A Relational Syntax-Semantics Interface Based on Dependency Grammar

Ralph Debusmann Denys Duchier* Alexander Koller Marco Kuhlmann Gert Smolka Stefan Thater

Saarland University, Saarbrücken, Germany *LORIA, Nancy, France

{rade|kuhlmann|smolka}@ps.uni-sb.de, duchier@loria.fr, {koller|stth}@coli.uni-sb.de

Abstract

We propose a syntax-semantics interface that realises the mapping between syntax and semantics as a relation and does not make functionality assumptions in either direction. This interface is stated in terms of Extensible Dependency Grammar (XDG), a grammar formalism we newly specify. XDG's constraint-based parser supports the concurrent flow of information between any two levels of linguistic representation, even when only partial analyses are available. This generalises the concept of underspecification.

1 Introduction

A key assumption of traditional syntax-semantics interfaces, starting with (Montague, 1974), is that the mapping from syntax to semantics is *functional*, i. e. that once we know the syntactic structure of a sentence, we can deterministically compute its semantics.

Unfortunately, this assumption is typically not justified. Ambiguities such as of quantifier scope or pronominal reference are genuine *semantic* ambiguities; that is, even a syntactically unambiguous sentence can have multiple semantic readings. Conversely, a common situation in natural language generation is that one semantic representation can be verbalised in multiple ways. This means that the relation between syntax and semantics is not functional at all, but rather a true *m*-to-*n* relation.

There is a variety of approaches in the literature on syntax-semantics interfaces for coping with this situation, but none of them is completely satisfactory. One way is to recast semantic ambiguity as syntactic ambiguity by compiling semantic distinctions into the syntax (Montague, 1974; Steedman, 1999; Moortgat, 2002). This restores functionality, but comes at the price of an artificial blow-up of syntactic ambiguity. A second approach is to assume a non-deterministic mapping from syntax to semantics as in generative grammar (Chomsky, 1965), but it is not always obvious how to reverse the relation, e. g. for generation. Finally, underspecification (Egg et al., 2001; Gupta and Lamping, 1998; Copestake et al., 2004) introduces a new level of representation, which can be computed functionally from a syntactic analysis and encapsulates semantic ambiguity in a way that supports the enumeration of all semantic readings by need.

In this paper, we introduce a completely relational syntax-semantics interface, building upon the underspecification approach. We assume a set of linguistic *dimensions*, such as (syntactic) immediate dominance and predicate-argument structure; a grammatical analysis is a tuple with one component for each dimension, and a grammar describes a set of such tuples. While we make no *a priori* functionality assumptions about the relation of the linguistic dimensions, functional mappings can be obtained as a special case. We formalise our syntax-semantics interface using *Extensible Dependency Grammar* (XDG), a new grammar formalism which generalises earlier work on Topological Dependency Grammar (Duchier and Debusmann, 2001).

The relational syntax-semantics interface is supported by a parser for XDG based on constraint programming. The crucial feature of this parser is that it supports the concurrent flow of possibly partial information between any two dimensions: once additional information becomes available on one dimension, it can be propagated to any other dimension. Grammaticality conditions and preferences (e. g. selectional restrictions) can be specified on their natural level of representation, and inferences on each dimension can help reduce ambiguity on the others. This generalises the idea of underspecification, which aims to represent and reduce ambiguity through inferences on a *single* dimension only.

The structure of this paper is as follows: in Section 2, we give the general ideas behind XDG, its formal definition, and an overview of the constraint-based parser. In Section 3, we present the relational syntax-semantics interface, and go through examples that illustrate its operation. Section 4 shows how the semantics side of our syntax-semantics interface can be precisely related to mainstream semantics research. We summarise our results and point to further work in Section 5.

2 Extensible Dependency Grammar

This section presents Extensible Dependency Grammar (XDG), a description-based formalism for dependency grammar. XDG generalizes previous work on Topological Dependency Grammar (Duchier and Debusmann, 2001), which focussed on word order phenomena in German.

2.1 XDG in a Nutshell

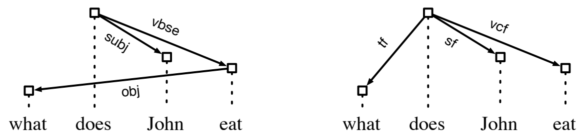
XDG is a description language over finite labelled graphs. It is able to talk about two kinds of constraints on these structures: The *lexicon* of an XDG grammar describes properties local to individual nodes, such as valency. The grammar's *principles* express constraints global to the graph as a whole, such as treeness. Well-formed analyses are graphs that satisfy all constraints.

An XDG grammar allows the characterisation of linguistic structure along several *dimensions* of description. Each dimension contains a separate graph, but all these graphs share the same set of nodes. Lexicon entries synchronise dimensions by specifying the properties of a node on all dimensions at once. Principles can either apply to a single dimension (*one-dimensional*), or constrain the relation of several dimensions (*multi-dimensional*).

Consider the example in Fig. 1, which shows an analysis for a sentence of English along two dimensions of description, immediate dominance (ID) and linear precedence (LP). The principles of the underlying grammar require both dimensions to be trees, and the LP tree to be a "flattened" version of the ID tree, in the sense that whenever a node v is a transitive successor of a node u in the LP tree, it must also be a transitive successor of u in the ID tree. The given lexicon specifies the potential incoming and required outgoing edges for each word on both dimensions. The word *does*, for example, accepts no incoming edges on either dimension and must therefore be at the root of both the ID and the LP tree. It is required to have outgoing edges to a subject (subj) and a verb base form (vbse) in the ID tree, needs fillers for a subject (sf) and a verb complement field (vcf) in the LP tree, and offers an optional field for topicalised material (tf). All these constraints are satisfied by the analysis, which is thus well-formed.

2.2 Formalisation

Formally, an XDG grammar is built up of dimensions, principles, and a lexicon, and characterises a set of well-formed analyses.



word	in_{ID}	out_{ID}	in_{LP}	out_{LP}
what	{obj?}	{}	{tf?}	{}
does	{}	{subj, vbse}	{}	{tf?, sf, vcf}
John	{subj?, obj?}	{}	{sf?, of?}	{}
eat	{vbse?}	{obj}	{vcf?}	{}

Figure 1: XDG analysis of "what does John eat"

A *dimension* is a tuple $D = (Lab, Fea, Val, Pri)$ of a set *Lab* of edge labels, a set *Fea* of features, a set *Val* of feature values, and a set of one-dimensional principles *Pri*. A *lexicon* for the dimension D is a set $Lex \subseteq Fea \to Val$ of total feature assignments (or lexical entries). A *D-structure*, representing an analysis on dimension D, is a triple (V, E, F) of a set V of nodes, a set $E \subseteq V \times V \times Lab$ of directed labelled edges, and an assignment $F : V \to (Fea \to Val)$ of lexical entries to nodes. V and E form a graph. We write Str_D for the set of all possible D-structures. The principles characterise subsets of Str_D that have further dimension-specific properties, such as being a tree, satisfying assigned valencies, etc. We assume that the elements of *Pri* are finite representations of such subsets, but do not go into details here; some examples are shown in Section 3.2.

An XDG *grammar* $((Lab_i, Fea_i, Val_i, Pri_i)_{i=1}^{n}, Pri, Lex)$ consists of n dimensions, multi-dimensional principles *Pri*, and a lexicon *Lex*. An XDG *analysis* $(V, E_i, F_i)_{i=1}^{n}$ is an element of $Ana = Str_1 \times \cdots \times Str_n$ where all dimensions share the same set of nodes V.

Multi-dimensional principles work just like one-dimensional principles, except that they specify subsets of *Ana*, i. e. couplings between dimensions (e. g. the flattening principle between ID and LP in Section 2.1). The lexicon $Lex \subseteq Lex_1 \times \cdots \times Lex_n$ constrains all dimensions at once. An XDG analysis is licenced by *Lex* iff $(F_1(w), \ldots, F_n(w)) \in Lex$ for every node $w \in V$.

In order to compute analyses for a given input, we model it as a set of *input constraints* (*Inp*), which again specify a subset of *Ana*. The parsing problem for XDG is then to find elements of *Ana* that are licenced by *Lex* and consistent with *Inp* and *Pri*. Note that the term "parsing problem" is traditionally used only for inputs that are sequences of words, but we can easily represent surface realisation as a "parsing" problem in which *Inp* specifies a semantic dimension; in this case, a "parser" would compute analyses that contain syntactic dimensions from which we can read off a surface sentence.

2.3 Constraint Solver

The parsing problem of XDG has a natural reading as a constraint satisfaction problem (CSP) (Apt, 2003) on finite sets of integers; well-formed analyses correspond to the solutions of this problem. The transformation, whose details we omit due to lack of space, closely follows previous work on axiomatising dependency parsing (Duchier, 2003) and includes the use of the *selection constraint* to efficiently handle lexical ambiguity.

We have implemented a constraint solver for this CSP using the Mozart/Oz programming system (Smolka, 1995; Mozart Consortium, 2004). This solver does a search for a satisfying variable assignment. After each case distinction (*distribution*), it performs simple inferences that restrict the ranges of the finite set variables and thus reduce the size of the search tree (*propagation*). The successful leaves of the search tree correspond to XDG analyses, whereas the inner nodes correspond to *partial analyses*. In these cases, the current constraints are too weak to specify a complete analysis, but they already express that some edges or feature values must be present, and that others are excluded. Partial analyses will play an important role in Section 3.3.

Because propagation operates on all dimensions concurrently, the constraint solver can frequently infer information about one dimension from information on another, if there is a multi-dimensional principle linking the two dimensions. These inferences take place while the constraint problem is being solved, and they can often be drawn before the solver commits to any single solution.

Because XDG allows us to write grammars with completely free word order, XDG solving is an NP-complete problem (Koller and Striegnitz, 2002). This means that the worst-case complexity of the solver is exponential, but the average-case complexity is still bearable for many grammars we have experimented with, and we hope there are useful fragments of XDG that would guarantee worst-case polynomial complexity.

3 A Relational Syntax-Semantics Interface

Now that we have the formal and processing frameworks in place, we can define a relational syntax-semantics interface for XDG. We will first show how we encode semantics within the XDG framework. Then we will present an example grammar (including some principle definitions), and finally go through an example that shows how the relationality of the interface, combined with the concurrency of the constraint solver, supports the flow of information between different dimensions.

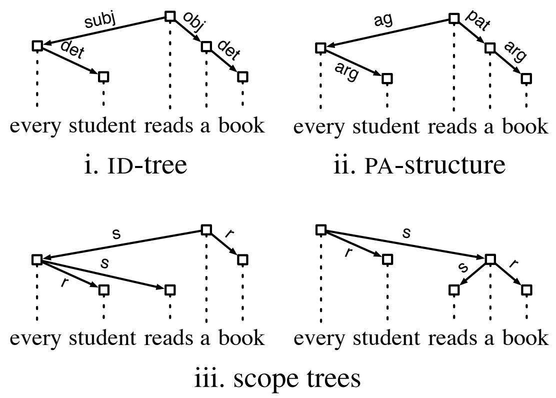


Figure 2: Two analyses for the sentence "every student reads a book."

3.1 Representing Meaning

We represent meaning within XDG on two dimensions: one for *predicate-argument structure* (PA), and one for *scope* (SC). The function of the PA dimension is to abstract over syntactic idiosyncrasies such as active-passive alternations or dative shifts, and to make certain semantic dependencies e. g. in control constructions explicit; it deals with concepts such as agent and patient, rather than subject and object. The purpose of the SC dimension is to reflect the structure of a logical formula that would represent the semantics, in terms of scope and restriction. We will make this connection explicit in Section 4. In addition, we assume an ID dimension as above. We do not include an LP dimension only for ease of presentation; it could be added completely orthogonally to the three dimensions we consider here.

While one ID structure will typically correspond to one PA structure, each PA structure will typically be consistent with multiple SC structures, because of scope ambiguities. For instance, Fig. 2 shows the unique ID and PA structures for the sentence "Every student reads a book." These structures (and the input sentence) are consistent with the two possible SC-structures shown in (iii). Assuming a Davidsonian event semantics, the two SC trees (together with the PA-structure) represent the two readings of the sentence:

- $\lambda e.\forall x.student(x) \rightarrow \exists y.book(y) \wedge read(e,x,y)$
- $\lambda e.\exists y.book(y) \wedge \forall x.student(x) \rightarrow read(e,x,y)$

3.2 A Grammar for a Fragment of English

The lexicon for an XDG grammar for a small fragment of English using the ID, PA, and SC dimensions is shown in Fig. 3. Each row in the table specifies a (unique) lexical entry for each part of speech (determiner, common noun, proper noun, transitive verb and preposition); there is no lexical ambiguity in this grammar. Each column specifies a feature. The meaning of the features will be explained together with the principles that use them.

The ID dimension uses the edge labels $Lab_{\text{ID}} = \{\mathsf{det}, \mathsf{subj}, \mathsf{obj}, \mathsf{prep}, \mathsf{pcomp}\}$ resp. for determined common noun,[1] subject, object, preposition, and complement of a preposition. The PA dimension uses $Lab_{\text{PA}} = \{\mathsf{ag}, \mathsf{pat}, \mathsf{arg}, \mathsf{quant}, \mathsf{mod}, \mathsf{instr}\}$, resp. for agent, patient, argument of a modifier, common noun pertaining to a quantifier, modifier, and instrument; and SC uses $Lab_{\text{SC}} = \{\mathsf{r}, \mathsf{s}, \mathsf{a}\}$ resp. for restriction and scope of a quantifier, and for an argument.

The grammar also contains three one-dimensional principles (tree, dag, and valency), and three multi-dimensional principles (linking, co-dominance, and contra-dominance).

Tree and dag principles. The tree principle restricts ID and SC structures to be trees, and the dag principle restricts PA structures to be directed acyclic graphs.

Valency principle. The valency principle, which we use on all dimensions, states that the incoming and outgoing edges of each node must obey the specifications of the *in* and *out* features. The possible values for each feature in_d and out_d are subsets of $Lab_d \times \{!, ?, *\}$. $\ell!$ specifies a mandatory edge with label ℓ, $\ell?$ an optional one, and $\ell*$ zero or more.

Linking principle. The linking principle for dimensions d_1, d_2 constrains how dependents on d_1 may be realised on d_2. It assumes a feature $link_{d_1,d_2}$ whose values are functions that map labels from Lab_{d_1} to sets of labels from Lab_{d_2}, and is specified by the following implication:

$$v \xrightarrow{l}_{d_1} v' \quad \Rightarrow \quad \exists l' \in link_{d_1,d_2}(v)(l) : v \xrightarrow{l'}_{d_2} v'$$

Our grammar uses this principle with the *link* feature to constrain the realisations of PA-dependents in the ID dimension. In Fig. 2, the agent (ag) of *reads* must be realised as the subject (subj), i. e.

$$reads \xrightarrow{\mathsf{ag}}_{\text{PA}} every \quad \Rightarrow \quad reads \xrightarrow{\mathsf{subj}}_{\text{ID}} every$$

Similarly for the patient and the object. There is no instrument dependent in the example, so this part of the *link* feature is not used. An ergative verb would use a *link* feature where the subject realises the patient; Control and raising phenomena can also be modelled, but we cannot present this here.

[1]We assume on all dimensions that determiners are the heads of common nouns. This makes for a simpler relationship between the syntactic and semantic dimensions.

Co-dominance principle. The co-dominance principle for dimensions d_1, d_2 relates edges in d_1 to dominance relations in the same direction in d_2. It assumes a feature $codom_{d_1,d_2}$ mapping labels in Lab_{d_1} to sets of labels in Lab_{d_2} and is specified as

$$v \xrightarrow{l}_{d_1} v' \quad \Rightarrow \quad \exists l' \in codom_{d_1,d_2}(v)(l) : v \xrightarrow{l'} \rightarrow^*_{d_2} v'$$

Our grammar uses the co-dominance principle on dimension PA and SC to express, e. g., that the propositional contribution of a noun must end up in the restriction of its determiner. For example, for the determiner *every* of Fig. 2 we have:

$$every \xrightarrow{\mathsf{quant}}_{\text{PA}} student \quad \Rightarrow \quad every \xrightarrow{\mathsf{r}} \rightarrow^*_{\text{SC}} student$$

Contra-dominance principle. The contra-dominance principle is symmetric to the co-dominance principle, and relates edges in dimension d_1 to dominance edges into the opposite direction in dimension d_2. It assumes a feature $contradom_{d_1,d_2}$ mapping labels of Lab_{d_1} to sets of labels from Lab_{d_2} and is specified as

$$v \xrightarrow{l}_{d_1} v' \quad \Rightarrow$$
$$\exists l' \in contradom_{d_1,d_2}(v)(l) : v' \xrightarrow{l'} \rightarrow^*_{d_2} v$$

Our grammar uses the contra-dominance principle on dimensions PA and SC to express, e. g., that predicates must end up in the scope of the quantifiers whose variables they refer to. Thus, for the transitive verb *reads* of Fig. 2, we have:

$$reads \xrightarrow{\mathsf{ag}}_{\text{PA}} every \quad \Rightarrow \quad every \xrightarrow{\mathsf{s}} \rightarrow^*_{\text{SC}} reads$$
$$reads \xrightarrow{\mathsf{pat}}_{\text{PA}} a \quad \Rightarrow \quad a \xrightarrow{\mathsf{s}} \rightarrow^*_{\text{SC}} reads$$

3.3 Syntax-Semantics Interaction

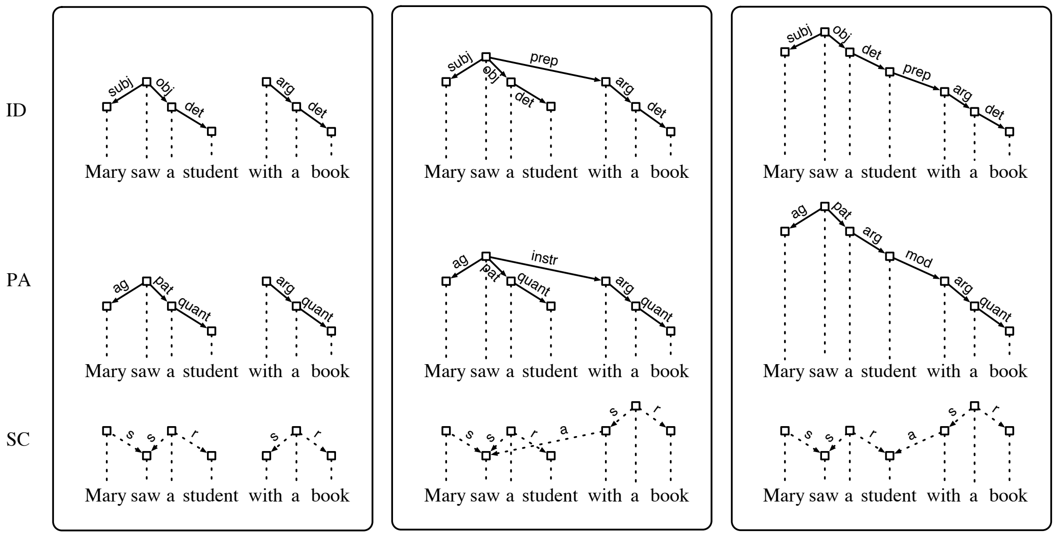
It is important to note at this point that the syntax-semantics interface we have defined is indeed relational. Each principle declaratively specifies a set of admissible analyses, i. e. a relation between the structures for the different dimensions, and the analyses that the complete grammar judges grammatical are simply those that satisfy all principles. The role of the lexicon is to provide the feature values which parameterise the principles defined above.

The constraint solver complements this relationality by supporting the use of the principles to move information between any two dimensions. If, say, the left-hand side of the linking principle is found to be satisfied for dimension d_1, a propagator will infer the right-hand side and add it to dimension d_2. Conversely, if the solver finds that the right-hand side

	in_{ID}	out_{ID}	in_{PA}	out_{PA}	in_{SC}	out_{SC}
DET	{subj?, obj?, pcomp?}	{det!}	{ag?, pat?, arg?}	{quant!}	{r?, s?, a?}	{r!, s!}
CN	{det?}	{prep∗}	{quant?}	{mod?}	{r?, s?, a?}	{}
PN	{subj?, obj?, pcomp?}	{prep∗}	{ag?, pat?, arg?}	{mod?}	{r?, s?, a?}	{r?, s!}
TV	{}	{subj!, obj!, prep∗}	{}	{ag!, pat!, instr?}	{r?, s?, a?}	{}
PREP	{prep?}	{pcomp!}	{mod?, instr?}	{arg!}	{r?, s?, a?}	{a!}

	link	*codom*	*contradom*
DET	{quant ↦ {det}}	{quant ↦ {r}}	{}
CN	{mod ↦ {prep}}	{}	{mod ↦ {a}}
PN	{mod ↦ {prep}}	{}	{mod ↦ {a}}
TV	{ag ↦ {subj}, pat ↦ {obj}, instr ↦ {prep}}	{}	{ag ↦ {s}, pat ↦ {s}, instr ↦ {a}}
PREP	{arg ↦ {pcomp}}	{}	{arg ↦ {s}}

Figure 3: The example grammar fragment



i. Partial analysis ii. verb attachment iii. noun attachment

Figure 4: Partial description (left) and two solutions (right) for "Mary saw a student with a book."

must be false for d_2, the negation of the left-hand side is inferred for d_1. By letting principles interact concurrently, we can make some very powerful inferences, as we will demonstrate with the example sentence "Mary saw a student with a book," some partial analyses for which are shown in Fig. 4.

Column (i) in the figure shows the state after the constraint solver finishes its initial propagation, at the root of the search tree. Even at this point, the valency and treeness principles have conspired to establish an almost complete ID-structure. Through the linking principle, the PA-structure has been determined similarly closely. The SC-structure is still mostly undetermined, but by the co- and contra-dominance principles, the solver has already established that some nodes must *dominate* others: A dotted edge with label s in the picture means that the solver knows there must be a path between these two nodes which starts with an s-edge. In other words, the solver has computed a large amount of semantic information from an incomplete syntactic analysis.

Now imagine some external source tells us that *with* is a mod-child of *student* on PA, i. e. the analysis in (iii). This information could come e. g. from a statistical model of selectional preferences, which will judge this edge much more probable than an instr-edge from the verb to the preposition (ii). Adding this edge will trigger additional inferences through the linking principle, which can now infer that *with* is a prep-child of *student* on ID. In the other direction, the solver will infer more dominances on SC. This means that semantic information can be used to disambiguate syntactic ambiguities, and semantic information such as selectional preferences can be stated on their natural level of representation, rather than be forced into the ID dimension directly.

Similarly, the introduction of new edges on SC could trigger a similar reasoning process which would infer new PA-edges, and thus indirectly also new ID-edges. Such new edges on SC could come from inferences with world or discourse knowledge

(Koller and Niehren, 2000), scope preferences, or interactions with information structure (Duchier and Kruijff, 2003).

4 Traditional Semantics

Our syntax-semantics interface represents semantic information as graphs on the PA and SC dimensions. While this looks like a radical departure from traditional semantic formalisms, we consider these graphs simply an alternative way of presenting more traditional representations. We devote the rest of the paper to demonstrating that a pair of a PA and a SC structure can be interpreted as a Montague-style formula, and that a partial analysis on these two dimensions can be seen as an underspecified semantic description.

4.1 Montague-style Interpretation

In order to extract a standard type-theoretic expression from an XDG analysis, we assign each node v two semantic values: a *lexical* value $\mathfrak{L}(v)$ representing the semantics of v itself, and a *phrasal* value $\mathfrak{P}(v)$ representing the semantics of the entire SC-subtree rooted at v. We use the SC-structure to determine functor-argument relationships, and the PA-structure to establish variable binding.

We assume that nodes for determiners and proper names introduce unique individual variables ("indices"). Below we will write $\langle\langle v\rangle\rangle$ to refer to the index of the node v, and we write $\downarrow\ell$ to refer to the node which is the ℓ-child of the current node in the appropriate dimension (PA or SC). The semantic lexicon is defined as follows; "$\mathfrak{L}(\mathrm{w})$" should be read as "$\mathfrak{L}(v)$, where v is a node for the word w".

$$\begin{aligned}\mathfrak{L}(\mathrm{a}) &= \lambda P\lambda Q\lambda e.\exists x(P(x)\wedge Q(x)(e))\\ \mathfrak{L}(\mathrm{book}) &= book'\\ \mathfrak{L}(\mathrm{with}) &= \lambda P\lambda x.(with'(\langle\langle\downarrow\mathsf{arg}\rangle\rangle)(x)\wedge P(x))\\ \mathfrak{L}(\mathrm{reads}) &= read'(\langle\langle\downarrow\mathsf{pat}\rangle\rangle)(\langle\langle\downarrow\mathsf{ag}\rangle\rangle)\end{aligned}$$

Lexical values for other determiners, common nouns, and proper names are defined analogously. Note that we do not formally distinguish event variables from individual variables. In particular, $\mathfrak{L}(with)$ can be applied to either nouns or verbs, which both have type $\langle e,t\rangle$.

We assume that no node in the SC-tree has more than one child with the same edge label (which our grammar guarantees), and write $n(\ell_1,\ldots,\ell_k)$ to indicate that the node n has SC-children over the edge labels $\ell_1,\ldots,\ell_k$. The phrasal value for n is defined (in the most complex case) as follows:

$$\mathfrak{P}(n(\mathsf{r},\mathsf{s})) = \mathfrak{L}(n)(\mathfrak{P}(\downarrow\mathsf{r}))(\lambda\langle\langle n\rangle\rangle.\mathfrak{P}(\downarrow\mathsf{s}))$$

This rule implements Montague's rule of quantification (Montague, 1974); note that $\lambda\langle\langle n\rangle\rangle$ is a binder for the variable $\langle\langle n\rangle\rangle$. Nodes that have no s-children are simply functionally applied to the phrasal semantics of their children (if any).

By way of example, consider the left-hand SC-structure in Fig. 2. If we identify each node by the word it stands for, we get the following phrasal value for the root of the tree:

$$\mathfrak{L}(a)(\mathfrak{L}(book))(\lambda x.\mathfrak{L}(every)(\mathfrak{L}(student)\\ (\lambda y.read'(y)(x)))),$$

where we write x for $\langle\langle a\rangle\rangle$ and y for $\langle\langle every\rangle\rangle$. The arguments of $read'$ are x and y because *every* and *a* are the arg and pat children of *reads* on the PA-structure. After replacing the lexical values by their definitions and beta-reduction, we obtain the familiar representation for this semantic reading, as shown in Section 3.1.

4.2 Underspecification

It is straightforward to extend this extraction of type-theoretic formulas from fully specified XDG analyses to an extraction of underspecified semantic descriptions from partial XDG analyses. We will briefly demonstrate this here for descriptions in the CLLS framework (Egg et al., 2001), which supports this most easily. Other underspecification formalisms could be used too.

Consider the partial SC-structure in Fig. 5, which could be derived by the constraint solver for the sentence from Fig. 2. We can obtain a CLLS constraint from it by first assigning to each node of the SC-structure a lexical value, which is now a part of the CLLS constraint (indicated by the dotted ellipses). Because *student* and *book* are known to be r-daughters of *every* and *a* on SC, we plug their CLLS constraints into the r-holes of their mothers' constraints. Because we know that *reads* must be dominated by the s-children of the determiners, we add the two (dotted) dominance edges to the constraint. Finally, variable binding is represented by the binding constraints drawn as dashed arrows, and can be derived from PA exactly as above.

5 Conclusion

In this paper, we have shown how to build a fully relational syntax-semantics interface based on XDG. This new grammar formalism offers the grammar developer the possibility to represent different kinds of linguistic information on separate dimensions that can be represented as graphs. Any two dimensions can be linked by multi-dimensional principles,

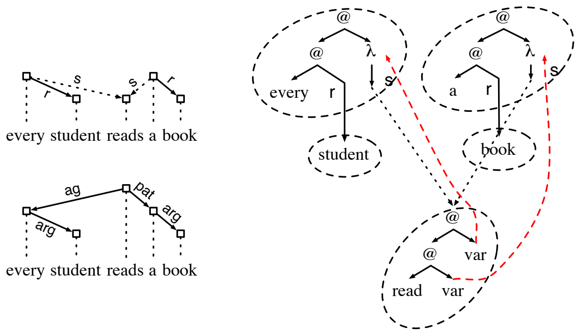


Figure 5: A partial SC-structure and its corresponding CLLS description.

which mutually constrain the graphs on the two dimensions. We have shown that a parser based on concurrent constraint programming is capable of inferences that restrict ambiguity on one dimension based on newly available information on another.

Because the interface we have presented makes no assumption that any dimension is more “basic” than another, there is no conceptual difference between parsing and generation. If the input is the surface sentence, the solver will use this information to compute the semantic dimensions; if the input is the semantics, the solver will compute the syntactic dimensions, and therefore a surface sentence. This means that we get bidirectional grammars for free.

While the solver is reasonably efficient for many grammars, it is an important goal for the future to ensure that it can handle large-scale grammars. One way in which we hope to achieve this is to identify fragments of XDG with provably polynomial parsing algorithms, and which contain most useful grammars. Such grammars would probably have to specify word orders that are not completely free, and we would have to control the combinatorics of the different dimensions (Maxwell and Kaplan, 1993). One interesting question is also whether different dimensions can be compiled into a single dimension, which might improve efficiency in some cases, and also sidestep the monostratal vs. multistratal distinction.

The crucial ingredient of XDG that make relational syntax-semantics processing possible are the declaratively specified principles. So far, we have only given some examples for principle specifications; while they could all be written as Horn clauses, we have not committed to any particular representation formalism. The development of such a representation formalism will of course be extremely important once we have experimented with more powerful grammars and have a stable intuition about what principles are needed.

At that point, it would also be highly interesting to define a (logic) formalism that generalises both XDG and dominance constraints, a fragment of CLLS. Such a formalism would make it possible to take over the interface presented here, but use dominance constraints directly on the semantics dimensions, rather than via the encoding into PA and SC dimensions. The extraction process of Section 4.2 could then be recast as a principle.

References

K. Apt. 2003. *Principles of Constraint Programming*. Cambridge University Press.

N. Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA.

A. Copestake, D. Flickinger, C. Pollard, and I. Sag. 2004. Minimal recursion semantics. an introduction. *Journal of Language and Computation*. To appear.

D. Duchier and R. Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *ACL 2001*, Toulouse.

D. Duchier and G.-J. M. Kruijff. 2003. Information structure in topological dependency grammar. In *EACL 2003*.

D. Duchier. 2003. Configuration of labeled trees under lexicalized constraints and principles. *Research on Language and Computation*, 1(3–4):307–336.

M. Egg, A. Koller, and J. Niehren. 2001. The Constraint Language for Lambda Structures. *Logic, Language, and Information*, 10:457–485.

V. Gupta and J. Lamping. 1998. Efficient linear logic meaning assembly. In *COLING/ACL 1998*.

A. Koller and J. Niehren. 2000. On underspecified processing of dynamic semantics. In *Proceedings of COLING-2000*, Saarbrücken.

A. Koller and K. Striegnitz. 2002. Generation as dependency parsing. In *ACL 2002*, Philadelphia/USA.

J. T. Maxwell and R. M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590.

R. Montague. 1974. The proper treatment of quantification in ordinary english. In *Formal Philosophy. Selected Papers of Richard Montague*, pages 247–271. Yale University Press, New Haven and London.

M. Moortgat. 2002. Categorial grammar and formal semantics. In *Encyclopedia of Cognitive Science*. Nature Publishing Group, MacMillan. To appear.

Mozart Consortium. 2004. The Mozart-Oz website. http://www.mozart-oz.org/.

G. Smolka. 1995. The Oz Programming Model. In *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag.

M. Steedman. 1999. Alternating quantifier scope in CCG. In *Proc. 37th ACL*, pages 301–308.