The reverse is the case for the type *Above*. Therefore, the probabilistic model correctly predicts the facts observed experimentally.

The preceding discussion shows that (i) perceptual and conceptual knowledge are learned separately from different sources – perceptual from sensory observations and conceptual from assertions in a knowledge database; and (ii) the integration events of perceptual with the conceptual knowledge must also be considered in the learning model as described in the preceding paragraph.

## 5 Conclusion and future work

In this paper we present a formal semantic model for natural language spatial descriptions in Type Theory with Records which can incorporate perceptual, conceptual and discourse constraints. In particular we discuss perceptual and conceptual constraints. Discourse may constrain the meaning of spatial descriptions in at least two ways. Firstly, the perceptual classifiers and conceptual knowledge may be updated dynamically as new language and perceptual knowledge is

acquired during a discussion of a scene [7, 17]. Secondly, in generating and interpreting projective spatial descriptions the conversation participants align their perspective or reference frame and keep this information as a discourse common ground [forthcoming]. Both kinds of discourse constraints can be modelled within the TTR framework.

We are also intending to implement our model computationally in a robotic agent. Here we foresee the following challenges: (i) how can we ensure the same kind of compositionality of perceptual meaning that we get with conceptual meaning; (ii) how do we learn the right kind of world knowledge relevant for a particular spatial description; and (iii) how does such world knowledge link to perception given that it has been learned from assertions and has never been experienced through sensory observations. To address the second issue we are exploring two strategies: (a) mining the knowledge about objects that occur with a particular spatial description from a large text corpora, and (b) using a variant of explanation based learning in an inquisitive dialogue agent that learns the conceptual constraints from a conversation with a human dialogue partner.

*Acknowledgements* This research was supported in part by the Semantic analysis of interaction and coordination in dialogue (SAICD) project, Vetenskapsrådet, 2009-1569, 2010–2012.

## References

1. Barwise, J.: The situation in logic. CSLI Publications, Stanford (1989)
2. Blackburn, P., Bos, J.: Representation and inference for natural language. A first course in computational semantics. CSLI (2005)
3. Clark, A., Lappin, S.: Linguistic nativism and the poverty of the stimulus. Wiley-Blackwell, Chichester, West Sussex (2011)
4. Cooper, R.: Austinian truth, attitudes and type theory. Research on Language and Computation 3, 333–362 (2005)
5. Cooper, R.: Records and record types in semantic theory. Journal of Logic and Computation 15(2), 99–112 (2005)
6. Cooper, R.: Type theory and semantics in flux. In: Kempson, R., Asher, N., Fernando, T. (eds.) Handbook of the Philosophy of Science, General editors: Dov M Gabbay, Paul Thagard and John Woods, vol. 14. Elsevier BV (2012)
7. Cooper, R., Larsson, S.: Compositional and ontological semantics in learning from corrective feedback and explicit definition. Proceedings of DiaHolmia 2009 Workshop on the Semantics and Pragmatics of Dialogue Stockholm Sweden pp. 59–66 (2009)
8. Coventry, K.R., Prat-Sala, M., Richards, L.: The interplay between geometry and function in the apprehension of Over, Under, Above and Below. Journal of memory and language 44(3), 376–398 (2001)
9. Davidson, D.: The logical form of action sentences. In: Rescher, N., Anderson, A.R. (eds.) The logic of decision and action. University of Pittsburgh Press, Pittsburgh (1967)
10. Dissanayake, M.W.M.G., Newman, P.M., Durrant-Whyte, H.F., Clark, S., Csorba, M.: A solution to the simultaneous localization and map building (SLAM) problem. IEEE Transactions on Robotic and Automation 17(3), 229–241 (2001)
11. Dobnik, S.: Teaching mobile robots to use spatial words. Ph.D. thesis, University of Oxford: Faculty of Linguistics, Philology and Phonetics and The Queen's College (September 2009)

12. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. IEEE Computer Society Conference on Computer Vision and Pattern Recognition pp. 1778–1785 (2009)
13. Gapp, K.P.: Basic meanings of spatial relations: Computation and evaluation in 3d space. In: Hayes-Roth, B., Korf, R.E. (eds.) AAAI. pp. 1393–1398. AAAI Press/The MIT Press (1994)
14. Ginzburg, J.: The interactive stance: meaning for conversation. Oxford University Press, Oxford (2012)
15. Guerra-Filho, G., Aloimonos, Y.: A language for human action. Computer 40(5), 42 –51 (may 2007)
16. Herskovits, A.: Language and spatial cognition: an interdisciplinary study of the prepositions in English. Cambridge University Press, Cambridge (1986)
17. Larsson, S.: The TTR perceptron: Dynamic perceptual meanings and semantic coordination. In: Artstein, R., Core, M., DeVault, D., Georgila, K., Kaiser, E., Stent, A. (eds.) SemDial 2011 (Los Angelogue): Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue. pp. 140–148. Los Angeles, California (September 21—23 2011)
18. Larsson, S., Cooper, R.: Towards a formal view of corrective feedback. In: Alishahi, A., Poibeau, T., Villavicencio, A. (eds.) Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition. pp. 1–9. EACL (2009)
19. Lassiter, D.: Vagueness as probabilistic linguistic knowledge. In: Proc. of the international conference on vagueness in communication (ViC'09). pp. 127–150. Springer-Verlag, Berlin, Heidelberg (2011)
20. Lauria, S., Bugmann, G., Kyriacou, T., Bos, J., Klein, E.: Training personal robots using natural language instruction. IEEE Intelligent Systems 16, 38–45 (September/October 2001)
21. Logan, G.D., Sadler, D.D.: A computational analysis of the apprehension of spatial relations. In: Bloom, P., Peterson, M.A., Nadel, L., Garrett, M.F. (eds.) Language and Space, pp. 493–530. MIT Press, Cambridge, MA (1996)
22. Matuszek, C., Herbst, E., Zettlemoyer, L., Fox, D.: Learning to parse natural language commands to a robot control system. In: Proc. of the 13th International Symposium on Experimental Robotics (ISER) (June 2012)
23. Miller, G.A., Johnson-Laird, P.N.: Language and perception. Cambridge University Press, Cambridge (1976)
24. Pulman, S.G., Liakata, M.: Learning domain theories. In: Nicolov, N., Bontcheva, K., Angelova, G., Mitkov, R. (eds.) RANLP. Current Issues in Linguistic Theory (CILT), vol. 260, pp. 29–44. John Benjamins, Amsterdam/Philadelphia (2003)
25. Regier, T., Carlson, L.A.: Grounding spatial language in perception: an empirical and computational investigation. Journal of Experimental Psychology: General 130(2), 273–298 (2001)
26. Roy, D.: Semiotic schemas: a framework for grounding language in action and perception. Artificial Intelligence 167(1–2), 170–205 (September 2005)
27. Siskind, J.M.: Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. Journal of Artificial Intelligence Research 15, 31–90 (2001)
28. Talmy, L.: Toward a cognitive semantics: concept structuring systems, vol. 1 and 2. MIT Press, Cambridge, Massachusetts (2000)
29. Tenenbaum, J.B., Kemp, C., Griffiths, T.L., Goodman, N.D.: How to grow a mind: Statistics, structure, and abstraction. Science 331(6022), 1279–1285 (2011)
30. Winograd, T.: Understanding Natural Language. Edinburgh University Press (1976)
31. Witten, I.H., Frank, E., Hall, M.A.: Data mining: practical machine learning tools and techniques. Morgan Kaufmann, Burlington, MA, 3rd edn. (2011)

# Ontology driven contextual reference resolution in Embodied Construction Grammar

Jesús Oliva[1], Jerome Feldman[2], Luca Gilardi[2], and Ellen Dodge[2]

[1] Bioengineering Group. Spanish National Research Council - CSIC. Carretera de Campo Real, km. 0,200. La Poveda CP: 28500. Madrid, Spain. jesus.oliva@csic.es
[2] International Computer Science Institute. 1947 Center Street Suite 600, Berkeley, CA 94704

**Abstract.** Constraint satisfaction has been central to the ICSI/UC Berkeley Neural Theory of Language (NTL) project, but this aspect has not previously been emphasized. In this paper we focus on a shallow reference resolution method that extends the Embodied Construction Grammar formalism (ECG). The approach is based on the combination of the recency principle with syntactic and semantic compatibility between the anaphora and the referent. The key to our method is the use of the ECG formalism: schemas, constructions and ontological knowledge. Within this framework we propose a method that, like humans, does not need very complex inferences in order to solve the basic reference resolution task. The method has been implemented and tested as part of a system capable of understanding Solitaire card-game instructions, with promising results.

## 1 Introduction

From a sufficiently general perspective, Constraint Satisfaction (CS) can be seen as one of the most fundamental processes in nature. A compact version of this story is depicted in Figure 1. and will be discussed further in the oral presentation. Most relevant here is the fact that language understanding, like all perception, involves constrained best-fit of the input to the context and goals of the perceiver. This has been understood for some time [1] and plays a central role in the analysis module of the ECG system for semantically driven natural language understanding, shown in Figure 2. The language input to the system is analyzed using the best-fit analyzer to produce a semantic representation called the *SemSpec* (see details below). Then, the Specializer tries to extract the task-relevant meaning from that structure and passes this information as N-tuples to the Application side.

One powerful example of best-fit CS in language understanding arises in languages, such as Mandarin, where almost any word can be omitted from an utterance if it is available form context. Partially to handle such situations, [2] built a Bayesian best-fit Discourse Analyzer (middle left of Figure 2) that can determine the best semantic analysis, even for quite sparse input, like the Mandarin equivalent of "give Auntie". The best-fit process combines three posterior

probability scores. The first is close to a conventional stochastic SFG. The second score is an estimate of the (deep) semantic compatibility of fillers for various constructional roles and the third score estimates the goodness of fit for contextual elements not explicitly mentioned.

**Constrained Best Fit in Nature**

| | inanimate | animate |
|---|---|---|
| physics | lowest energy state | |
| chemistry | molecular fit | |
| biology | | fitness, MEU neuroeconomics |
| vision | | threats, friends |
| language | | errors, NTL, OT |
| society, politics | | framing, compromise |

**Fig. 1.** The ubiquity of Constrained Best Fit

More generally, language understanding is highly context dependent. In particular, anaphors are constantly used in order to avoid unnecessary repetitions of particular words or structures. The meaning of many elements of each sentence and, by extension, the meaning of each sentence, depends on the meaning of previous utterances. Examples of this are pronouns (like *he*) or definite noun phrases (like *the boy*). The reference resolution task consists of linking these semantically undefined structures (the anaphor) to an entity previously found in the discourse (the antecedent) to which they refer. Therefore, reference resolution methods constitute a very important part of any language understanding system. That importance has attracted a lot of researchers from the beginnings of the field of natural language processing to the last years. However, perfect performance is still out of reach.

Many approaches have been tried for reference resolution[3]. Knowledge-based systems (from the first reference resolution methods [4, 5] to some recent approaches [6]), were the first approach but not the only one since they are very complex systems, difficult to build, and they lacked robustness. Heuristic systems [7, 8] tried to solve those problems using well designed heuristics to avoid the complexity of previous systems. Finally, machine learning systems reformu-

[3] See [3] for an introduction

lated the reference resolution task as a binary classification problem. An early approach of this kind was presented by [9] and was followed by many other researchers introducing variations on that former algorithm [10–12].

Despite this great amount of work, naïve reference resolution methods [5, 8] still have very good performance. These methods are based on selecting the most recent antecedent that is grammatically compatible with the anaphora. Our method is also based on combining anaphora referent compatibility with the recency principle (humans tend to select antecedent candidates attending to their recency in discourse [7]). However, we use a different kind of compatibility not restricted to grammatical features. We introduce some semantic features (such as the functionalities of the concepts referred to by the anaphora and the antecedent) in order to improve the performance of the method.

All this work is done using the framework of Embodied Construction Grammar (ECG) [1]. ECG is a formalism for representing linguistic knowledge in the form of construction-based grammars. This formalism allows us to transfer much of the work load of the reference resolution method to the design of the grammar and the ontology. The use of those structures extends the probabilistic best-fit analyzer implemented for ECG [2, 13]. In particular, the reference resolution method presented in this paper has been developed as a part of an ongoing project of the Neural Theory of Language (NTL) project with the objective of implementing a system that can follow instructions and synthesize actions and procedures in natural language. The two initial task domains are artificial agents in simulated robotics and card games. Specifically, for the card games domain, the goal is to develop a system that is able to understand published Solitaire game descriptions in order to play the game. Within this framework we implemented a resolution method that, like humans, does not need very complex inferences in order to solve the reference resolution task.

The structure of the paper is the following: section 2 gives a brief introduction to the Embodied Construction Grammar formalism. Sections 3 and 4 present the core components of our method: the ontology and the grammar. Finally, section 5 explains the reference resolution method with some explanatory examples and section 6 points the general conclusions and some future work.

## 2 Embodied Construction Grammar

Embodied Construction Grammar is a formalism for representing linguistic knowledge in the form of construction-based grammars that supports models of language understanding (see [14] or the Neural Theory of Language wiki (`http:\\ecgweb.pbworks.com`) for a more extensive review of ECG). ECG is the result of large efforts of the NTL group to give rise to a formalism based on many insights from cognitive sciences and construction-based theories of language and covers many empirical findings from linguistics, psychology and computational sciences.

There are two main components of construction grammars: schemas and constructions. Schemas are the basic unit of meaning while constructions represent

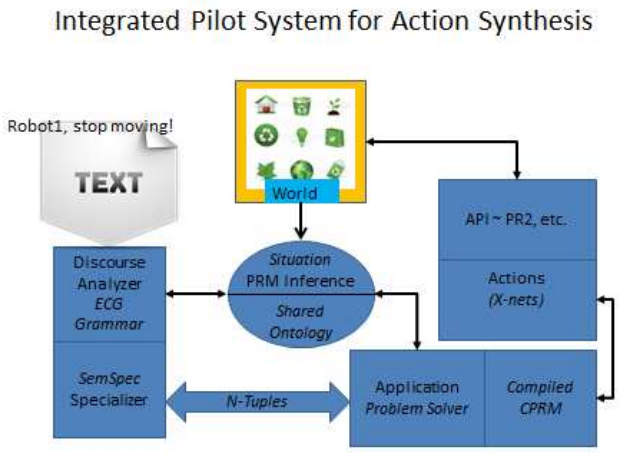



**Fig. 2.** Global system architecture

mappings between form and meaning. Schemas are formed by a list of components (roles) and the different constraints and bindings between these roles. Constructions have different constituents, and represent the form-meaning pairing with the correspondent bindings between the different constituents and the roles of their meaning schemas. Finally, schemas and constructions are not defined in isolation. They are hierarchically structured by *is-a* relations, supporting inheritance semantics along with multiple inheritance.

Figure 3 shows an example of ECG constructions and schemas. The *ActiveDitransitive* construction has three constituents, which are a two NPs and a Verb, v, (inherited from the ArgumentStructure construction by the subcase relation). The form block shows the ordering constraints among the constituents of the construction. In our case, it states that the constituent $v$ must appear in the sentence before *np1*, and *np1* before *np2*. The meaning of this construction is an *ObjectTransfer*, which is a subcase of *ComplexProcess* and has the roles shown on the right in figure 3. Constructions include a *constraints* block that imposes some bindings between the different constituents of the construction and the roles in its meaning schema. In this case, the *giver* is the *profiled participant* of the event and the *getter* and the *theme* are identified with the meaning of the first noun phrase (*np1*) and the second noun phrase (*np2*) respectively.

In addition to schemas and constructions, the ECG formalism makes use of an ontology that comprises general knowledge about the particular entities present in the discourse. As usual, the ontology is also hierarchically structured allowing inheritance between its elements. In our approach, we expand the typical entity-based ontology with a lattice of functional features that are domain dependent. We discuss the ontology used in the following section.

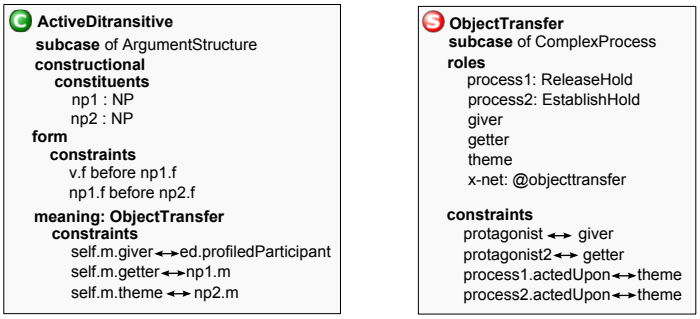



**Fig. 3.** Example of ECG constructions and schemas.

Using these structures, the Analyzer program [2, 13] produces a deep semantic representation (SemSpec) of the given sentences. The ECG analyzer uses the best-fit score, a metric using a combination of syntactic and semantic factors in order to produce the SemSpecs. Semantic specifications are networks formed by the bindings and unifications of the ontology items and schemas found in the meaning poles of the recognized constructions. The SemSpec captures the semantic and pragmatic information present on the input. SemSpecs are used in the simulation process in the ECG framework (cf. Figure 2). This simulation process is modeled by the x-nets (executing networks) which model events and its aspectual structure [15].

Some previous work has been done on reference resolution within the ECG formalism. For example, [16, 17] present a structured, dynamic context model incorporated in an ECG system for modeling child language learning. This context model is represented using ECG schemas in order to exploit the best-fit mechanisms of the analyzer. Basically, the majority of the workload of the reference resolution method is done by the analyzer using grammatical features (such as number, gender or case) and the ontological categories (when known) of the referent and the anaphora. The resolution process finds the possible antecedents that match the constraints imposed by the referent. This is a very shallow reference resolution mechanism (see [18] for a more complex best-fit reference resolution method) with some drawbacks and limitations such as the number of possible anaphors and antecedents considered by the method and the limited set of features.

## 3 Functional and entity ontology lattices

As stated in the introduction, the ontology and the grammar itself are the two core elements of the reference resolution method presented here. Any ontology comprises, in a hierarchical structure, general knowledge about entities and concepts present in the discourse. Our approach expands this general knowledge about entities and concepts by taking into account the functional properties of

those concepts. These properties can be domain dependent since the functionalities of each entity can differ depending on the domain they are used. For example, a column has different functionalities in architecture than in solitaire games. Therefore, the ontology has two main functions. It captures general knowledge about entities, concepts, and their functions and also it stores other facts related to the different senses of each of its elements depending on the particular domain, e.g. differentiating between a solitaire game column and an architectural column. The ontology is used by the grammar in order to constrain the roles in meaning schemas of the recognized constructions, reducing the possible schemas and helping the analysis process.

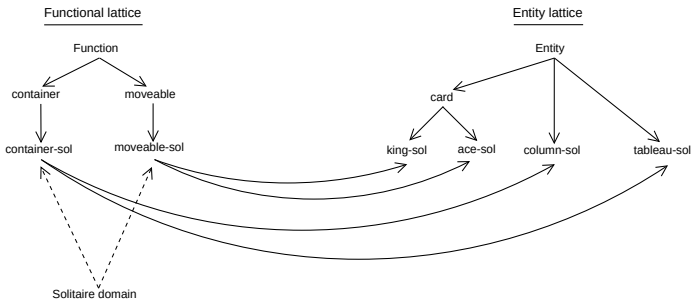



**Fig. 4.** Fragment of the entity and functional ontology lattices

The ontology used by our system is structured by two connected inheritance lattices (see figure 4):

- Entity lattice: entities are structured hierarchically using the typical *is-a* relations in order to specify the categories to which each of its elements belongs to.
- Functional lattice: this lattice of the ontology is a hierarchical structure of functions and properties of the elements in the *entity lattice*.

The two lattices are connected. A link between a concept in the entity lattice and a function in the functional lattice represent that the entity involved has the corresponding function. Moreover, all the relations intra- and inter-lattice support the usual inheritance semantics including multiple inheritance.

The third element of the ontology is the domain-specific information. This information is needed to distinguish the different senses of each word. For example, a column in the solitaire domain has the function of *container*. However, a column in an architectural domain does not have that function. And the same is applicable to the functional lattice: the *container* or *movable* functions can have different interpretations depending on the domain. So the different senses of each word have to be related to a specific domain (in figure 4, for example,

*colum-sol* inherits from *container-sol* which in turn, inherits directly from the *solitaire* domain).

## 4 Schemas, constructions and ontological constraints

As stated in the introduction, we avoid a very complex reference resolution algorithm using a proper design of the grammar constructions and schemas and constraining some roles to the functional categories described in the previous section.

Solitaire instructions usually describe card movements. In other words, typical sentences involve a trajector (a card, or rather, any *movable* entity) and a landmark (a position in the tableau, or rather, any *container*). See, for example, sentences like: "move aces to foundation spaces", "put cards on the waste" or "fill tableau spaces with kings". Given that all these sentences have the meaning of moving something movable to a container, we used a schema called *Update-sol* to express their meanings. That schema imposes that the landmark should have the ontological feature of being a *container-sol* and the trajector should be *movable-sol*. It is important to note that the *Update-sol* schema constrains those roles to functional categories and not to entity categories. In a simple solitaire game, we could think that only cards can be moved so we could constrain the *trajector* role to the *card* ontological category. However, in general, we can move many other things such as groups of cards, columns or piles. Thus, the roles should not be constrained to concrete entities but to functional categories.

In order to correctly analyze the instruction sentences, different argument structure constructions were built. All of them have in common that they have the *Update-sol* schema as their meaning pole. Each of the constituents of those argument structure constructions is bound to the corresponding role of the *Update-sol* schema. Therefore, the constituents inherit the functional constraints imposed by the *Update-sol* schema. This point is of crucial importance in the reference resolution method proposed here. The problem with many naïve methods is that they only look for compatibility of grammatical features and of ontological categories. However, there exist some undefined anaphors that do not have an ontological category (see, for example, pronouns like *it*). The ECG formalism allows us to define the functional categories of all the elements in a given argument structure. This way, although we can not state the entity-ontological category of pronouns, we can know their functional categories (i.e. we can know whether *it* should be a container or should be movable in this context). As we will see below, this distinction is of great importance in our reference resolution method.

Figure 5 shows one of the constructions used in the analysis of the sentence: "Fill tableau spaces with kings". The *Update-sol* schema transmits the functional constraints to the constituents of the argument structure. This way, we impose that the *patient* constituent of a *Fill-with* argument structure has to be a *container* and that the adverbial phrase has to be *movable*.

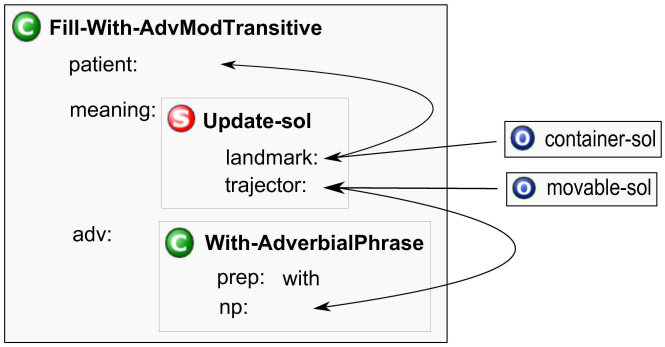



**Fig. 5.** Fragment of the constructions used to analyze sentences like: "Fill tableau spaces with kings".

## 5 Ontology driven reference resolution

Reference resolution is driven by a variety of constraints. Much work has been done on how best to combine those constraints in many sophisticated ways. However, we will focus only in two specific constraints. Syntactic constraints like agreement compatibility (i.e. number, gender or case compatibility) and semantic constraints like functional compatibility. Given the constructions and ontological structures presented in the two previous sections, our reference resolution process is quite simple. We exploited the probabilistic features of the best-fit analysis included in the ECG framework to restrict possibilities and based our method on the search of possible anaphors and antecedents and the comparison of their syntactic and semantic features.

As stated before, the analyzer builds a semantic specification as the meaning representation of the given sentence. A semantic specification is basically a network of the constructions that fit the structure of the analyzed sentence and the schemas and ontology items that fill the meaning poles of those constructions. Given the semantic specifications of the analyzed sentences, the resolution algorithm is straightforward. Our algorithm is based on syntactic and semantic compatibility between the anaphora and the antecedent and the recency principle. For each anaphor, it selects the most recent antecedent that shares its syntactic features (basically agreement features such as number, gender or case) and its semantic features (the functional roles) with the anaphor.

The resolution module goes through the semantic specifications and keeps track of the possible antecedents it finds. Possible antecedents could be: proper nouns or noun phrases in general. A list of the possible antecedents and their syntactic and semantic features is stored in inverse order (so the first element is the most recent antecedent). Once the module finds a possible anaphor (which is usually a pronoun or a noun phrase like "those cards"), it tries to resolve it. Given an anaphor, it goes through the list of possible antecedents and tries to find

one with the same syntactic features (number, gender) and the same functional category. The method returns the first antecedent candidate that matches. In other words, it chooses the most recent antecedent which shares the grammatical features and functional categories with the anaphora.

It is important to note that a given noun phrase could be, at the same time an anaphor and an antecedent. For example, in these sentences:

a) *Kings can be moved to an empty space.*
b) *Those cards can also be moved to a foundation space.*
c) *Moreover, they can be moved to another column.*

The noun phrase "those cards" in sentence b) is an anaphor for "Kings" and the antecedent of "they" in sentence c). Therefore, when our method find a noun phrase, it applies the two mechanisms: store the noun phrase as a possible antecedent and, if it is anaphoric, try to resolve it with the previous possible antecedents. This way, our method would find that "kings" is the antecedent of "those cards" and "those cards" is the antecedent of "they". Therefore, our method also establishes a link between "they" and "Kings" to capture the rule that "Kings" can be moved to another column.

In order to gain a better understanding of how our method works, we will walk through an example. Suppose that the following two sentences are found in a solitaire game description.

– *Move the top card of a column to the top of another column.*
– *Move it to a foundation pile.*

As mentioned in the previous section, *move*, imposes that its trajector must have the functional category of being movable. In 'constructional' terms, the *move* construction binds its trajector role with the functional category *movable-sol*. When the resolution binding is made, the pronoun *it*, which has no ontological category by default, takes the *movable-sol* category through unification. Then, the system goes through the list of possible antecedents comparing the morphological and functional features. The first element would be *column*, whose morphological features match up with the ones of the pronoun *it* (in this case, we just have to compare the grammatical number of the anaphora and the antecedent). However, their functional categories do not match since column has the *container-sol* functional category. Thus, the system continues to the next possible antecedent in the list, which is "*the top card of a column*". In this case, the morphological and functional categories are the same so the system establish that *it* refers to "*the top card of a column*". It is important to note that the inclusion of the semantic constraints in the reference resolution algorithm is crucial. A simple reference resolution method based only on the recency principle and syntactic compatibility would say that "column" is the antecedent for "it" in our example. However, the ECG formalism, allows us to require that "it" should be something movable and therefore, can not be linked to "column".

As stated before, this simple reference resolution method has been tested on a small solitaire domain with remarkably good performance. Obviously, it could

be improved in many ways but the use of deep semantics and domain-specific functional categories appears to a generally powerful tool in reference resolution. Current work is focused on more general semantic constraints such as affordances and materials.

## 6 Conclusions and future work

In this paper we present a reference resolution mechanism in the framework of ECG and its application to the understanding of card games instructions. The method exploits the features of the ECG formalism and the best-fit analyzer in order to avoid very complicated reference resolution approaches. Within this framework we built a method that, like humans, does not need very complex inferences in order to solve the basic reference resolution task. The method is based on the recency principle and grammatical and conceptual compatibility between the anaphor and the referent. In order to check the compatibility, we used agreement features (such as number, gender or case) and we also introduced some semantic features (such as the different functionalities of the concepts referred by the anaphor and the antecedent) in order to improve the performance of the method. All this work gave rise to a very efficient and also accurate method that has been tested as a part of a prototype system that tries to understand any solitaire game description well enough to play the game.

These results, along with earlier ECG uses of constraint satisfaction methods, are promising as part of systems for language understanding based on deep, embodied semantics. Current efforts are focused on natural language instruction of robots and on general metaphor understanding. Referring back to Figure 2, the reference resolution techniques described above are part of the *Specializer*, shown in lower left of the Figure. For any Application, the role of the Specializer is to convert the general semantic analysis given as the SemSpec into task specific N-tuples that convey the information needed by the Application. This obviously includes determining any references resolvable by discourse and situational context, as discussed in this paper. Unfortunately, the Specializer itself is the one module of the system that we have not been able to specify non-procedurally and we view this as a major open problem for the constraint satisfaction approach.

## Acknowledgments

This work was partly funded by a JAE Predoctoral Fellowship from CSIC (Spain). We are thankful to the International Computer Science Institute (University of California at Berkeley) and, in particular, to the people on the Neural Theory of Language project.

## 7 References

## References


1. Feldman, J.: From Molecule to Metaphor: A Neural Theory of Language. MIT Press (2006)
2. Bryant, J.: Best-Fit constructional analysis. PhD thesis, UC Berkeley, Berkeley,CA (2008)
3. Kehler, A.: Coherence, reference and the theory of grammar. CSLI Publications, Stanford, CA (2002)
4. Hobbs, J.: Pronoun resolution. Technical report, CUNY (1976)
5. Hobbs, J.: Resolving pronoun references. Lingua **44** (1978) 311–338
6. Asher, N., Lascarides, A.: Logics of conversation. Cambridge University Press, Cambridge, UK (2003)
7. Lappin, S., Leass, H.: An algorithm for pronominal anaphora resolution. Computational Linguistics **4**(20) (1994) 535–561
8. Mitkov, R.: Robust pronoun resolution with limited knowledge. In: Proc. COLIN/ACL'98. (1998) 869–875
9. Soon, W., Ng, H., Lim, D.: A machine learning approach to coreference resolution of noun phrases. Computational Linguistics **4**(45) (2001) 521 – 544
10. Ponzetto, S., Strube, M.: Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In: Proceedings of the HLT 2006, New York City (2006) 192 – 199
11. Ng, V., Cardie, C.: Improving machine learning approaches to coreference resolution. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. (2002) 104 – 111
12. Ng, V.: Semantic class induction and coreference resolution. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07). (2007)
13. Bryant, J., Gilardi, L.: A cognitive model of sentence interpretation. In HC, B., ed.: Computational approaches to embodied construction grammar, San Diego, John Benjamins (2011)
14. Feldman, J., Dodge, E., Bryant, J.: Embodied construction grammar. In Heine B, N.H., ed.: The Oxford handbook of linguistic analysis, Oxford, Oxford University Press (2009) 111 – 138
15. Narayanan, S.: KARMA: Knowledge-based action representations for metaphor and aspect. PhD thesis, UC Berkeley, Berkeley, CA (1997)
16. Chang, N., Mok, E.: A structured context model for grammar learning. In: Proceedings of the 2006 International Joint Conference on Neural Networks, Vancouver, BC (2006)
17. Mok, E.: Contextual bootstrapping for grammar learning. PhD thesis, UC Berkeley, Berkeley,CA (2009)
18. Poon, H., Domingos, P.: Joint unsupervised coreference resolution with markov logic. In: Proceedings of the 2008 conference on empirical methods in natural language processing (EMNLP), Honolulu, Hawaii (2008) 650–659

# Resolving relative time expressions in Dutch text with Constraint Handling Rules

Matje van de Camp[1] and Henning Christiansen[2]

[1] Tilburg Centre for Cognition and Communication
Tilburg University, The Netherlands
E-mail: M.M.v.d.Camp@uvt.nl
[2] Research group PLIS: Programming, Logic and Intelligent Systems
Department of Communication, Business and Information Technologies
Roskilde University, Denmark
E-mail: henning@ruc.dk

**Abstract.** It is demonstrated how Constraint Handling Rules can be applied for resolution of indirect and relative time expressions in text as part of a shallow analysis, following a specialized tagging phase. A method is currently under development, optimized for a particular corpus of historical biographies related to the development of the Dutch social movement between 1870 and 1940. It appears that CHR provides a modular language which is well-suited for tailoring a method optimized for the different linguistic constructs and the textual conventions applied in a specific corpus. We explain the principles and illustrate by sample rules. The project is currently in a phase of development and testing, and this paper focuses on the methodological issues, while a detailed evaluation is not relevant at present. Identified problems and possible extensions are discussed.

## 1 Introduction

Information regarding the social networks that underlie our society's history can provide new insights into known data for social historical research. Automatically extracting such a network from text involves finding and identifying the real world entities that are described in the text, such as persons, organizations, and locations. In order to connect the entity mentions to a timeline so that the evolution of the network can be examined, time expressions need to be detected, normalized, and put in their chronological order.

In this research, we study the problem of resolving the chronological ordering of time expressions found in a collection of biographies. Solving this task will facilitate the identification of events and relationships that involve two or more entities, including the time when the events took place and how long they lasted. Another point of interest is the positioning in time of general events and states of affairs.

In (historical) biographical texts, time may be indicated explicitly ("*June 12, 1876*","*in 1903*") or in a variety of indirect ways ("*on that day*","*May of*

*that year*"), depending on the writing style of the author and the availability of specific information. We are especially interested in resolving implicit time expressions and representing them as (perhaps imprecise) knowledge.

We use constraint solving techniques for this task, but instead of using a fixed constraint solver (e.g., a traditional finite domain solver) we use the language of Constraint Handling Rules (CHR). With this, we can write a constraint solver that incorporates linguistic knowledge and heuristics, including syntactic, semantic and pragmatic information extracted from the text.

Section 2 gives a short account on related work. Section 3 provides background on the chosen corpus and explains the goals for the analysis. Section 4 gives a gentle introduction to CHR and its application to language analysis. The preprocessing is briefly described in Section 5, and Section 6 explains some of the CHR rules that we have designed for time expressions. Considerations on tests and the current state of the implementation are given in Section 7, and finally we give some conclusions and indicate possible improvements of this technique.

## 2 Related work

Most standard methods for extraction of temporal information from text rely on machine learning, as exemplified by [1, 2]; see also [3] for an overview and more references. In our approach, we use a rule-based and transparent approach to tailor an extraction method optimized for a particular corpus in order to complete the suggestions produced by a rule-based tagger as the one explained in Section 5.

Applications of CHR for language processing have been demonstrated by different authors for a variety of tasks, such as parsing from Property Grammars [4], building UML diagram from use case text [5, 6], analyzing biological sequences [7, 8] and Chinese Word Segmentation [9]. The technique of using CHR for bottom up parsing is investigated in depth in [10] that also suggests a powerful notation of CHR Grammars on top of CHR; this notation was not used for the present work as CHR Grammars index only by positions in the text, whereas we apply an indexing that distinguishes between individual documents, paragraphs and sentences as well.

Temporal logic has been implemented in CHR [11] but we find that this, as well as other fully fledged calculi or logics concerning time, is far more general and computationally complex than we need here. A generic language processing system using typed feature structures implemented in CHR is described by [12] that demonstrates the use of event calculus for handling time. Instead, we have developed a constraint solver that reflects the particular linguistic constructions concerning time within our chosen corpus and their inherent, varying degrees of imprecision. The problem here does not seem not to be to reason about time, but to reason about context and discourse in natural language text, and here CHR provides a relevant tool [13]. We have not seen earlier work on CHR applied for extracting temporal information from text in a similar way.

## 3 The case study: Historical biographies in Dutch

The data used in this study is a collection of biographies, in Dutch, detailing the lives of the 575 most notable actors related to the rise of socialism in the Netherlands (± 1870–1940). The biographies describe a coherent group of people and organizations, with many connections between them.

The data set is provided to us by the International Institute of Social History (IISH) in Amsterdam.[3] IISH hosts the biographies online.[4] The documents are presented as separate pages, accessible either through an alphabetized index of person names, or via keyword search using simple Boolean operators. Links between documents are minimal: only the first mention of a person who also has a biography in the collection links to that biography, other entities are not linked. The biographies are accompanied by a database that holds personal information regarding the 575 main characters. It includes their first names, surname, dates of birth and death, and a short description. These details are used to generate the index on the website.

The overall goal is to enhance the biographies in such a way, that new entry points are created into the data, connecting all relevant pieces, and allowing for more serendipitous research to be performed on the data set. We believe that by interpreting the entity mentions as a co-occurrence network, which includes entities of types organization and location, and attaching it to a timeline, new perspectives are opened up that are not necessarily person-centric, as is the case now.

Part of the overall task is to identify how certain phenomena are related to time, e.g., particular events that are related to a specific date or entity attributes that hold for some period of time. The data contains time expressions of various types and formats. We restrict ourselves to resolving dates and intervals spanning days, months, or years. Time expressions with a finer granularity are not considered.

Globally, two types of temporal expressions are found in the biographies: specific dates, where day, month, and year are all known, and non-specific dates, where at least one of the attributes is unknown. Besides the distinction between specific and non-specific, another form of ambiguity exists in the expression of time in text. Dates can be given explicitly, as for instance “14 November 1879”, or implicitly, as “14 November of that year” or even “on that day”. In the latter case, the time expression can only be resolved in reference to one or more (non-) specific dates mentioned elsewhere in the text. Also, for each instance, we need to determine if it concerns a reference to a single point in time, or if it denotes the start or end of (or even some point during) an interval.

Ultimately, we aim to represent all explicit and implicit references to specific and non-specific dates or pairs thereof as points and intervals anchored to the timeline underlying all biographies, where we leave room for imprecisely defined dates.

[3] http://www.iisg.nl/
[4] http://www.iisg.nl/bwsa/

## 4 Constraint Handling Rules and text analysis

Constraint Handling Rules [14, 11], for short CHR, is an extension to Prolog that introduces bottom-up, forward chaining computations. Operationally, CHR is defined as rewriting rules over a constraint store, which can be seen as a global resource to be used by a Prolog program for storing, manipulating and consulting different hypotheses.

CHR inherits the basic nomenclature and syntactic conventions of Prolog such as variables written with capital letters and terms as a generic representation. The constraints of CHR can be called from Prolog programs and whenever a constraint is called, it is added to the constraint store, and the CHR rules supplied by the programmer apply as long as possible. CHR provides the following sorts of rules.

$$\text{Simplification rules: } c_1, \ldots, c_n \texttt{ <=> } \mathit{Guard} \mid c_{n+1}, \ldots, c_m$$
$$\text{Propagation rules: } c_1, \ldots, c_n \texttt{ ==> } \mathit{Guard} \mid c_{n+1}, \ldots, c_m$$

The $c$'s are atoms that represent constraints, possibly including variables. What is to the left of the arrow symbols is called the *head*, and what is to the right of the guard the *body*.

A simplification rule works by replacing in the constraint store, a possible set of constraints that matches the pattern given by $c_1, \ldots c_n$, with the constraints given by $c_{n+1}, \ldots, c_m$, however, only if the condition given by *Guard* holds. A propagation rule executes in a similar way, but it does not remove the head constraints from the store. The declarative semantics is hinted by the applied arrow symbols (bi-implication, resp., implication formulas, with variables assumed to be universally quantified) and it can be shown that the indicated procedural semantics agrees with this.

CHR provides a third kind of rules, called *simpagation rules*, which can be thought of as a combination of the two or, alternatively, as an abbreviation for a specific form of simplification rules.

$$\text{Simpagation rules: } c_1, \ldots, c_i \setminus c_{i+1}, \ldots c_n \texttt{ <=> } \mathit{Guard} \mid c_{n+1}, \ldots, c_m$$
$$\text{which can be though of as: } c_1, \ldots, c_n \texttt{ <=> } \mathit{Guard} \mid c_1, \ldots, c_i, c_{n+1}, \ldots, c_m$$

When this rule is applied, $c_1, \ldots, c_i$ stay in the constraint store and $c_{i+1}, \ldots, c_n$ are removed. In practice, the body of a rule can be any executable Prolog code.

It is straightforward to write bottom-up parsers in CHR when the strings to be analyzed, as well as recognized phrases, are represented as constraints with their position in the string represented by constraints carrying word boundaries as attributes. The following sample rule could be part of a simplistic natural language parser.

```
nounPhrase(N0,N1), verbPhrase(N1,N2) ==> sentence(N0,N2).
```

Informally it reads: if a noun phrase has been recognized between positions `N0` and `N1`, and a verb phrase between `N1` and `N2`, then we recognize a sentence

between `N0` and `N2`. These `N`$x$ variables are assigned natural numbers that represents a numbering of the spaces between the words. It is possible to extend this principle to model various context sensitive dependencies, as rules can also refer to non-consecutive elements and other constraints that represent semantic and pragmatic information. Rules can describe both shallow and deep analyses and flexible combinations thereof, and it is also possible to control in detail, the order in which rules are applied using special constraints that serve as triggers.

## 5 The starting point: Tagged text

The data is preprocessed in a number of ways. First, the text is lemmatized and tagged with part-of-speech information using Frog, a morpho-syntactic parser for Dutch [15]. Next, proper nouns are automatically identified and classified into types person, organization, and location, using the Stanford Named Entity Recognizer [16], which we have retrained for Dutch named entities. The entity mentions are then clustered to separate the distinct real world entities that they refer to. Finally, temporal expressions are identified and normalized using HeidelTime [17], a rule-based temporal tagger. An in depth analysis of the preprocessing stage will be given in a forthcoming article. For now, we briefly expand on the detection and normalization of temporal expressions.

### 5.1 Normalization of temporal expressions

HeidelTime was originally developed for German and English by Strötgen & Gertz (2012) [17]. Because the system is rule-based, its functionality can easily be extended by adding new sets of rules. We have demonstrated so by adding a rule set for the extraction and normalization of Dutch time expressions. HeidelTime extracts the temporal expressions by matching them and their immediate context to predefined patterns of regular expressions and converting them to a uniform format (TimeML[5]), making it easier to reason over them.

HeidelTime recognizes four types of temporal expressions: *date*, *time*, *duration*, and *set* [18]. We consider only types *date* and *duration* for this experiment. Both include normalization rules for expressions with various degrees of specificity. For example, "halverwege de jaren negentig" (mid 90's) is initially normalized to `'XX95-XX-XX'`, where day, month, and century are left undefined. A single mention of "juni" (June) is normalized to `'XXXX-06-XX'`. In a next step, HeidelTime applies some basic reasoning to these cases, trying to resolve them by relating them to either the document creation time, or the time expressions that have occurred previously in the text. If these do not provide sufficient information, the system will look at linguistic cues, such as verb tense, to determine how the expression is related to its context.

However, this is not a guaranteed method, especially not when dealing with historic documents. The document creation time in this case cannot be trusted,

[5] http://www.timeml.org

since there is an obvious gap between the creation time and the time when the described events occurred. `'XX95-XX-XX'` might mistakenly be normalized to `'1995-XX-XX'`, while in light of the domain under consideration, it is far more likely to refer to the year 1895.

A downfall of adhering to the TimeML format is that some information is lost in translation. A specific mention of the year "95" is normalized to the exact same tag as the example above, losing the distinction between a more or less exact point in time and a slightly more ambiguous version thereof. As the ambiguity of the expression increases, so does the severity of the information loss. Expressions such as "een paar dagen later" (a few days later) or "enkele jaren daarna" (several years later), are normalized to descriptive, imprecise tags. In this case both expressions are normalized to `'FUT_REF'`, which does not reflect the most important difference: their unit of calculation. This may not form such a huge problem in the analysis of, for instance, news articles, where the described events are likely to span only a short period. For the current task of resolving and *ordering* time expressions that span decades, however, this information is crucial.

### 5.2 From tags to constraints

After preprocessing, the tagged text is converted into constraints of the following form, one for each recognized token in a text:

`token(`*Doc*`,`*Par*`,`*Sent*`,`*WordStart*`,`*WordEnd*`,`
*NamedEntity*`,`*TimeML*`,`*TimeId*`,`
*Token*`,`*PosTag*`,`*Lemma*`)`

The *WordStart* and *WordEnd* attributes are word boundaries, as explained above, relative to a specific sentence within a specific paragraph of a specific document (i.e., a biography) as indicated by the first three attributes. The attributes *Token*, *PosTag*, and *Lemma* indicate a token, its part-of-speech tag, and its stem. The actual tense, mood, etc., are ignored since we are only interested in extracting specific pieces of information and not a full syntactic and semantic analysis of the text. The *NamedEntity* attribute represents whether the token is part of a named entity and, if so, which type of entity. The *TimeML* attribute contains the output received from HeidelTime, if the token is part of a temporal expression. The *TimeId* attribute contains, when relevant, the TimeML representation converted to a Prolog term of the form `date(`*Century*`,`*Decade*`,`*Year*`,`*Month*`,`*Day*`)`, using a `null` value to represent components that are not specified.

Since the biographies have already been processed automatically, we may expect occasional parsing or tagging errors. In the present experiment, we take the processed data as they are.

## 6 Examples of CHR rules for resolving time expressions

We are interested in identifying expressions that refer – either explicitly or indirectly through ana- or even kataphora of different sorts – to time points and

time periods (or intervals), and resolving them as reliably as possible to actual time. The relationship between these time expressions and particular events or states of affairs mentioned in the surrounding text is a different issue that we do not consider in the present paper.

Biographies have some specific features that we can anticipate and rely on. For time expressions, we may in some cases encounter a degree of uncertainty, either due to lack of precise information, or because the biographer judges the exact date to be irrelevant. On the other hand, we can expect that there is real history underneath, i.e., references to real time points and intervals. This implies that we can count on some general properties of time, such as temporal linearity, which might not exist in other genres of text, such as fiction.

All temporal expressions could be translated into constraints that treat time as points on a numerical axis and apply standard constraint solving techniques for equations and inequalities. However, this would result in the same loss of information that we see in the normalization to TimeML provided by HeidelTime. To bypass this problem, we find it more fitting to use constraints that reflect the linguistic expressions retaining their uncertainty. For example, to represent the meaning of *"in de late jaren zeventig"* (in the late seventies) it seems inappropriate to select an interval with hard, exact boundaries, for instance, from January 1, 1876, until New Year's eve 1879. A more intuitive solution is to replace the exact start date with a fuzzy representation. Finding the best way to do this is an issue for future research. Here we show some rules that are used in ongoing experiments.

### 6.1 Explicit dates and intervals

The tagger has already done an effort to recognize exact dates. The following rule converts an exact date recognized in the text into a single, more manageable constraint. Notice that the tagger via the *NamedEntity* attribute also indicates a relative position of each token in the assumed time expression, so that `'B-TIME'` indicates the beginning of a time expression, and `'I-TIME'` any immediately following tokens that are part of the same expression.

```
token(Bio,P,S,N0,N1,'B-TIME',_,date(C,De,Y,M,Da),_,'TW',Da),
token(Bio,P,S,N1,N2,'I-TIME',_,date(C,De,Y,M,Da),_,'SPEC',MonthName),
token(Bio,P,S,N2,N3,'I-TIME',_,date(C,De,Y,M,Da),_,'TW',_)
   ==> isPossibleDate(C,De,Y,M,Da), isMonthName(MonthName,M)
   | exactDate(Bio,P,S,N0,N3,date(C,De,Y,M,Da)).
```

It matches `token` constraints originating from a text such as *"12 februari 1889"*. In this rule we also perform a check that ensures the output from the tagger is valid, although this is fairly trivial for exact dates. The `isPossibleDate` and `isMonthName` predicates in the guard are part of a small lexical toolbox implemented for this application. The `'TW'` indicates a numeral.

An explicit year which is not part of a date is extracted as follows.

```
token(Bio,P,S,N1,N2,'B-TIME',_,Y,'TW',Year)
  ==> isPossibleExactYear(Year,C,De,Y)
   | exactYear(Bio,P,S,N1,N2,date(C,De,Y,null,null)).
```

The `isPossibleExactYear` predicate checks that the observed number, for example 1889, is a reasonable year and splits it into century, decade and year, for the example 18, 8, 9. The `null` values for month and date indicate that it is not relevant (at this level of analysis) to insert further information, as it is not recognized whether this mention of a year is intended as some exact, but not specified, date in that year, or a time interval spanning most of that year.

We have similar rules for identifying exact months in a given year and months without an explicitly given year. The latter are expected to be resolved using their context in a later phase of the analysis.

Exact time intervals are indicated in different ways in Dutch, one of the most common is to use the word *tot* (until) between two exact dates. This gives rise to the following CHR rule. In a simpagation rule like the following, the order of the constituents in the rule do not match the textual order, which is controlled by the `N`$x$ attributes.

```
token(Bio,P,S,N1,N2,'O',_,null,tot,'VZ',tot)
\
exactDate(Bio,P,S,N0,N1,date(C1,D1,Y1,M1,D1)),
exactDate(Bio,P,S,N2,N3,date(C2,D2,Y2,M2,D2))
   <=>
     timeInterval(Bio,P,S,N0,N3, date(C1,D1,Y1,M1,D1),
                                 date(C2,D2,Y2,M2,D2)).
```

Notice that the rule removes the exact dates from the constraint store as they are no longer relevant after this pattern has been applied.

### 6.2 Indirect time points

We often encounter time expressions where part of it is not expressed directly. There may be a pronoun as in *"september van dat jaar"*, or simply *"september"*. The following rule attempts to identify a relevant year for such cases.

```
token(Bio,P,S,N1,N2,'O',_,null,dat,'VG',dat),
token(Bio,P,S,N2,N3,'O',_,null,jaar,'N',jaar)
\
month(Bio,P,S,N0,N1,date(null,null,null,M,null))
   <=> nearestBeforeYear(Bio,P,S,N0, NBY),
       isPossibleExactYear(NBY,C,De,Y)
    |
       month(Bio,P,S,N0,N3,date(C,De,Y,M,null)).
```

The `nearestBeforeYear` device is a special constraint that searches backwards through the text to match the first constraint that contains a reasonable year.

This constraint is then supplied to the new `month` constraint that replaces the one matched in the head.

When the nearest, previous mention of a year is part of a reference to literature, the system might make a wrong guess. To correct this, a preprocessing phase neutralizes such citation years before we apply the normalization rules. This phase is implemented using CHR rules that match the conventions for references used in this corpus.

### 6.3 Time intervals with partly implicit information

Consider the fragment *"... mei tot september 1904"* (May until September 1904), for which it seems reasonable to assume that *"mei"* refers to *"mei 1904"*. For this particular pattern, we implement the following rule.

```
token(Bio,P,S,N1,N2,'O',_,null,tot,'VZ',tot)
\
month(Bio,P,S,N0,N1,date(null,null,null,M1,null)),
month(Bio,P,S,N2,N3,date(C2,De2,Y2,M2,null))
   <=>
     timeInterval(Bio,P,S,N0,N3, date(C2,De2,Y2,M1,null),
                                 date(C2,De2,Y2,M2,null)).
```

However, this rule will be incorrect if applied to *"december tot september 1904"*, where (at least in the lack of other strong indicators) *"december"* most likely refers to *"december 1903"*. There are yet other cases where the year is given for the first and not the last given month and so on. We could write a set of almost identical CHR rules for the different cases, but instead it is more convenient to write one general rule as follows (thus also subsuming the rule above).

```
token(Bio,P,S,N1,N2,'O',_,null,tot,'VZ',tot)
\
month(Bio,P,S,N0,N1,date(C1,De1,Y1,M1,null)),
month(Bio,P,S,N2,N3,date(C2,De2,Y2,M2,null))
   <=> canResolveMonthInterval(C1,De1,Y1,M1, C2,De2,Y2,M2,
                               C1r,De1r,Y1r,M1r, C2r,De2r,Y2r,M2r)
    |  timeInterval(Bio,P,S,N0,N3, date(C1r,De1r,Y1r,M1r),
                                    date(C2r,De2r,Y2r,M2r)).
```

The predicate `canResolveMonthInterval` in the guard is a Prolog predicate that covers all the different cases of known and unknown values. For the example *"december 1903 tot mei"*, the variables `C2r`, `De2r`, `Y2r` would be instantiated to indicate the year 1904. The guard fails (which prevents the rule from being applied) when there is no sensible way of filling in the missing information.

### 6.4 Open intervals

Biographical text often indicates time intervals by giving only a start or an end time, where the missing point is either obvious according to the timeline or

not considered important. Following the recognition of closed intervals, we can define rules that resolve expressions such as *"tot september 1904"*, when it is not immediately preceded by another time expression.

```
token(Bio,P,S,N1,N2,'O',_,null,tot,'VZ',tot)
\
exactDate(Bio,P,S,N2,N3,date(C2,De2,Y2,M2,Da2))
   <=>
       timeInterval(Bio,P,S,N1,N3, DATE, date(C2,De2,Y2,M2,Da2)),
       DATE <=< date(Y2,M2,D2).
```

The symbol `<=<` is a constraint that represents time order. Current experiments investigate how to optimally combine such basic algebraic constraints with the principle of searching in the surrounding text for a reasonable start or end time.

## 7 Expected results and considerations about test

The method presented so far allows us to introduce new rules optimized for the chosen corpus, where, in some cases, the rules are so specific that they are only applied once. These tailored rules allow us to obtain maximal recall and precision measures on the given corpus.

However, this conclusion is not very interesting in itself, and we still need to design tests that provide a fair measurement for comparison with related methods. This could be based on a traditional splitting between training and validation, and allowing only rules of a certain generality, measured by the number of times each rule is applied to the training set. The rule set exemplified above is still under development and not yet at a stage that justifies a proper evaluation.

However, some initial tests have shown incorrect results caused by the CHR rules, for instance, when they fail to recognize a time expression reported by the temporal tagger. A possible cause of this problem could be the fact that the tagger was developed independently before the use of CHR was considered. To get the best out of the combination of a smart tagger and CHR matching, there needs to be a clearer separation of the tasks performed at each level and, thus, a more consistent interface between the two levels.

## 8 Conclusions

An approach to extract and resolve time expressions from natural language text is proposed. Time in natural language is often expressed in indirect and/or underspecified ways that reflect lack of exact information. We suggest a rule- and constraint-based approach that can be tailored carefully to the pragmatics of a particular class of texts, here exemplified by interrelated personal biographies written in Dutch concerning a given historical period and context. We start from text that has been processed by a fairly advanced, but not flawless, temporal

tagger. Our rules can both rely on and correct for consistent mistakes in the tagging.

We expect that the principle described here can be developed into a general method for resolution of time expressions, that can be applied to many specific cases in a modular way, and provides a flexibility for long distance references in any direction.

The explicit manipulation of `null` values for the resolution of implicit information, as described above, is only an intermediate solution. It seems more appropriate to represent unspecified time values as constrained logical variables and develop a full constraint solver to handle the relevant collection of temporal representations, thus resolving unknown values in an automatic way. This will reduce the need for explicit search for missing information, as the inference from context is done via relationships between time expressions, thus providing a symmetrical flow of information forwards, or backwards, through the text.

## References


1. de Medeiros Caseli, H., Villavicencio, A., Teixeira, A.J.S., Perdigão, F., eds.: Computational Processing of the Portuguese Language - 10th International Conference, PROPOR 2012, Coimbra, Portugal, April 17-20, 2012. Proceedings. In de Medeiros Caseli, H., Villavicencio, A., Teixeira, A.J.S., Perdigão, F., eds.: PROPOR. Volume 7243 of Lecture Notes in Computer Science., Springer (2012)
2. Hovy, D., Fan, J., Gliozzo, A.M., Patwardhan, S., Welty, C.A.: When did that happen? - linking events and relations to timestamps. In Daelemans, W., Lapata, M., Màrquez, L., eds.: EACL, The Association for Computer Linguistics (2012) 185–193
3. Verhagen, M., Gaizauskas, R.J., Schilder, F., Hepple, M., Moszkowicz, J., Pustejovsky, J.: The tempeval challenge: identifying temporal relations in text. Language Resources and Evaluation **43**(2) (2009) 161–179
4. Dahl, V., Blache, P.: Implantation de grammaires de propriétés en chr. In Mesnard, F., ed.: JFPLC, Hermes (2004)
5. Christiansen, H., Have, C.T., Tveitane, K.: From use cases to UML class diagrams using logic grammars and constraints. In: RANLP '07: Proc. Intl. Conf. Recent Adv. Nat. Lang. Processing. (2007) 128–132
6. Christiansen, H., Have, C.T., Tveitane, K.: Reasoning about use cases using logic grammars and constraints. In: CSLP '07: Proc. 4th Intl. Workshop on Constraints and Language Processing. Volume 113 of Roskilde University Computer Science Research Report. (2007) 40–52
7. Bavarian, M., Dahl, V.: Constraint based methods for biological sequence analysis. J. UCS **12**(11) (2006) 1500–1520
8. Dahl, V., Gu, B.: A CHRG analysis of ambiguity in biological texts. In: CSLP '07: Proc. 4th Intl. Workshop on Constraints and Language Processing. Volume 113 of Roskilde University Computer Science Research Report. (2007) 53–64
9. Christiansen, H., Li, B.: Approaching the chinese word segmentation problem with CHR grammars. In: Proceedings of CSLP 2011, the 6th International Workshop on Constraints and Language Processing. A workshop at CONTEXT '11: The 7th International and Interdisciplinary Conference on Modeling and Using Context 2011. Volume 134 of Roskilde University Computer Science Research Report. (2011) 21–31

10. Christiansen, H.: CHR Grammars. Int'l Journal on Theory and Practice of Logic Programming **5**(4-5) (2005) 467–501
11. Frühwirth, T.: Constraint Handling Rules. Cambridge University Press (2009)
12. Penn, G., Richter, F.: The other syntax: Approaching natural language semantics through logical form composition. In Christiansen, H., Skadhauge, P.R., Villadsen, J., eds.: Constraint Solving and Language Processing. First International Workshop, CSLP 2004, Roskilde, Denmark, September 1-3, 2004, Revised Selected and Invited Papers. Volume 3438 of Lecture Notes in Computer Science., Springer (2005) 48–73
13. Christiansen, H., Dahl, V.: Meaning in Context. In Dey, A., Kokinov, B., Leake, D., Turner, R., eds.: Proceedings of Fifth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-05). Volume 3554 of Lecture Notes in Artificial Intelligence. (2005) 97–111
14. Frühwirth, T.W.: Theory and practice of Constraint Handling Rules. Journal of Logic Programming **37**(1-3) (1998) 95–138
15. van den Bosch, A., Busser, B., Daelemans, W., Canisius, S.: An efficient memory-based morphosyntactic tagger and parser for dutch. In van Eynde, F., Dirix, P., Schuurman, I., Vandeghinste, V., eds.: Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting (CLIN17), Leuven, Belgium (2007) 99–114
16. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005). (2005) 363–370
17. Strötgen, J., Gertz, M.: Multilingual and cross-domain temporal tagging. Language Resources and Evaluation (2012)
18. Strötgen, J., Gertz, M.: Heideltime: High quality rule-based extraction and normalization of temporal expressions. In: Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval-2010). (2010) 321–324

# Describing Music with MetaGrammars

Simon Petitjean

LIFO, University of Orleans

**Abstract.** This paper aims to show metagrammars are not only suited for large grammars development, but can also deal with other types of descriptions. We give the example of musical chords, that can be quite naturally described within the modular scope offered by eXtensible MetaGrammar.

## 1 Introduction

Metagrammars, introduced by [1], are known as a way to reduce the amount of work and time needed for the development and maintenance of large tree grammars for natural languages. The main concept of metagrammars is to factorize redundant information in order to minimize the description. XMG (eXtensible MetaGrammar) is a metagrammatical approach that has been used to describe the syntax and semantics of some languages (French, German, English), and a preliminary verbal morphology in Ikota [2], using tree formalisms (Tree Adjoining Grammars and Interaction Grammars). The modularity given by the scope allows it to be used for other purposes. In order to illustrate the extensibility of the tool, we chose to explore a quite different language, the one of music. The goal of this work is to generate, from an abstract description, a lexicon of musical chords. The choice of this grammar can be explained in different ways: the first is that we aim to prove XMG can process really different types of lexicons. Another interesting aspect about this work is that the rules involved in musical theory (rhythm, harmony,...) are really different from the ones we dealt with until now. Musical theory can be found in [3] for example. Nevertheless, if large scale grammars can cover a significant part of the language, it is hard to imagine that a grammar of musical sentences could do the same with a significant part of melodies. In a first part, we will present the metagrammatical tool used, then we will give the metagrammar of musical chords. Finally, we will conclude and give some perspectives.

## 2 The scope: eXtensible MetaGrammar

XMG [4] stands both for a metagrammatical language and the compiler for this language. The input of the tool is the metagrammar, that is to say the abstract description of the grammar, written by the linguist. The description is then compiled and executed by XMG to produce the lexicon.The metagrammatical language mainly relies on the notion of abstraction and the concepts of logical

programming. The core of the language is called the control language, and is defined as follows:

$$\begin{aligned} \mathit{Class} &:= \mathit{Name}[p_1, \ldots, p_n] \rightarrow \mathit{Content} \\ \mathit{Content} &:= \langle \mathit{Dim} \rangle \{ \mathit{Desc} \} \mid \mathit{var} = \mathit{Name}[\ldots] \mid \mathit{Content} \vee \mathit{Content} \\ &\quad \mid \mathit{Content}; \mathit{Content} \end{aligned}$$

Abstraction is made possible by classes, which associate a content to a name. Contents can be conjunctions (;) and disjunctions (|) of descriptions or calls to other classes. Descriptions are accumulated within a dimension, which allows to separate the different types of information (syntactic, semantic, morphological...). Every dimension comes with its dedicated description language, the next step is to create a new one, adapted to music.

### 2.1 A music dimension

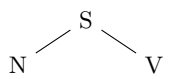
To describe trees, we use the following language:

$$\begin{aligned} \mathit{Desc} &:= x \rightarrow y \mid x \rightarrow^{+} y \mid x \rightarrow^{*} y \mid x \prec y \mid x \prec^{+} y \mid x \prec^{*} y \\ &\quad \mid x\ [f{:}E] \mid x\ (p{:}E) \mid \mathit{Desc}\ ;\ \mathit{Desc} \mid \mathit{SynExpr} = \mathit{SynExpr} \\ \mathit{SynExpr} &:= \mathit{var} \mid \mathit{const} \mid \mathit{var.const} \end{aligned}$$

The elementary unit we manipulate is the node. Nodes can be declared with feature structures for linguistic information (syntactic category, ...), and some properties. We also need to accumulate relations between the nodes, in order to describe the structure of the tree. $\rightarrow$ and $\prec$ are respectively the operators for dominance and precedence, $^{+}$ and $^{*}$ representing the transitive closure and the transitive reflexive closure. Unification of expression (=) can also be written. The '.' operator allows to access a variable when applied to a class and the value corresponding to a key when applied to a feature structure.

After being accumulated, the description has to be solved. The result is the set of all the minimal models of the description. For example, here is a XMG description and a tree fragment it models :

```
CanSubj ->
<syn>{ S [cat=s]; N [cat=n]; V [cat=v]; S->N; S->V; N<<V }
```




In this work, we will accumulate notes. The tree structures we already can describe seem to be adapted to order the notes. A note variable is declared as a node, with a feature structure which contains the information that characterizes it. Here, we will use the same language and just add an arithmetic operator

so that we can compare the values associated to notes. The music description language is consequently the same, except for the '+' in the expressions:

$$\begin{aligned} MDesc &:= x \rightarrow y \mid x \rightarrow^{+} y \mid x \rightarrow^{*} y \mid x \prec y \mid x \prec^{+} y \mid x \prec^{*} y \\ &\quad \mid x\ [f{:}E] \mid x\ (p{:}E) \mid MDesc\ ;\ MDesc \mid MExpr\ =\ MExpr \\ MExpr &:= var \mid const \mid var.const \mid MExpr\ +\ MExpr \end{aligned}$$

### 2.2 Principles

To ensure the well-formedness of the generated models, XMG comes with a library of principles that the user can activate. They consist in a set of linguistic constraints over the accumulated description. In the description of music, a principle could ensure that every measure is coherent with the time signature. Another principle could apply rewrite rules to chord sequences (as done in [5]) to generate a set of variations for those sequences.

### 2.3 Sharing data with other dimensions?

As it was the case for syntax and semantics, the accumulation produced by the execution of the metagrammar could take place in various dimensions, allowing to share data between them. One of the options would be to create a dimension for another clef: if we assume the grammar we produce is written in G-clef, another dimension could contain the transposition of the accumulation in F-clef. This can also be used for transposition to gather instruments tuned in different tones (for example B♭ clarinet and alto saxophone in E♭). Another dimension could also encode the fingerings to realize the notes (or chords) on some instrument. The accumulation of predicates in the semantic dimension is also an interesting perspective to experiment some theories on semantics of music.

## 3 A metagrammar for musical chords

### 3.1 Describing notes

The first abstraction we create is the elementary unit of music: the note. From a unique class, we want to generate any note, with the information needed to characterize it. The feature structure of the note contains typed data, types being checked during the compilation. The essential features we will use are the name of the note, its accidental (here we will only consider notes with at most one accidental), and its global height (in term of semi tones). As we will not describe rhythm in this preliminary work, we chose not to represent the duration of the note.

In the metagrammar, 'name' is the human readable name of the note. Its type is enumerated, with possible values {A,B,C,D,E,F,G}, 'acc' is the accidental of

the note (sharp, flat, or none), 'namerank' is an integer associated to the name of the note, beginning with A=1, 'height' is the number of semi tones from A to the note. 'namerank' and 'height' depend on the two previous features and are only used in the metagrammar to compare notes. For example, the feature structure for a D♯ will be [name=D, acc=sharp, namerank=4, height=5]. Scales are not taken into account, which means that the unique G we consider is both 7 semi tones above C and 5 semi tones below it.

Two things have to be constrained:
- the value and the value integer have to match
- the global height depends on the note value and its accidental

```
note ->
<music>{
   N [name=V, namerank=NR, acc=ACC, height=H];
  {
    { V=A ; NR=1 ; HT=1 } |
    { V=B ; NR=2 ; HT=3 } |
    { V=C ; NR=3 ; HT=4 } |
    { V=D ; NR=4 ; HT=6 } |
    { V=E ; NR=5 ; HT=8 } |
    { V=F ; NR=6 ; HT=9 } |
    { V=G ; NR=7 ; HT=11 }
  };
  {
    { ACC=sharp;    H = HT + 1 } |
    { ACC=flat;     H = HT - 1 } |
    { ACC=none;     H = HT     }
  }
}
```

XMG builds a description for each case it tries (if all unifications succeed). As expected, the execution of the class *Note* (with 7 notes and 3 accidentals) leads to 21 descriptions.

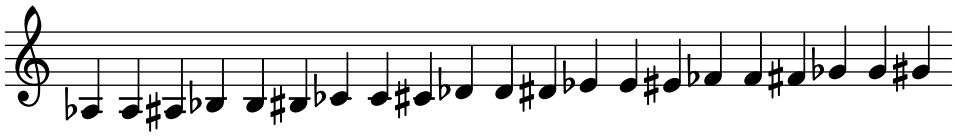


**Fig. 1.** The 21 generated notes

### 3.2 Describing intervals

An interval features two notes, and is defined by its number and its quality. The number is the difference between the staff positions of the notes, and the quality

is relative to the number of semi tones between the nodes. Thus the number is given by the difference between the name ranks of the notes, and the quality by the difference between the heights. To represent a major third, we have to accumulate two notes separated by exactly two tones. The class for this interval has to feature two notes, and two constraints : the name ranks of the notes have to be separated by two unities, and the global height by four semi tones. We add another node to the description, dominating the two notes, and holding the information about the interval into its feature structure.

```
Mthird ->
  First=note[];                Third=note[];
  FirstF=First.Feats;          ThirdF=Third.Feats;
  TRank=FirstF.namerank;       ThRank=ThirdF.namerank;
  THeight=FirstF.height;       ThHeight=ThirdF.height;
  FirstN=First.N;              ThirdN=Third.N;
  {   ThRank=TRank + 2     | ThRank=TRank - 5     };
  {   ThHeight=THeight + 4 | ThHeight=THeight - 8 };
  <music>{
  Root [third=major];
  Root ->+ FirstN;             Root ->+ ThirdN;
  FirstN >> ThirdN
  }
```

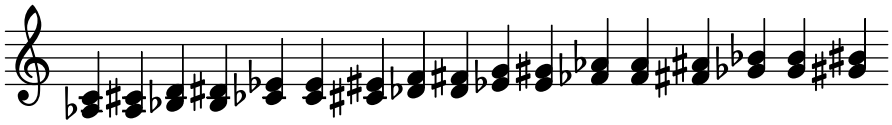


**Fig. 2.** The 17 generated major thirds

Intuitively, each one of the 21 notes should be the tonic for a major third, but only 17 major thirds are generated. The reason is the four remaining intervals involve double accidentals. We do the same for minor third, with an interval of three semi tones.

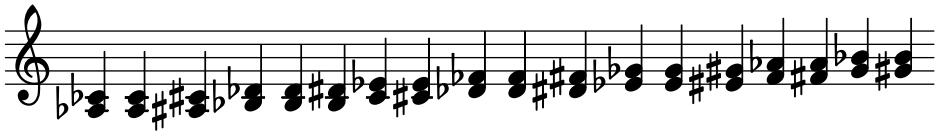


**Fig. 3.** The 18 generated minor thirds

### 3.3 Describing chords

Now, let us consider three notes chords, called triads, in their major and minor modes. A triad is composed of a note, called *tonic*, a third and a perfect fifth. The major chord features a major third, and the minor chord a minor one. We thus need to accumulate two intervals, and constraint their lowest notes to unify, the result being the tonic of the chord.

```
Major ->
  Third=Mthird[];
  Fifth=fifth[];
  Tonic=Third.First;
  Tonic=Fifth.First;
  Third.Root=Fifth.Root;
  ThirdN=Third.N;
  FifthN=Fifth.N;
  Root=Third.Root
```

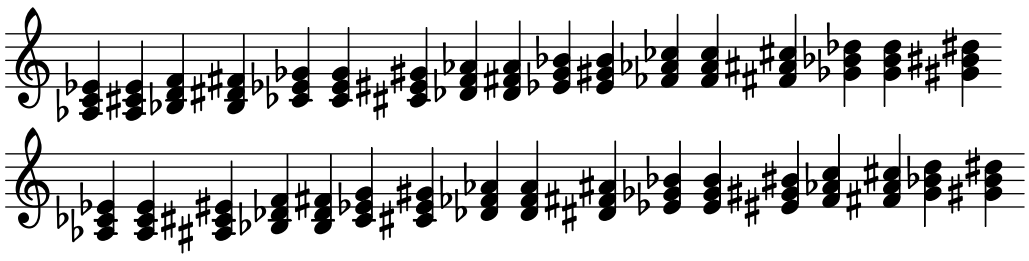


**Fig. 4.** The 17 generated major and minor triads

The same method can be applied to produce seventh chords: a seventh chord would then be defined as the conjunction of a triad and a seventh interval (the tonic of the triad unifying with the lowest note of the interval).

## 4 Conclusion and future work

We proposed a metagrammar for a part of musical theory: from a short abstract description, we generated lexicons of major and minor triads. The initial aim of this work was to prove XMG can quickly be extended in order to deal with really different theories (not necessarily linguistic theories). The metagrammatical approach appeared to be useful in the context of music description. The redundancy we factorized was not essentially structural, as in most of the cases studied previously, but also at the level of the algebraic constraints. This very preliminary work however needs to be extended to more complex musical phenomena. Describing melodies could obviously not be done in a similar way, generating every solution, but sequences of chords seem to be an interesting next step. Adapted principles could be designed and used to limit the number of models.

## References


1. Candito, M.: A Principle-Based Hierarchical Representation of LTAGs. In: Proceedings of the 16$^{\text{th}}$ International Conference on Computational Linguistics (COLING'96). Volume 1., Copenhagen, Denmark (1996) 194–199
2. Duchier, D., Magnana Ekoukou, B., Parmentier, Y., Petitjean, S., Schang, E.: Describing Morphologically-rich Languages using Metagrammars: a Look at Verbs in Ikota. In: Workshop on "Language technology for normalisation of less-resourced languages", 8th SALTMIL Workshop on Minority Languages and the 4th workshop on African Language Technology, Istanbul, Turkey (May 2012)
3. Grove, G., Sadie, S.: The New Grove dictionary of music and musicians. Number vol. 13 in The New Grove Dictionary of Music and Musicians. Macmillan Publishers (1980)
4. Crabbé, B., Duchier, D.: Metagrammar redux. In Christiansen, H., Skadhauge, P.R., Villadsen, J., eds.: Constraint Solving and Language Processing, First International Workshop (CSLP 2004), Revised Selected and Invited Papers. Volume 3438 of Lecture Notes in Computer Science., Roskilde, Denmark, Springer (2004) 32–47
5. Chemillier, M.: Toward a formal study of jazz chord sequences generated by steedman's grammar. Soft Comput. **8**(9) (2004) 617–622

# Estimating Constraint Weights from Treebanks

Philippe Blache

Laboratoire Parole et Langage
CNRS & Aix-Marseille Université
blache@lpl-aix.fr

**Abstract.** We present in this paper a technique aiming at estimating constraint weights from treebanks. After a presentation of constraint-based treebank acquisition, a method assigning weights to individual constraint is proposed. An experiment by means of human acceptability judgments gives a first validation of the approach.

**Keywords:** Constraint weights, parsing, Property Grammars, treebank

## 1 Introduction

Constraint-based parsing can be computationally expensive because of overgeneration. This drawback becomes even severe when using constraint relaxation as in several linguistic theories such as *Optimality Theory* (see [1]) or *Property Grammars* [2]. The problem is that constraint relaxation is not a syntactic sugar, but the core of these formalisms for different reasons. Typically, OT is an optimization problem: given a set of syntactic structures, the parser identifies the optimal (the one that violates less constraints). In model-theoretic approaches such as PG, the question is different, but the consequence the same: building a model consists in evaluating a constraint system for a given input. In this case, a model can be sub-optimal (in other words, some constraints can be violated).

Constraint relaxation is then an important mechanism, especially when trying to deal with non canonical linguistic material such as spoken language (which is one of the main challenges for natural language processing). As a consequence, it becomes necessary to develop techniques for controlling the parsing process. One of those consists in *weighting* constraints. Such mechanism has been integrated in the theoretical framework of *Harmonic Grammar* (see [3]), from which OT has been proposed. On the applicative side, constraints weights have been implemented in the development of *Weighted Constraint Dependency Grammars* (see [4]). This formalism allows for a distinction between *hard* and *soft* constraints through the use of weights set by the grammar developer. This task is manual and depends on the developer knowledge of the linguistic features and the consequence of ranking on the parsing process[1]. The validation of the resulting set of weights can be done automatically by evaluating the weight model against a reference corpus.

From the cognitive side, different works tried to distinguish between hard and soft constraint in the human parsing process. In this vein, [6] proposes a description of gradience phenomena based on constraint violation: violating hard constraints leads to strong unacceptability by human subjects where soft constraints violation entails mild unacceptability. In the PG framework, several works have been done proposing grammaticality models that associate weights to constraint types (see [7], [8]).

These approaches illustrate the importance of weighting constraint, both for computational, theoretical and cognitive reasons. However, constraint weighting mainly remains a manual and empirical task. We propose in this paper a technique making it possible to acquire automatically constraint weights from treebanks. This technique is presented in the framework of *Property Grammars*, even though it is theory-neutral: the principle relying on constraint quantification can be applied whatever the formalism, provided that the number of violated and satisfied constraints can be evaluated.

[1] An experiment trying to automatically acquire constraint weights by means of genetic algorithm is reported in [5], but not applied to further WCDG grammar development.

In the remainder of the paper, we will briefly present the formalism before describing its application in the development of a constraint-based treebank. We present then the elaboration of the weighting model before proposing an experimental validation.

## 2 Property Grammars: main features

Property Grammars [2] (noted hereafter *PG*) is a fully constraint-based theory in the sense that all information is represented by means of constraints. The idea consists in (1) representing explicitly all kinds of syntactic information (such as linearity, cooccurrence, exclusion, etc.) and (2) to implement this information only by means of constraints; not using any generate-and-test architecture such as in OT or WCDG. The consequence is twofold: (1) the parsing mechanism is purely constraint satisfaction and (2) a syntactic description (which is in classical approaches a tree) is the state of the constraint system after evaluation (which can be represented as a graph).

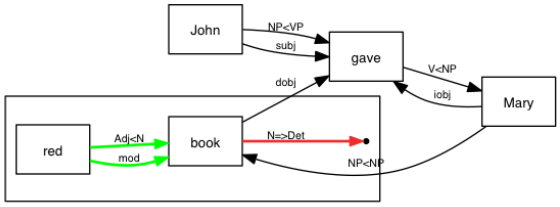
A *PG* formal presentation can be found in [9], which proposes the following notations for the different constraint types:

| | | |
|---|---|---|
| Obligation | $A : \Delta B$ | at least one $B$ |
| Uniqueness | $A : B!$ | at most one $B$ |
| Linearity | $A : B \prec C$ | $B$ precedes $C$ |
| Requirement | $A : B \Rightarrow C$ | if $\exists B$, then $\exists C$ |
| Exclusion | $A : B \not\Rightarrow C$ | not both $B$ and $C$ |
| Constituency | $A : S?$ | all children $\in S$ |
| Dependency | $A : B \rightsquigarrow C$ | dependency constraints |

A *PG* grammar describes non-terminals with sets of constraints. The following example illustrates the grammar of the *AP* in French:

| | |
|---|---|
| *Const* | AP: {AdP, A, VPinf, PP, Ssub, AP, NP}? |
| *lin* | A $\prec$ {VPinf, Ssub, PP, NP, AP}<br>AdP $\prec$ {A, Ssub, PP}<br>AP $\prec$ {A, AdP}<br>PP $\prec$ {Ssub} |
| *Dep* | AP: {AdP, VPinf, PP, Ssub, NP} $\rightsquigarrow$ A |
| *Uniq* | AP: {A, VPinf, Ssub}! |
| *Oblig* | AP: $\Delta$ A |
| *Excl* | AP: VPinf $\not\Rightarrow$ {PP, Ssub} |

In this approach, all constraints can be relaxed, which makes it possible to describe any kind of input, including ill-formed ones. The following figure illustrates the situation when a constraint is violated: in this example, the determiner is not realized, which violates the requirement constraint between *Det* and *N*. On the basis of such description, it becomes possible to know precisely for each category the constraints that are satisfied or violated, opening the way to a quantified modeling of grammaticality.




In the remaining of this paper, we will show how to identify the relative importance of constraints (in other words calculate their weights) thanks to the possibilities of the approach applied to a corpus. We present in the next section the *PG* treebank from which data are extracted and the weighting model has been built.

| | AdP | AP | NP | PP | SENT | Sint | Srel | Ssub | VN | VNinf | VNpart | VP | VPinf | VPpart | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Const | 10 | 7 | 13 | 7 | 8 | 8 | 7 | 10 | 6 | 7 | 7 | 10 | 9 | 8 | 115 |
| Dep | 5 | 6 | 18 | 5 | 3 | | | | 5 | 5 | 6 | 8 | | | 59 |
| Exc | 1 | 2 | 44 | | 2 | 6 | 3 | 3 | | | | | | | 61 |
| Req | | | 6 | | | | | | | 4 | 4 | | | | 14 |
| Lin | 18 | 10 | 36 | 6 | 5 | 4 | 7 | 14 | 11 | 6 | 7 | 24 | 13 | 7 | 165 |
| Oblig | 1 | 1 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 2 | 1 | 1 | 1 | 20 |
| Uniq | 4 | 3 | 10 | 3 | 3 | 4 | 4 | 1 | 2 | 4 | 5 | 3 | 7 | 6 | 59 |
| | 39 | 22 | 131 | 22 | 22 | 23 | 22 | 29 | 25 | 29 | 31 | 46 | 30 | 22 | 493 |

**Fig. 1.** Distribution of the constraints in the FTB-acquired grammar

## 3 Source corpus: the PG-enriched French Treebank

Our experiment relies on a *PG* treebank , built by enriching the *French Treebank* (hereafter *FTB*, see [10]) with *PG* descriptions. The original treebank consists of articles from the newspaper "*Le Monde*". It contains 20,648 sentences and 580,945 words.

The first step for building the *PG* treebank consisted in acquiring a *PG* grammar from the original *FTB*. This has been done semi-automatically thanks to a technique described in [11]. Figure 1 gives the number of acquired constraints, for each non-terminal. We can note that the *PG* grammar is rather compact (only 493 constraints for the entire grammar) and that the constraint distribution is not balanced (*NP* and *VP* represent 35% of the total amount of constraints). Moreover, some categories (such as *VP* or *VPinf*) do not make use of all the constraint types in their description. This kind of information is important in the identification of the constraint weights: a given constraint can be more or less important depending on the total number of constraints of the same type used for the category. For example, if a category only has one linearity constraint, the probability for this constraint to be strong is high.

Once the *PG* grammar built, enriching the treebank consists in applying a grammatical constraint solver to each constraint system describing a category. The data used in the remaining of the paper rely on an experiment done on a subset of 470 sentences (14,695 words) taken from the PG-enriched *FTB*. The figure 2 presents some results for the *NP* (4,199 different realizations in this subcorpus).

As already shown in the previous section, each category is described by different constraints for each constraint type. For example, the *NP* has 36 different linearity constraints. In the following table, rows represent the index of the constraint in the grammar. This example presents the first 12 constraints for each constraint type in the grammar of *NP*:

| | lin | dep | exc | req | oblig | uniq |
|---|---|---|---|---|---|---|
| 0 | Det $\prec$ N | Det ,N | Pro,Det | Det ,N | N | Det |
| 1 | Det $\prec$ Np | AP,N | Pro,N | PP,N | Pro | N |
| 2 | Det $\prec$ AdP | AdP,N | Pro,AdP | AP,N | Np | Srel |
| 3 | Det $\prec$ AP | NP,N | Pro,AP | VPpart,N | Clit | VPpart |
| 4 | Det $\prec$ VPpart | VPpart,N | Pro,NP | VPinf,N | | VPinf |
| 5 | Det $\prec$ VPinf | VPinf,N | Pro,VPpart | Ssub,N | | Ssub |
| 6 | Det $\prec$ Ssub | PP,N | Pro,VPinf | | | AdP |
| 7 | Det $\prec$ Srel | Srel,N | Pro,Ssub | | | Pro |
| 8 | Det $\prec$ PP | Ssub,N | AdP,VPpart | | | NP |
| 9 | Det $\prec$ NP | AP,Np | AdP,VPinf | | | Clit |
| 10 | N $\prec$ AdP | NP,Np | AdP,Srel | | | |
| 11 | N $\prec$ NP | PP,Np | Ssub,AdP | | | |

Each line in the figure 2 indicates the type of the constraint, its polarity (for example $lin_p$ stands for *satisfied linearity constraints* where $lin_m$ represents unsatisfied ones) and the number of times this constraint is evaluated. For example, the column 0 indicates that the linearity constraint number 0 (i.e. $Det \prec N$) is satisfied 1,933 times and never violated, or that the linearity constraint 3 ($Det \prec AP$) is satisfied 455 times and violated 13 times. The second column of the table indicates the total amount of evaluated constraints used in the description of the *NP* for this corpus.

| | Const. gram. | Total | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\text{lin}_p$ | 35 | 5451 | 1933 | 114 | 24 | 455 | 73 | 23 | 7 | 92 | 754 | 140 | 27 | 220 | 89 | 25 | 996 | 119 |
| $\text{lin}_m$ | 16 | 101 | 0 | 0 | 4 | 13 | 0 | 1 | 0 | 0 | 1 | 0 | 6 | 2 | 0 | 1 | 2 | 0 |
| $\text{obl}_p$ | 4 | 3989 | 3187 | 90 | 563 | 149 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\text{obl}_m$ | 1 | 210 | 210 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\text{dep}_p$ | 16 | 4368 | 1933 | 662 | 32 | 222 | 89 | 26 | 997 | 119 | 7 | 13 | 164 | 27 | 7 | 18 | 22 | 30 |
| $\text{dep}_m$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\text{exc}_p$ | 44 | 32917 | 2172 | 3277 | 150 | 818 | 860 | 193 | 117 | 98 | 169 | 89 | 219 | 68 | 740 | 778 | 115 | 39 |
| $\text{exc}_m$ | 26 | 163 | 12 | 0 | 2 | 6 | 2 | 4 | 2 | 2 | 0 | 2 | 6 | 4 | 4 | 4 | 2 | 0 |
| $\text{req}_p$ | 6 | 3714 | 1933 | 997 | 662 | 89 | 26 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\text{req}_m$ | 6 | 274 | 161 | 71 | 21 | 15 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\text{uniq}_p$ | 10 | 6258 | 2094 | 3187 | 161 | 101 | 29 | 10 | 58 | 90 | 379 | 149 | 0 | 0 | 0 | 0 | 0 | 0 |
| $\text{uniq}_m$ | 4 | 179 | 0 | 0 | 1 | 3 | 0 | 0 | 2 | 0 | 173 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Fig. 2.** Constraint distribution for the NP in the PG-enriched FTB

These results call for several remarks. First, distribution is not balanced from one category to another: some constraints are frequently evaluated, some other not. For example, the first linearity constraint (noted *lin[0]*) is satisfied 1,933 times, which represents 34.8% of the total number of linearity constraints evaluated in the description of the different *NP* realizations in this subcorpus.

Moreover, the number of constraints per type can vary a lot. The first column indicates the number of *NP* constraints in the grammar that are actually used. In the case of our corpus, 35 different linearity constraints are used among the 36 possible in the grammar. This gives an indication of the relative importance of each constraint in terms of distribution: a constraint frequently used is supposed to be more important when the total number of constraints evaluated in the type is high. This is the case for the linearity constraint *lin[0]* which is both frequent and belongs to a set of 35 evaluated constraints.

## 4 The weighting model

Several parameters can be calculated from these figures, that can be used in the elaboration of the weighting model. We note in the following, for a given category:

- $E_t^+$ the total number of times constraints are satisfied for the type $t$. In the example figure 2, the total number of satisfied linearity constraints is $E_{lin}^+ = 5,451$. The corresponding figure for violated constraints is noted $E_t^-$
- $E_t$ is the total number of evaluated constraints: $E_t = E_t^+ + E_t^-$
- $C_{t,i}^+$ the number of times the constraint indexed $i$ in the type $t$ is satisfied. In our example, the linearity constraint indexed 0 is satisfied 1,933 times, noted $C_{lin,0}^+ = 1,933$. The corresponding figure for violated constraints is noted $C_{t,i}^-$
- $C_{t,i}$ is the number of times the constraint $t[i]$ is satisfied or violated: $C_{t,i} = C_{t,i}^+ + C_{t,i}^-$
- $G_t$ the number of constraints of type $t$ in the grammar that are used in the evaluation. In our example, 35 different linearity constraints types are used in this corpus (noted $G_{lin} = 35$)

Besides this figures, we define the following functions:

*Raw coverage* : calculated as the total number of times the constraint is evaluated (satisfied or not) divided by the total number of evaluated constraints in the type: $RC_{t,i} = C_{t,i}/E_t$

*Balanced coverage* : the raw coverage balanced by the number of different constraints used in the evaluation of the category in the corpus: $BC_{t,i} = RC_{t,i}/G_t$

*Satisfiability ratio* : rate of satisfied vs. unsatisfied constraints. $SR_{t,i} = (C_{t,i}^+ - C_{t,i}^-)/C_{t,i}$

This different figures make it possible to propose an automatic constraint weight estimation. We propose for this to take into account the constraint coverage (balanced by the number of different types of constraints) and the satisfaction ratio, giving an estimation on the frequency of the violation. The weights are calculated as follows:

$$W_{[t,i]} = SR_{[t,i]} * BC_{[t,i]}$$

The following table presents these different parameters values for the two first constraints (indexed 0 and 1) of the *NP*, on the basis of the figures extracted from the corpus:

| | Index 0 | | | | Index 1 | | | |
|---|---|---|---|---|---|---|---|---|
| | RC | BC | SR | Weight | RC | BC | SR | Weight |
| lin | 0.35 | 12.19 | 1.00 | 12.19 | 0.02 | 0.72 | 1.00 | 0.72 |
| obl | 0.76 | 3.04 | 0.88 | 2.66 | 0.02 | 0.09 | 1.00 | 0.09 |
| dep | 0.44 | 7.08 | 1.00 | 7.08 | 0.15 | 2.42 | 1.00 | 2.42 |
| exc | 0.07 | 2.89 | 0.99 | 2.86 | 0.10 | 4.36 | 1.00 | 4.36 |
| req | 0.48 | 2.91 | 0.85 | 2.46 | 0.25 | 1.50 | 0.87 | 1.30 |
| uni | 0.33 | 3.25 | 1.00 | 3.25 | 0.50 | 4.95 | 1.00 | 4.95 |

These partial results show important differences between the weights, illustrating different situations. Some constraints (only few of them when taking into account the total results for all the categories) have both high coverage and good satisfaction ratio. This is the case for the first linearity constraint of the *NP* ($Det \prec N$) that gets the higher weight. It is interesting to note the importance of balancing the coverage: some constraints, such as obligation, are very frequent but can also be often violated, which is correctly taken into account by the model.

The following table presents the constraint ranking for the first *NP* constraints. We can note that usually, the highest ranked constraint has a weight significantly higher than others. This effect confirms the fact that when parsing real-life corpora, there is no great syntactic (as well as lexical) variability. In the case of a weighting repartition between hard and soft constraints, this characteristics would militate in favor of having only one or two hard constraint per type.

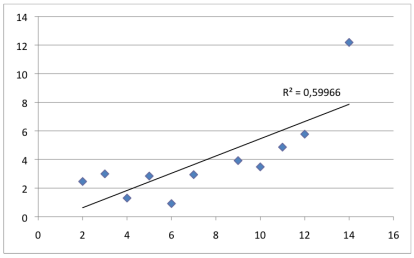
| rank | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| lin | Det $\prec$ N<br>12.19 | Det $\prec$ Np<br>0.72 | N $\prec$ PP<br>0.18 | Det $\prec$ PP<br>0.14 | Det $\prec$ AP<br>0.08 | N $\prec$ NP<br>0.04 | Det $\prec$ NP<br>0.03 | N $\prec$ Srel<br>0.02 | Det $\prec$ Srel<br>0.02 | N $\prec$ VPpart<br>0.02 |
| obl | N<br>2.66 | Np<br>0.13 | Pro<br>0.09 | Clit<br>0.04 | | | | | | |
| dep | Det $\rightsquigarrow$ N<br>7.08 | AP $\rightsquigarrow$N<br>2.42 | PP $\rightsquigarrow$N<br>0.23 | NP $\rightsquigarrow$N<br>0.05 | NP $\rightsquigarrow$Np<br>0.04 | Srel $\rightsquigarrow$N<br>0.03 | VPpart $\rightsquigarrow$N<br>0.02 | AdP $\rightsquigarrow$ N<br>0.01 | PP $\rightsquigarrow$ Pro<br>0.01 | PP $\rightsquigarrow$ Np<br>0.01 |
| exc | Pro,N<br>4.36 | Pro,Det<br>2.86 | Np,N<br>0.12 | Clit,N<br>0.10 | Clit,Det<br>0.07 | Ssub,PP<br>0.03 | Clit,NP<br>0.03 | Clit,AP<br>0.03 | Pro,NP<br>0.03 | NP,Pro<br>0.03 |
| req | Det $\Rightarrow$ N<br>2.46 | PP $\Rightarrow$ N<br>1.30 | AP $\Rightarrow$ N<br>0.16 | VPpart $\Rightarrow$ N<br>0.02 | VPinf $\Rightarrow$ N<br>0.01 | Ssub $\Rightarrow$ N<br>0.00 | | | | |
| uni | Det<br>3.25 | N<br>4.95 | Srel<br>0.02 | VPpart<br>0.01 | VPinf<br>0.00 | Ssub<br>0.00 | AdP<br>0.01 | Pro<br>0.01 | NP<br>0.02 | Clit<br>0.02 |

## 5 Result validation

As a first validation of the result, we have compared the constraint ranking with acceptability judgments acquired in the framework of another study (see [7]). This experiment consisted in asking subjects to rank following a scale (using the method of *magnitude estimation*) the grammaticality of different sentences. Each sentence violates one constraint. The data set was built using the following matrix sentences (we indicate in the second column the violated constraint). In this table, sentences are presented by increasing unacceptability judgment. The rightmost column shows the corresponding constraint weight.

| Sentence | Constraint | Weights |
|---|---|---|
| Marie a emprunté très long chemin pour le retour | NP: N $\Rightarrow$ Det | 2,46 |
| Marie a emprunté un très long chemin pour pour le retour | PP: uniq(Prep) | 2,99 |
| Marie a emprunté un très long chemin chemin pour le retour | NP: uniq(N) | 1,30 |
| Marie a emprunté emprunté un très long chemin pour le retour | VP: uniq(VN) | 2,84 |
| Marie a emprunté un très long chemin le retour | PP: oblig(Prep) | 0,92 |
| Marie a emprunté un très long long chemin pour le retour | AP: uniq(Adj) | 2,94 |
| Marie a emprunté un très long chemin pour | PP: Prep $\rightsquigarrow$ NP | 3,92 |
| Marie a emprunté un long très chemin pour le retour | AP : Adv $\prec$ Adj | 3,48 |
| Marie un très long chemin a emprunté pour le retour | VP : VN $\prec$ NP | 4,86 |
| Marie a emprunté un très long chemin le retour pour | PP : Prep $\prec$ NP | 5,77 |
| Marie a emprunté très long chemin un pour le retour | NP : Det $\prec$ N | 12,19 |

One can note a tendency for linearity constraint violation to be less acceptable than uniqueness one, which is globally confirmed by the total figures we obtained on the corpus. When comparing more precisely the judgment ranking with constraint weights, we observe a very good correlation (correlation coefficient: 0,77) as illustrated with the tendency figure (ranks in X axis and weights in Y axis):




This first result, if not a true evaluation, gives an indication of the validity of the method. The method has now to be applied to the entire corpus, acquiring a more complete weighting model. An actual evaluation would then consist in evaluating how weights improve the control of the process and reduce the search space of the parser.

## 6 Conclusion

We have presented in this paper a method making it possible to acquire automatically constraint starting from treebanks. The method is generic and language-independent. A first validation have shown that automatic weightings have a good correlation with human judgments. This method constitutes a preliminary answer to the difficult question of automatic constraint weighting in the context of constraint-based parsing.

## References

1. Prince, A., Smolensky, P.: Optimality Theory: Constraint Interaction in Generatire Grammar. Technical report, TR-2, Rutgers University Cognitive Science Center, New Brunswick, NJ. (1993)
2. Blache, P.: Les Grammaires de Propriétés : Des contraintes pour le traitement automatique des langues naturelles. Hermès (2001)
3. Smolensky, P., Legendre, G.: The Harmonic Mind From Neural Computation to Optimality-Theoretic Grammar. MIT Press (2006)
4. Foth, K., Menzel, W., Schröder, I.: Robust Parsing with Weighted Constraints. Natural Language Engineering **11**(1) (2005) 1–25
5. Schröder, I., Pop, H.F., Menzel, W., Foth, K.A.: Learning weights for a natural language grammar using genetic algorithms. In Giannakoglou, K., Tsahalis, D., Periaux, J., Papailiou, K., Fogarty, T., eds.: Evolutionary Methods for Design, Optimisation and Control. (2002)
6. Sorace, A., Keller, F.: Gradience in linguistic data. Lingua **115**(11) (2005) 1497–1524
7. Blache, P., Hemforth, B., Rauzy, S.: Acceptability prediction by means of grammaticality quantification. In: ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, Association for Computational Linguistics (July 2006)
8. Blache, P.: Evaluating language complexity in context: New parameters for a constraint-based model. In: CSLP-11, Workshop on Constraint Solving and Language Processing. (2011)
9. Duchier, D., Prost, J.P., Dao, T.B.H.: A model-theoretic framework for grammaticality judgements. In: Conference on Formal Grammar (FG'09). (2009)
10. Abeillé, A., Clément, L., Toussenel, F.: Building a treebank for french. In Abeillé, A., ed.: Treebanks, Kluwer, Dordrecht (2003)
11. Blache, P., Rauzy, S.: Hybridization and treebank enrichment with constraint-based representations. In: Workshop on Advanced Treebanking. (2012)

# On Language Acquisition Through Womb Grammars

Veronica Dahl[1], J. Emilio Miralles[1], and Leonor Becerra[2]

[1] Simon Fraser University, Burnaby, BC, V5A-1S6, Canada,
veronica@cs.sfu.ca, emiralle@sfu.ca
[2] Laboratoire Hubert Curien, Jean Monnet University, 18 rue Benoit Lauras, 42100 Saint-Etienne, France
leonor.becerra@univ-st-etienne.fr

**Abstract.** We propose to automate the field of language acquisition evaluation through Constraint Solving; in particular through the use of Womb Grammars. Womb Grammar Parsing is a novel constraint based paradigm that was devised mainly to induce grammatical structure from the description of its syntactic constraints in a related language. In this paper we argue that it is also ideal for automating the evaluation of language acquisition, and present as proof of concept a CHRG system for detecting which of fourteen levels of morphological proficiency a child is at, from a representative sample of the child's expressions. Our results also uncover ways in which the linguistic constraints that characterize a grammar need to be tailored to language acquisition applications. We also put forward a proposal for discovering in what order such levels are typically acquired in other languages than English. Our findings have great potential practical value, in that they can help educators tailor the games, stories, songs, etc. that can aid a child (or a second language learner) to progress in timely fashion into the next level of proficiency, and can as well help shed light on the processes by which languages less studied than English are acquired.

**Keywords:** Womb Grammar Parsing, Language Acquisition, Constraint Order Acquisition, Constraint Based Grammars, Property Grammars, CHRG

## 1 Introduction

Constraint-based linguistic models, such as HPSG [17] or Property Grammars [1], view linguistic constraints in terms of property satisfaction between categories, rather than in the more traditional terms of properties on hierarchical representations of completely parsed sentences. This view has several advantages, including allowing for mistakes to be detected and pointed out rather than blocking the analysis altogether, and has yielded good results for language analysis and grammar development. Language acquisition is a research area where constraint-based approaches can potentially make important contributions. Applications of constraint-based approaches to processing learner language have

been surveyed in [13], mostly around error detection, as in [3], which represents parsing as a constraint satisfaction problem and uses constraint relaxation with a general-purpose conflict detection algorithm. A recent and original development of constraint-based parsing is the Womb Grammar Parsing paradigm [9], which was designed to induce a target language's constraints on given simple phrases (e.g., noun phrases) from the corresponding constraints of another language called the source, given a corpus of correct phrases in the target language. Womb grammars have proved valuable not only for traditional applications such as analysis, but also for grammar sanctioning and to induce a correct grammar from that of another, related language.

In this article we propose a tailoring of Womb Grammars for another unusual application: that of computer-supported language acquisition, and we exemplify our ideas in terms of a novel application of constraint-based parsing: that of inducing the (incorrect) grammar in use by a person learning a language and detecting the level of proficiency of such a learner.

After presenting our methodological background in the next section, section 3 describes how to use Womb Grammars to detect a child's level of grammatical proficiency and how to induce the linguistic constraints that describe his or her grammar fragment. Section 4 briefly discusses how to adapt our research to discover the learning stages that are followed by languages other than English. Section 5 presents our concluding remarks.

## 2 Background

Womb Grammar Parsing was designed to induce, from known linguistic constraints that describe phrases in a language called the source, the linguistic constraints that describe phrases in another language, called the target. Grammar induction has met with reasonable success using different views of grammar: a) as a parametrized, generative process explaining the data [16, 14], b) as a probability model, so that learning a grammar amounts to selecting a model from a pre-specified model family [6, 19, 8], and c) as a Bayesian model of machine learning [12].

Most of these approaches have in common the target of inferring *syntactic trees*. As noted, for example, in [2], constraint-based formalisms that make it possible to evaluate each constraint separately are advantageous in comparison with classical, tree-based derivation methods. For instance the Property Grammar framework [1] defines phrase acceptability in terms of the properties or constraints that must be satisfied by groups of categories (e.g. English noun phrases can be described through a few constraints such as precedence (a determiner must precede a noun), uniqueness (there must be only one determiner), exclusion (an adjective phrase must not coexist with a superlative), and so on). Rather than resulting in either a parse tree or failure, such frameworks characterize a sentence through the list of the constraints a phrase satisfies and the list of constraints it violates, so that even incorrect or incomplete phrases will be parsed.

Womb Grammar Parsing follows these general frameworks, but focuses on *generating the constraints* that would sanction the input as correct, rather than on characterizing *sentence acceptability* in terms of (known) linguistic constraints. This is because it was conceived for grammar induction rather than only for parsing sentences, so it can operate on a corpus of sentences deemed correct to generate the set of grammatical constraints (i.e., the grammar description) that *would* result in all constraints being satisfied—i.e., the grammar for the language subset covered by the corpus. Thus, it is ideal for grammar correction and grammar induction, not just for flexible parsing.

More concretely: let $L^S$ (the source language) be a human language that has been studied by linguists and for which we have a reliable parser that accepts correct sentences while pointing out, in the case of incorrect ones, what grammatical constraints are being violated. Its syntactic component will be noted $L^S_{syntax}$, and its lexical component, $L^S_{lex}$.

Now imagine we come across a dialect or language called the target language, or $L^T$, which is close to $L^S$ but has not yet been studied, so that we can only have access to its lexicon ($L^T_{lex}$) but we know its syntax rules overlap significantly with those of $L^S$. If we can get hold of a sufficiently representative corpus of sentences in $L^T$ that are known to be correct, we can feed these to a hybrid parser consisting of $L^S_{syntax}$ and $L^T_{lex}$. This will result in some of the sentences being marked as incorrect by the parser. An analysis of the constraints these "incorrect" sentences violate can subsequently reveal how to transform $L^S_{syntax}$ so it accepts as correct the sentences in the corpus of $L^T$—i.e., how to transform it into $L^T_{syntax}$. If we can automate this process, we can greatly aid the work of our world's linguists, the number of which is insufficient to allow the characterization of the myriads of languages and dialects in existence.

**An Example.** Let $L^S = English$ and $L^T = Spanish$, and let us assume that English adjectives always precede the noun they modify, while in Spanish they always post-cede it (an oversimplification, just for illustration purposes). Thus "a hot burner" is correct English, whereas in Spanish we would more readily say "una hornalla caliente".

If we plug the Spanish lexicon into the English parser, and run a representative corpus of (correct) Spanish noun phrases by the resulting hybrid parser, the said precedence property will be declared unsatisfied when hitting phrases such as "una hornalla caliente". The grammar repairing module can then look at the entire list of unsatisfied constraints, and produce the missing syntactic component of $L^T$'s parser by modifying the constraints in $L^S_{syntax}$ so that none are violated by the corpus sentences.

Some of the necessary modifications are easy to identify and to perform, e.g. for accepting "una hornalla caliente" we only need to delete the (English) precedence requirement of adjective over noun (noted $adj < n$). However, subtler modifications may be in order, requiring some statistical analysis, perhaps in a second round of parsing: if in our $L^T$ corpus, which we have assumed representative, *all* adjectives appear after the noun they modify, Spanish is sure to include

the reverse precedence property as in English: $n < adj$. So in this case, not only do we need to delete $adj < n$, but we also need to add $n < adj$.

## 3 Detecting and Characterizing Grammatical Proficiency

The generative power of Womb Grammars can be used to find out the set of linguistic constraints (i.e. the grammar) in use by a person learning a language. For this end, rather than using the grammar of a known related language as in our example above, we propose that of a Universal Womb Grammar which for each type or phrase lists all *possible* properties or constraints. For instance, for every pair of allowable constituents in a phrase (say noun and adjective in a noun phrase), it would list both possible orderings: $noun < adjective$ and $adjective < noun$. By running a student's input through this universal grammar and deleting any constraints not manifest in the input, we are left with a characterization of the student's proficiency. This grammar represents an individual interlanguage system, which is in line with the general perspective in Second Language Acquisition which brands as a mistake the study of the systematic character of one language by comparing it to another[15].

For instance, the order in which children acquire basic grammatical English constructions is fairly predictable. Table 1 shows a widely accepted morpheme acquisition ordering initially postulated by Brown[4] and corroborated, for instance, by de Villiers and de Villiers[18]. According to these results children acquire a series of 14 morphemes in essentially the *same order*, but *not at the same speed*.

| Order | Morpheme |
|---|---|
| 1 | Present progressive (-ing) |
| 2-3 | Prepositions (in, on) |
| 4 | Plural (-s) |
| 5 | Irregular past tense |
| 6 | Possessive (-'s) |
| 7 | Uncontractible copula (is, am are) |
| 8 | Articles (the, a) |
| 9 | Regular past tense (-ed) |
| 10 | Third person present tense, regular (-s) |
| 11 | Third person present tense, irregular |
| 12 | Uncontractible auxilliary (is, am are) |
| 13 | Contractible copula |
| 14 | Contractible auxilliary |

**Table 1.** Order of acquisition of morphemes [5]

To use Womb Grammars for this task, we must find an appropriate set of initial constraints (just as in the example in section 2, the constraints of English

are used as the initial set) from which we can weed out those not satisfied at the child's proficiency level. For the present task, we assume a lexicon that includes all features necessary to morphological analysis as per table 1. Morpheme identification is well studied in the literature for uses such as automatic translation, grammar checking, and spelling correction.

Some of the needed initial constraints we are already familiar with. For instance, if we include the constraint that a noun requires a determiner, any input corpus that violates this constraint will prompt its deletion. The resulting output grammar will be incorrect with respect to adult speech, but will adequately characterize the linguistic competence of the child. We can also output the level of proficiency by noting, for instance, that if the output grammar does contain the said requirement constraint, the child is at least at level 8 of the table. In addition, novel uses of the familiar constraints need to come into play. For instance, precedence is typically tested by itself, but when using it to evaluate language proficiency, it needs to be tested *as a condition* to other constraints, e.g. to check that the copula is not contracted when it is the first word in a question (as in "Is he sick?"), we need to check *if* it precedes all other elements in a question, rather than stipulate that it must precede them.

Other necessary constraints need to be tailored to our task. For instance, several of the constraints in the table boil down to whether some particular feature appears in appropriate contexts within the corpus, so all we have to do is check that this is the case. A case in point: the mastery of irregular verbs, as evidenced by their correct inclusion in the corpus, would indicate acquisition at level 5.

Our prototype implementation of incorrect grammar acquisition incorporates both kinds of constraints. This implementation uses CHRG[7], the grammatical counterpart of CHR[11]. The mastery criteria can be set as 90% accuracy, which was the cutoff proposed by Brown[4]. De Villiers and de Villiers[18] introduced a new "ranking method" based on the relative accuracy with which the morphemes were used in obligatory contexts. Many subsequent studies have used some variant of these approaches, and our system can be readily adapted to different methods of rank ordering.

In order for our results to be precise we need that the input corpus be as representative as possible. For instance, if we are targeting 90% success rate as mastery, we need a sufficient number of relevant utterances such that we can reliably translate the number of occurrences into a percentage. Existing child language acquisition corpora, such as the CHILDES database (http://childes.psy.cmu.edu/), could be used for this purpose.

## 4 Inducing Learning Stages for Languages Other than English

English language acquisition has been very well studied, but there is a dire need for studying the vast majority of languages in existence. An interesting application of Womb Grammars would therefore be to test how much of the English

ordering of morpheme acquisition still holds in any other given language. This could proceed by running our English level proficiency detection system (described in section 3) with the (adequately annotated) morphological lexicon of the language studied. If the results are consistent with those of English, this would indicate a similar learning order. Any differences could be further analyzed in order to find out what the actual acquisition order is in this language. Admittedly this is a rough first approximation, and further work is needed to perfect the approach. There are some constructions that will be learned in wildly different order than in English, for example the plural in Egyptian Arabic is very complex, so children generally do not master it until reaching adolescence.

## 5 Concluding Remarks

We have argued that Womb Grammar Parsing, whose CHRG implementation is described in [9], is an ideal aid to guide a student through language acquisition by using our proposed Universal Womb Grammar. We have also complimented this prototype with a component that can detect a child's morphological level of proficiency in English.

Our work is most probably also applicable for learning English as a second language, as suggested by studies that show that such learners also progress orderly along the same stages[10]. Unlike previous work, which focuses on machine learning techniques (e.g. [20]), our contribution to quality assessment of utterances in a language being learned proceeds through pointing out which linguistic constraints are being violated. From these, an accurate (while probably incorrect by academic standards) grammar of the users language proficiency can be produced, as well as a set of exercises targeting his or her progress.

Admittedly, much more work is needed for a completely encompassing rendition of this first proposal. For instance, we will need to include phonological and semantic considerations in future work. This process will surely further uncover new constraints that need to be added to the familiar ones for the purposes of our research. Reliable evaluation schemes also need to be devised.

To the best of our knowledge, this is the first time the idea of detecting grammatical performance levels for language acquisition materializes through weeding out constraints from a kind of universal constraint-based grammar fragment. With this initial work we hope to stimulate further research along these lines.

## References


1. Blache, P.: Property grammars: A fully constraint-based theory. In: Christiansen, H., Skadhauge, P.R., Villadsen, J. (eds.) CSLP. Lecture Notes in Computer Science, vol. 3438, pp. 1–16. Springer (2004)
2. Blache, P., Guenot, M.L., van Rullen, T.: A corpus-based technique for grammar development. In: Archer, D., Rayson, P., Wilson, A., McEnery, T. (eds.) Proceedings of Corpus Linguistics 2003, University of Lancaster. pp. 123–131 (2003)

3. Boyd, A.A.: Detecting and Diagnosing Grammatical Errors for Beginning Learners of German: From Learner Corpus Annotation to Constraint Satisfaction Problems. Ph.D. thesis, Ohio State University (2012)
4. Brown, R.: A first language: The early stages. George Allen and Unwin (1973)
5. Carroll, D.: Psychology of Language. Thomson/Wadsworth (2008)
6. Charniak, E., Johnson, M.: Coarse-to-fine n-best parsing and maxent discriminative reranking. In: Knight, K., Ng, H.T., Oflazer, K. (eds.) ACL. The Association for Computer Linguistics (2005)
7. Christiansen, H.: Chr grammars. TPLP 5(4-5), 467–501 (2005)
8. Cohen, S.B., Smith, N.A.: Covariance in unsupervised learning of probabilistic grammars. Journal of Machine Learning Research 11, 3017–3051 (2010)
9. Dahl, V., Miralles, J.E.: Womb parsing. In: 9th International Workshop on Constraint Handling Rules (CHR 2012), Budapest, Hungary, September 2012. KU Leuven, Department of Computer Science. pp. 32–40 (2012), Tech. Report CW 624
10. Dulay, H., Burt, M.: Should we teach chidren syntax? In: Language Learning. vol. 23, pp. 245–258 (1973)
11. Frühwirth, T., Raiser, F. (eds.): Constraint Handling Rules: Compilation, Execution, and Analysis (March 2011)
12. Headden, W.P., Johnson, M., McClosky, D.: Improving unsupervised dependency parsing with richer contexts and smoothing. In: HLT-NAACL. pp. 101–109. The Association for Computational Linguistics (2009)
13. Heift, T., Schulze, M.: Errors and Intelligence in Computer-Assisted Language Learning. Parsers and Pedagogues. Routledge, New York, USA (2007)
14. Klein, D., Manning, C.D.: Corpus-based induction of syntactic structure: Models of dependency and constituency. In: Scott, D., Daelemans, W., Walker, M.A. (eds.) ACL. pp. 478–485. ACL (2004)
15. Lakshmanan, U., Selinker, L.: Analysing interlanguage: how do we know what learners know? Second Language Research 14(4), 393–420 (2001)
16. Pereira, F.C.N., Schabes, Y.: Inside-outside reestimation from partially bracketed corpora. In: Thompson, H.S. (ed.) ACL. pp. 128–135. ACL (1992)
17. Pollard, C., Sag, I.A.: Head-driven Phrase Structure Grammars. CSLI, Chicago University Press (1994)
18. de Villiers, J., de Villiers, P.: A cross-sectional study of the acquisition of grammatical morphemes in child speech. In: Journal of Psycholinguistic Research. vol. 2, pp. 267–278 (1973)
19. Wang, M., Smith, N.A., Mitamura, T.: What is the jeopardy model? a quasi-synchronous grammar for qa. In: EMNLP-CoNLL. pp. 22–32. ACL (2007)
20. Yannakoudakis, H., Briscoe, T., Medlock, B.: A new dataset and method for automatically grading esol texts. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. pp. 180–189. HLT '11, Association for Computational Linguistics, Stroudsburg, PA, USA (2011), http://dl.acm.org/citation.cfm?id=2002472.2002496

# What constraints for representing multilinearity in Sign language?

Michael Filhol, Annelies Braffort

LIMSI-CNRS, B.P. 133, F-91403 Orsay cx, France
{michael.filhol,annelies.braffort}@limsi.fr

**Abstract.** In this paper, we present our current investigations on one of the main challenges in Sign Language modelling: multilinearity. We explain the way in which we extract grammatical rules from Sign Language corpus, and how we represent them with constraints. Two kinds are needed: time synchronisation and geometric constraints.

## 1 Introduction on Sign languages

Sign languages (SLs) are visual-gestural natural languages used by the deaf communities. SLs are less-resourced languages and have no writing system. In other words, there are very few reference books describing these languages (grammar rules, etc.), a limited number of dictionaries, and mainly small-sized corpora. Moreover, SL linguistics and processing research is quite recent, and a proper formal description system of the rules governing SL is still to be proposed.

In view of building SL processing systems (recognition, generation, translation...), we need a database of representations for SL elements such as lexical units or grammatical rules. Formal knowledge on these still being in its infancy, this requires thorough investigation beforehand, based on annotation and analysis of SL corpora.

The main challenges in SL modelling are: the relevant use of space in SL discourse, the iconicity that is present at various levels (iconic morphemes, spatial agreement...), and the multilinearity of the language. This paper gives an overview of our current investigations on this last property, started during the European project Dicta-Sign[1]. Section 2 explains how we analyse corpora to build our rules, and section 3 presents our use of constraints to represent them.

## 2 Building rules by linking form to function

The first important fact about SL is that **all body articulators from the waist up**, whether manual, non-manual, skaletal or muscular, can be relevant as they may all—though they do not always—carry meaning. Therefore, all visible activity in a Sign production must be included in the study.

[1] http://www.dictasign.eu

We have observed and partially annotated the French part of the DictaSign corpus [10] and the older LS-Colin corpus [2]. With as few articulators as eyebrows, eye-gaze and both left and right hand activity (and yet the signing body involving many more), it is already clear that **articulators do not align** in any trivial way. That is the second important specificity of SL to deal with.

With these two SL characteristics in mind, we proceed with the corpus study to find systematic links between:

- *surface form features*, i.e. observable/annotated movements or positions of body articulators like a cheek puff, directed eyegaze, hand location change or shoulder line rotation;
- and *linguistic function*, i.e. the possibly language-dependant interpretation of the form, taken to be part of the signer's discourse intent.

Funtions can be identified on all layers of the language; none is linked to a specific one. Any linguistic operation purposely produced may be called a function, whether it inclines to a morphemic, lexical, syntactic or even discourse level. For example, turning a clause into an interrogative, using a known lexical sign, adding modality to a verb or outlining a full speech with intermediate titles are all possible functions. The point for our study is now to find features of the production form that are consistantly used over many occurrences of a linguistic function. Factoring out those form *invariants* raises a candidate set of necessary features for a given function. Describing this set of features under a label for the function raises what we call a *production rule* for the function.

For example, it is already well-documented that in LSF, space locations can be activated by pointing at the target in the signing space with the strong hand, immediately after directing the gaze towards the same target. Other elements are sometimes held on facial articulators or performed by the weak hand, but they do not occur consistantly while the eyegaze-then-point structure is invariant.

Formally describing the invariant above will create a production rule which can be labelled "point and activate locus". We explain how we handle those descriptions in the next section.

## 3 Constraints to represent production rules

There are two problems here to address. One is time synchronisation of parallel activity on a multi-linear signing score, in our example the simultaneity of gaze and index. The second is articulation of body articulators in space, e.g. the hand shape for the pointing gesture and the common direction for both eyegaze and index.

The next section explains the limits in existing models; the following two present the approach we use at LIMSI.

### 3.1 Traditional models

Most models in traditional Sign linguistics do not account for articulator synchronisation issues as fundamental elements of the SL discourse. They rather

define elementary movements and/or sign units that can be concatenated [12, 11]. Some models [8] do promote time to a more essential place in their descriptions. Liddell and Johnson have shown that signs reveal (rather short) periods when all "features" of the hands align to form intentional **postures**, between which **transitions** take place to change from one established posture to the next (see fig. 1a further down). However, all of these models have only really been used (and even designed in the first place) for lexical units of SL, primarily even for "fully lexical signs" (i.e. non depicting). Though, SLs generally enrol more body articulators and even provide with productive ways of creating unlisted iconic units on the fly [3, 7].

Unfortunately, not much exists on SL formal modelling on the clause/sentence level. The most relevant progress is the ViSiCAST[2] and eSign projects and their generative HPSG framework [9]. The basic timing unit in all generative approaches is the lexical item at the bottom of the syntax tree. Thus, any phonetic feature of the body must be borne by a lexical unit composing the clause and the productions burn down to a single tier carrying a lexical sequence. Articulator synchronisation is then only possible using the lexical boundaries.

A first account for multi-track description was given by the "P/C" formalism [6]. It defines two ways of conjoining parts of a signing activity: *partitioning* for parallel specification and *constituting* for sequence. But we argue [5] that it does not allow for abstract description of parts that overlap in time. Section 3.2 presents our approach to tackle this problem with free sets of constraints.

In addition to the timing issues, another question raised is how to specify the signing activity happening at a given time. The problem with L&J's representation is that every moment in time requires the same fixed set of parametric values, numbered a, b, c... in fig. 1a. But similarly to time constraints, we want this new specification to be flexible hence allow for under-specification if it is needed. Section 3.3 below deals with this issue.

### 3.2 Time synchronising constraints

While L&J produce good evidence that much of the signing activity does synchronise with postures, observation reveals that many constructs fall out of the scheme and require a more complex synchronisation technique. Many productions break the set of body articulators into subsets that act more or less independently, each in its own active period (a time interval, TI henceforth). Sometimes the full face acts as a group while the hands are moving; other times a few facial parts take on a different linguistic role and de-synchronise from rest of the face and hands; hands do not always work together; etc. Our proposition is to combine those two sync schemes into a same language for a better coverage of SL linguistic structures. We propose the **AZee** language, combination of the following two.

**Zebedee** is a language designed to specify the posture–transition patterns [4]. A description in Zebedee (or *zebedescription*) specifies:

[2] http://www.visicast.cmp.uea.ac.uk

1. an alternation of “key posture” (KP) and transition units (the horizontal axis in figure 1b below;
2. the articulation that takes place in each of those units (this is further explained in the next section).

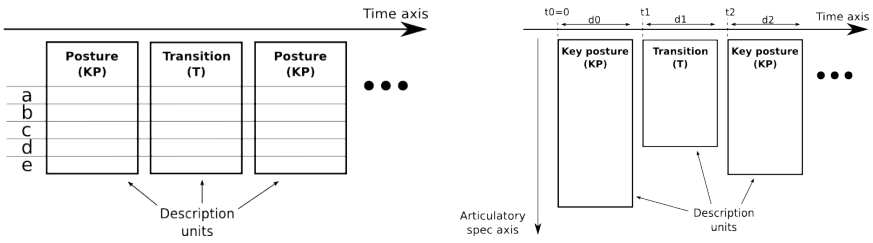



**Fig. 1.** (a) Liddell and Johnson’s phonetic sequence with a fixed number of parameters; (b) Timing units in Zebedee: each unit is described with a variable number of elements

**Azalee** is a language designed to specify the free-hand (i.e. not necessarily aligned) synchronisation patterns for sets of separate time intervals (the TIs) on a timeline. Each Azalee specification (or *azalisting*) is a container for the two following parts, together specifying the intended rule fully:

1. all (usually overlapping) TIs are arranged on the timeline following temporal precedence constraints;
2. each TI specifies the signing activity taking place within.

In part (1), TIs are time-arranged with a *minimal* yet *sufficient* set of constraints in order not to over-specify the surface production and allow for any needed flexibility. The reason for preferring minimality for these sets is that we observe a lot of nested structures, all acting on the body articulators simultanously. Any superfluous constraint on an articulator in a rule will prevent other rules to use it freely, which may result in unsatisfactory productions.

The temporal precedence constraints can be of any form, whether strict or flexible, of precedence or duration, over full TIs using Allen logic operators [1] or over single boundaries. In part (2), a TI can either make use of a nested Azalee description, or use Zebedee for a simpler sync pattern.

Figure 2 illustrates our example rule. Its Azalee pattern (outer box) is composed of 2 TIs, ‘eye gaze’ and ‘pointing sign’, which are zebedescriptions (KP–T alternation patterns). This flexible combination of two nestable timing patterns allows to account for complex synchronisation between body articulators.

### 3.3 Articulatory constraints

Fixed sets of parameters or features are not satisfactory to describe articulatory postures. As we have explained:

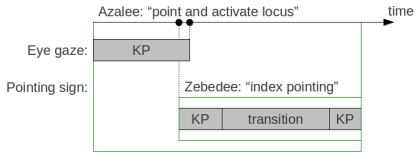



**Fig. 2.** AZee: an example of Azalee and Zebedee combination

– on the one hand, articulators are not all linguistically relevant all the time, thus including all of them in a fixed-set model would lead to a lot of over-specification;
– on the other hand however, any articulator of the upper-part of the body can carry meaning at some point, thus any exclusion from the fixed set leads to possible under-specification.

To avoid both problems, we choose sets of necessary and sufficient constraints to specify both as much and as little as needed at any moment in time. In other words recalling the minimality of the sets of time constraints, we aim at minimal sets for the articulatory constraints. The constraints are mainly of two types: orientation (to align bones along vectors) and placement (to bring a body location to a point in space or in contact with another).

For example, throughout the pointing gesture, we need to extend the index, which takes at least the following constraint, literally aligning the proximal index bone of the strong hand (side `S` below) in the same direction as the second:

```
Orient DIR!index(S,1) along DIR!index(S,2)
```

Placing the hand—the palm site (`@PA`) of the strong hand—in front of the body (forward fron the sternum site `@ST`) for the initial key posture will be written:

```
Place @WR(S) at @ST + <FWD | medium>
```

Each timing unit holds a conjunction of such constraints, and the resulting set is as large as necessary and as small as sufficient. Figure 1b illustrates this approach and contrasts with previous figure 1a. The vertical depth needed to specify body articulation in a zebedescription may vary in time, according to the number of articulatory constraints. Note that they are fundamentally geometric; all typical geometric transformations (translations, symmetries, etc.) are supported.

Also, expressions can contain context references enclosed in square brackets to account for significant (often iconic) context variations. For our example above, the distal bone of the pointing finger `!index(S,3)` must be oriented to point at the target, which is a context dependency. The constraint will be something like[3]:

```
Orient DIR!index(S,3) along <ORIGIN!index(S,3), [target]>
```

[3] Operator `<..., ...>` creates a vector from two geometric points.

## 4 Prospects

We have presented the issues in formalising systematic links between form and function. Our methodology, based on SL corpus analysis, is applied regardless of the level, from the sub-lexical level to discourse structure. The formalisation of the production rules uses the AZee model, which allows to express Sign articulator synchronisation with time constraints, and body postures and movement with geometric constraints.

While the Zebedee part of this model is already well assessed (2,000+ LSF signs described), we have so far only really explored a dozen Azalee structures. We intend to continue this work to increase this number. A thorough evaluation will then be a starting point for implementation of AZee rules in Sign synthesis software applications, a longer-term prospect being text-to-SL translation.

In parallel and in order to collect 3d/spatial data, we will have to collect a new kind of SL corpus, built using a motion capture system. This will allow us to conduct similar studies on the spatial organisation of SL and then provide with a finer representation of the signing space and its use in SL discourse.

## References


1. Allen, J. F.: Maintaining Knowledge about Temporal Intervals. ACM 26:11, pp. 832–843 (1983)
2. Braffort, A., Chtelat-Pel, .: Analysis and Description of Blinking in French Sign Language for Automatic Generation, in Gesture and Sign Language in Human-Computer Interaction and Embodied Communication, LNCS/LNAI vol. 7206, Springer (2012, tbp)
3. Cuxac, C.: Langue des signes franaise, les voies de l'iconicit. Ophrys, Paris (2000)
4. Filhol, M.: Zebedee: a lexical description model for Sign Language synthesis. LIMSI-CNRS, Orsay (2009)
5. Filhol, M.: Combining two synchronisation methods in a linguistic model to describe Sign Language, in Gesture and Sign Language in Human-Computer Interaction and Embodied Communication, LNCS/LNAI vol. 7206, Springer (2012, tbp)
6. Huenerfauth, M.: Generating American Sign Language classifier predicates for English-to-ASL machine translation. PhD thesis, University of Pennsylvania (2006)
7. Johnston, T., Schembri, A.: Australian Sign Language (Auslan): An Introduction to Sign Language Linguistics. Cambridge University Press (2007)
8. Liddell, S. K., Johnson, R. E.: American Sign Language, the phonological base. Sign Language studies 64, pp. 195–278, Cambridge University press (1989)
9. Marshall, I., Sfr, .: Sign Language generation in an ALE HPSG, in Proceedings of the HPSG04 conference, Leuven, Belgium (2004)
10. Matthes, S., Hanke, T., Storz, J., Efthimiou, E., Dimiou,, N., Karioris, P., Braffort, A., Choisier, A., Pelhate, J., Safar, E.: Elicitation tasks and materials designed for DictaSign's multi-lingual corpus. LREC, Corpora and Sign Language technologies, pp. 158–163 (2010)
11. Prillwitz, S., Leven, R., Zienert, H., Hanke, T., Henning, J.: HamNoSys version 2.0, an introductory guide. International studies on Sign Language communication of the Deaf, Signum press, Hamburg, vol. 5 (1989)
12. Stokoe, W.: Sign Language structure: an outline of the visual communication system of the American Deaf Studies. Linguistics, occasional papers, vol. 8 (1960)

## Author Index