The Constraint Language for Lambda Structures

Markus Egg (egg@coli.uni-sb.de) and
Alexander Koller (koller@coli.uni-sb.de)
Dept. of Computational Linguistics, University of the Saarland

Joachim Niehren (niehren@ps.uni-sb.de)
Programming Systems Lab, University of the Saarland

Abstract. This paper presents the Constraint Language for Lambda Structures (CLLS), a first-order language for semantic underspecification that conservatively extends dominance constraints. It is interpreted over lambda structures, tree-like structures that encode λ-terms. Based on CLLS, we present an underspecified, uniform analysis of scope, ellipsis, anaphora, and their interactions. CLLS solves a variable capturing problem that is omnipresent in scope underspecification and can be processed efficiently.

Keywords: Underspecification, scope ambiguity, ellipsis, anaphora, tree descriptions.

1. Introduction

Underspecification is a recent approach to the issue of combinatorial explosion caused by ambiguity. Its key idea is to derive a single, compact *description* of all readings instead of the (exponential number of) readings themselves. For as long as possible, later processing steps operate on these descriptions; readings are only enumerated by need.

One type of ambiguity for which underspecification has been investigated particularly well is scope ambiguity. Common to many recent scope underspecification formalisms (Reyle, 1993; Muskens, 1995; Bos, 1996; Niehren et al., 1997b; Egg et al., 1998) is that they provide the 'semantic material' of which the description of the readings is built, plus structural relations that constrain possible combinations of this material. *Dominance constraints*, a tree logic which can express dominance of nodes in a tree, are a very natural way for doing this. Muskens (1995) was the first to do this explicitly, but Reyle (1993) and Bos (1996) can be subsumed under this kind of approach as well.

In this article, we show how to integrate a treatment of other linguistic phenomena – in particular, anaphora and VP ellipsis – with a scope underspecification formalism based on dominance constraints. The interaction of these three phenomena has been considered before in Crouch (1995) and Shieber et al. (1996). In our approach, it is modelled by a conservative extension of the language of dominance constraints by additional atomic formulas, viz., *anaphoric binding constraints* and *parallelism constraints*. While the former are based on an idea by Kehler (1993), the latter are formally equivalent to *context unification* as used in Niehren et al. (1997b), which was shown in Niehren and Koller (1998). In addition, λ-*binding constraints* are introduced in order to solve a capturing problem that appears in all dominance-based underspecification formalisms.

The combination of all of these constraints, called CLLS (Constraint Language for Lambda Structures), correctly describes many challenging cases from the literature that exhibit interaction of scope, anaphora, and ellipsis. In addition, its being an extension of dominance constraints makes for some appealing differences to existing formalisms with comparable coverage: Unlike Shieber et al. (1996), the CLLS analysis is completely order-independent; and in contrast to the QLF-based analysis in Crouch (1995), it achieves a clean separation between underspecified descriptions and the described readings.

The paper is structured as follows. In Section 2, we introduce the phenomena under consideration and present the intuitions behind our linguistic analysis without going into detail. Section 3 presents the formal definitions of lambda structures and CLLS and fills in the formal details that were omitted in Section 2. In Section 4, we extend the analysis to more involved cases from the literature. Section 5 defines a syntax/semantics interface that derives CLLS constraints on the basis of a simple phrase structure syntax. In Section 6, we give a brief overview over results on formal and computational aspects of CLLS. Finally, Section 7 concludes and outlines directions for future work.

2. Elements of CLLS

This section introduces the three main phenomena considered in the paper – scope, ellipsis, and anaphora – and the major concepts used in their CLLS analyses. After discussing λ-structures and their correspondence to λ-terms, we go through the phenomena and present the intuitions underlying our analysis. The section does not even attempt to be formally precise; the formal details are put off until section 3.

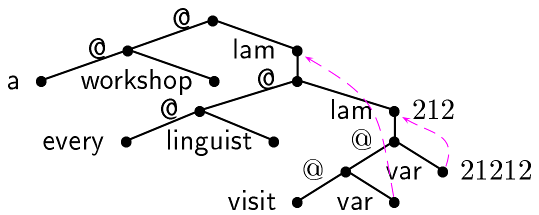
2.1. Lambda structures and lambda terms

The idea behind λ-*structures* is to represent a λ-term by a tree-like graph; a node in a λ-structure corresponds to an occurrence of a subterm in the λ-term it represents. Consider e.g. the λ-term in (1), which happens to represent one of the readings of (3) below:

(1)
```
(a workshop)(λx
    (every linguist)(λy
        (attend x) y))
```

We draw the corresponding λ-structure as the graph in (2). An occurrence of a λ-abstraction corresponds to a node labelled by lam, an occurrence of an application, to a node labelled by @, and an occurrence of a bound variable, to a node labelled by var. Variable binding is represented in a lambda structure by dashed arrows from the variables to their binders. For instance, if we identify each node by its address, i.e. the word of natural numbers that indicates the path from the root to the node, the unique occurrence of λx in (1) corresponds to the node with the address 212 in (2); it binds the variable represented by the node with the address 21212.

(2)



Note that variable names in a λ-term (such as x and y in (1)) are not reflected by the node labels of the corresponding λ-structure. Consequently, a λ-structure represents a closed λ-term uniquely modulo consistent renaming of bound variables (α-equality).

Via the detour through λ-structures, we can use tree descriptions in semantic underspecification: We simply talk about λ-structures in the same way one usually talks about trees. The explicit binding relation is important in that context because variable names may not be sufficient to ensure proper binding in a radically underspecified framework; we will come back to this in a minute.

2.2. Scope

The first class of phenomena we are concerned with is *scope ambiguity*, as in the following sentence.

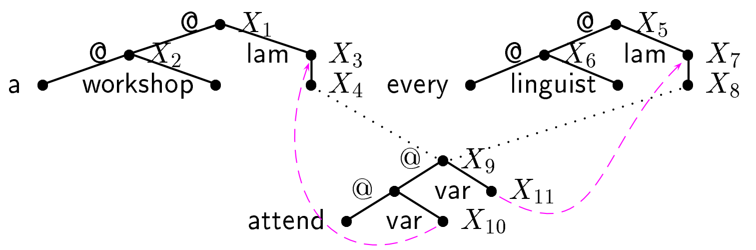
(3) Every linguist attends a workshop.

This sentence has two readings, which differ in a permutation of quantifiers only: Either the sentence means that there is one specific workshop which every linguist attends, or it means that every linguist attends a workshop, but not necessarily the same as the others. The first reading is represented by the λ-term in (1), and the second by (4):

(4)
(every linguist)(λy
(a workshop)(λx
(attend x) y))

Closer inspection reveals that (1) and (4), and hence their corresponding lambda structures, are closely related. The lambda structures are composed from the same tree-like subgraphs, corresponding to the term fragments (every linguist)(λx(.)), (a workshop)(λy(.)), and ((attend x) y). But these fragments are composed in different order. We can describe both lambda structures at once by specifying the fragments and their relationships. Such a description is given in (5).

(5)



Intuitively, description (5) is satisfied by all λ-structures into which the graph in (5) can be embedded in such a way that no labelled nodes of the graph overlap. Dotted edges in (5) signify *dominance*: Of the two nodes they connect, the upper one must be above the lower one in the lambda structure. The dashed arrows, for λ-binding, act like elastic bands, which can be stretched indefinitely without breaking.

The description leaves the relative ordering between the two quantifier fragments unspecified. But since both fragments dominate the nuclear scope and, just like trees, λ-structures cannot branch in the bottom-up direction, one of the two quantifier fragments must dominate the other. This situation is very common in scope underspecification. Note that variable binders must always dominate their bound variables; so the λ-binding relations, $\lambda(X_{10}) = X_3$ and $\lambda(X_{11}) = X_7$, logically entail the dominance relations, $X_4 \triangleleft^* X_9$ and $X_8 \triangleleft^* X_9$. Inserting these dominance edges in (5) only enhances readability.

In Section 3 we will give pictures like (5) a formal meaning as *constraints over λ-structures*, i.e. logical formulas that are interpreted over the class of λ-structures. Then nodes of the graph correspond to variables denoting nodes of a λ-structure, whereas labels, edges, and arrows correspond to various types of atomic constraints. In addition, such a graphical description imposes implicit inequality constraints between labelled nodes, which excludes proper overlap of fragments.

We have said that a description like (5) is satisfied by all lambda structures into which they can be embedded. In particular, the lambda structure may contain additional fragments that are not mentioned in the description. This is a desired feature because we want to compute with limited information that can in principle be augmented at any later point; the fact that the information is complete must be expressed independently. As long as one is only concerned with scope, one can impose a general *constructiveness* restriction: each node in a lambda structure that we describe must be denoted by a variable in the description. But constructiveness would be too strict for our treatment of ellipses below and also for an underspecified account of reinterpretation (Koller et al., 1999; Egg, 1999).

2.3. Ellipsis

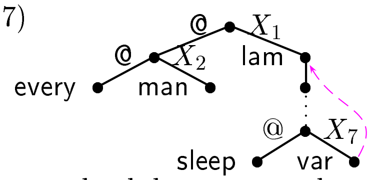
The second phenomenon we treat is *VP ellipsis*. VP ellipsis interacts with scope ambiguities, thus the treatment of both phenomena must be integrated. In the framework presented in this article, this integration simply amounts to adding further constraints on lambda structures.

For ellipses, we introduce *parallelism constraints*, which permit to express structural equality between fragments of λ-structures. As an example, consider the elliptical sentence (6):

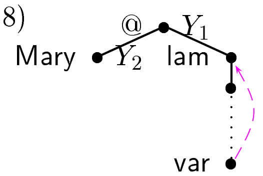
(6) Every man sleeps, and so does Mary.

The meaning of the target sentence *so does Mary* is exactly like the meaning of the source sentence *every man sleeps*, except that the source parallel element *every man* is replaced by the target parallel element *Mary*. This can be modelled in two steps. First, we describe the λ-structure for the source sentence, as in (7), and the λ-structure for the target parallel element, as in (8).

(7)

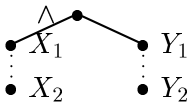


(8)



In any lambda structure described by these constraints, the semantics for the source sentence will be below X_1, and the semantics for the target sentence will be below Y_1. The λ-structure for the entire sentence (6) will be the conjunction of source and target sentence, like this:

(9)



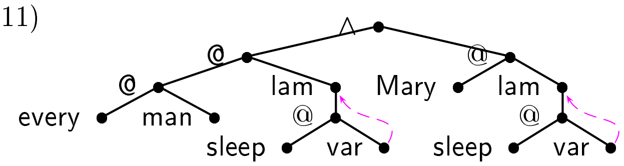
The second step is to impose the *parallelism constraint* (10), which says that the lambda structures for the source and target sentence (i.e.

below X_0 and Y_0) are equal, with the exception of the sub-lambda structures representing the parallel elements (below X_2 and Y_2):

(10) $X_1/X_2{\sim}Y_1/Y_2$

Indeed, the intended semantic representation of (6), shown in (11), simultaneously solves all constraints in (7), (8), (9), and (10).

(11)



This shows that ellipsis resolution can be reduced to solving constraints over λ-structures.

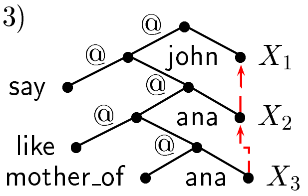
Since proper names and quantifier NPs can be parallel elements in an ellipsis (as in (6)), our analysis requires type-raising of proper names; the constant Mary in (8) corresponds to a λ-term of type $\langle\langle e,t\rangle,t\rangle$. Where there is no ellipsis, proper names are freely analyzed as constants of type e, as this simplifies the presentation. To distinguish the two analyses of proper names, we will capitalize the first letter of label used in the former; that is, the label for the type-raised analysis of proper nouns is Mary and the one for the non-type-raised analysis, mary.

2.4. Anaphora

Anaphora are the final class of phenomena we consider. In order to represent anaphora, we extend our notion of λ-structures. Anaphora are represented as nodes labelled by ana which are linked to their antecedents by *anaphoric links*. We draw anaphoric links as dashed arrows using straight lines. For instance, a representation of (12) is shown in (13). For simplicity, we use a constant mother_of here; we will analyze the possessive differently later.

(12) John^i said he$_i^j$ liked his$_j$ mother.

(13)



In the example, the two anaphoric links form a *link chain*. Link chains go back to Kehler (1993, 1995); they have been worked out in this setting by Xu (1998). This representation of anaphoric reference can cope with a series of problematic cases from the literature, such as strict/sloppy ambiguities, the many-pronouns puzzle (of which (12) is the source clause), and cascaded ellipses.

2.5. The Capturing Problem

A major difference between CLLS and other underspecification formalisms is that variable binding is modelled by explicit λ-binding arrows and not by variable names. This looks like an unnecessary deviation from standard procedure, but it turns out that variable names are inconvenient in the context of underspecification and become completely unusable once we add parallelism.

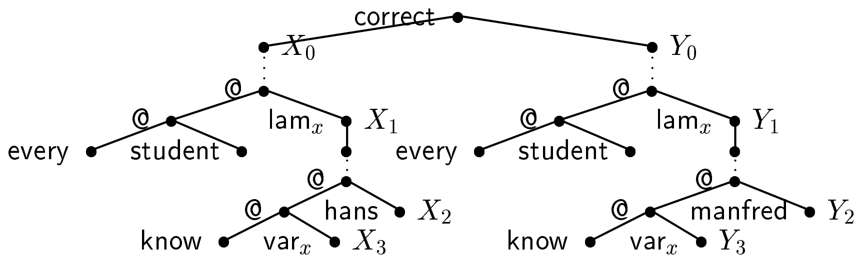
Let us assume that variable binding is modelled by using variable names; for instance, that we use new labels lam_x and var_x for each object-level variable x. Now imagine that a binder lam_x appears twice in a description, and that there is a scope ambiguity between the two occurrences. In this case, it would be unclear which the intended binder of a node with label var_x is; in each solution, the lower binder wins.

The obvious way to avoid this problem is to name all variables apart when generating a description. But this restriction cannot be maintained once parallelism comes into play. This is best illustrated by corrections, as in the following example.

(14) Hans knows every student. No, *Manfred* knows every student.

(15) is an underspecified description of the meaning of (14):

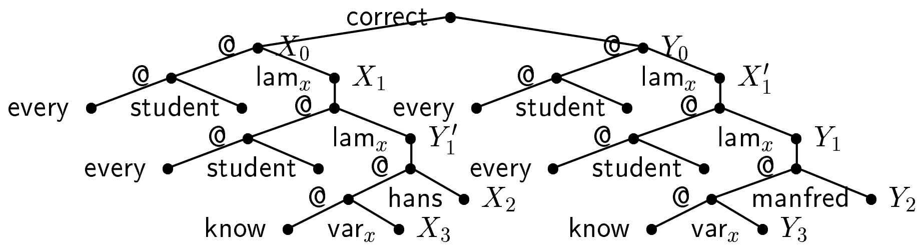
(15) $X_0/X_2 \sim Y_0/Y_2$



In the analysis, we used the same variable name x in both binders. This is necessary since in solving the parallelism constraint, we must make X_1 correspond to Y_1. If these two nodes carried different variable names, we would have to soften the meaning of parallelism constraints in that the labels of corresponding nodes need not be identical, but only unifiable in some sense; but this raises very difficult problems of bookkeeping which variables have been identified with which. The only feasible option is using the same variable name twice.

There are other problems, too. We have seen in Section 2.3 that we cannot even maintain the constructiveness restriction on satisfying lambda structures; parallelism may force us to copy material. But if we do not require constructiveness, we can get (16) as a solution for (15):

(16)



More precisely, the problem is that parallelism (not only for corrections) may force us to copy binders, and if we cannot always rename variables apart, we must enforce that copying variables does not break the intended binding relations (as the one between X_1 and X_3 in (16)).

The problems with variable binding are real problems in the presence of parallelism only; they can be avoided otherwise by naming variables apart and requiring constructiveness, both of which is still slightly unpleasant. Modelling variable binding with an explicit binding relation between nodes offers a clean solution to all of these capturing problems: No matter how much and what kind of material is added between a variable and its binder, the binding relation can never be broken.

3. Syntax and Semantics of CLLS

Section 2 presented the general intuitions behind lambda structures. We showed how to describe λ-structures and how to apply their descriptions in an underspecified analysis of scope, ellipsis, and anaphora. Now we give a rigorous definition of the constraint language for lambda structure (CLLS), which we will use throughout the paper. We will start with the semantic level by first presenting tree structures, then move over to lambda structures as an extension of tree structures. Once the semantics is fixed, a definition of a syntax to talk about it is simple.

3.1. Tree Structures

A tree corresponds to a unique tree structure in the same way that a lambda structure relates to a unique λ-term modulo α-equivalence. Lambda structures are extended tree structures because λ-terms are extended trees (trees plus variable binding).

Let Σ be a set of *function symbols*, f, g, a, b. Each function f symbol comes with a fixed *arity*. We write f^k for a function symbol with arity $k \geq 0$. A (finite constructor) tree is simply a ground term (without variables) built from a set of function symbols. For instance, $f(g(a, a))$ is a tree built from function symbols f^1, g^2, a^0. Taking $\mathbb{N}$ to be the set of natural numbers starting at 1, we can define a *path* as a word over $\mathbb{N}$. We identify each node of a tree with the path from the root to this node; the empty word, ϵ, identifies the root. Paths are named with lowercase Greek letters, such as $\pi, \pi_1, \pi_2, \mu, \mu_1$. Concatenation of words π and μ is written as $\pi\mu$, and a word π is a *prefix* of μ iff there is a word π_1 such that $\mu = \pi\pi_1$.

A tree structure is a first-order model structure which characterizes a tree. The domain of a tree structure contains all nodes of the corresponding tree, and its relations specify labelling and childhood. E.g., the domain of the tree structure for $f(g(a, a))$ is $\{\epsilon, 1, 11, 12\}$. The fact that the root ϵ is labelled with f and has a single child 1 is expressed by a relation we write as $\epsilon{:}f(1)$; the children and labels of the remaining nodes are specified by the relationships $1{:}g(11, 12)$, $11{:}a$, and $12{:}a$.

More generally, a *tree domain* Δ is a finite nonempty set of nodes which is prefixed-closed (that is, $\pi\mu \in \Delta$ implies $\pi \in \Delta$) and closed under left siblings (i.e. if $\pi i \in \Delta$ then $\pi j \in \Delta$ for all $1 \leq j < i$), and a tree structure is defined as follows:

DEFINITION 3.1. *A* tree structure *over Σ consists of a tree domain $\Delta \subseteq \mathbb{N}^*$ and a relation $:f \subseteq \Delta^{n+1}$ for each n-ary function symbol $f \in \Sigma$, such that for all nodes $\pi \in \Delta$ and the maximal $n \in \mathbb{N}$ with $\pi n \in \Delta$, there is a unique symbol $f^n \in \Sigma$ such that $\pi{:}f(\pi 1, \ldots, \pi n)$.*

Given nodes $\pi_0, \ldots, \pi_n$, we write $\pi_0{:}f(\pi_1, \ldots, \pi_n)$ for $(\pi_0, \pi_1, \ldots, \pi_n) \in {:}f$. This relation expresses that π_0 is a node of the tree structure with the label f and the children $\pi_1, \ldots, \pi_n$, in this order. Note that two nodes may be labelled by the same constant without being equal. The condition in the definition means that the number of children of a node is specified by the arity of its label.

3.2. Lambda Structures

A λ-structure extends a tree structure with two additional relations for λ-binding and anaphoric links. For λ-structures, we assume:

$$\{\mathsf{var}^0, \mathsf{ana}^0, \mathsf{lam}^1, @^2\} \subseteq \Sigma$$

DEFINITION 3.2. *A* λ-structure *over Σ consists of a tree structure over Σ with tree domain Δ and two additional binary relations λ and* ante *on Δ which satisfy the following properties.*

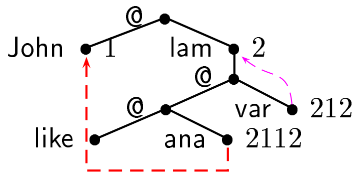
1. *$\lambda \subseteq \Delta \times \Delta$ is a partial function that maps every node $\pi \in \Delta$ with label* var *to a prefix of π which is labelled with* lam.

2. $\mathsf{ante} \subseteq \Delta \times \Delta$ *is a partial function that maps every node with label* ana *to another node.*

The lambda structures we are interested in here have additional properties, which we have not included in the definition because they are not essential for what follows. For instance, there are no cyclic anaphoric link chains, and if a link chain ends at a node with label var, then each node in the chain (all of which are labelled by ana) must be below the binder of that var-labelled node.

We draw lambda structures as tree-like graphs. E.g., the binding and anaphoric linking functions of the lambda structure (17) are specified by $\lambda(212) = 2$ and $\mathsf{ante}(2112) = 1$. Note that we annotate only those nodes with addresses in the picture that we want to refer to.

(17)



Lambda structures and λ-terms can almost be mapped to each other one-to-one. The two exceptions are that lambda structures do not mention variable names and that lambda structures contain anaphoric links. We do not elaborate on the further interpretation of anaphoric links here; roughly, they express that the interpretation of the last element of the link chain determines the interpretation of all other elements of the chain.

3.3. Dominance and Parallelism

Finally, we conservatively extend lambda structures by two additional relations between nodes: *dominance* and *parallelism.* These relations were not included in the original definition as they can be defined from the relations specified there; but they are essential for the application.

Let Λ be a λ-structure and π, μ two of its nodes. We say that π *dominates* μ ($\pi \lhd^* \mu$) if it lies above it, i.e. if π is a prefix of μ. The dominance relation is a partial order on the domain of Λ, i.e. reflexive, transitive, and antisymmetric.

For the definition of parallelism, we first define for any two nodes $\pi \lhd^* \mu$ in Λ the *context of* μ *below* π *in* Λ as the set of nodes of Λ which are dominated by π, but not by μ. Intuitively, the context of μ below π is the set of nodes 'between' μ and π. Taking the labelling relations of Λ into account, a context looks almost like a tree domain starting at π, except that one node (the mother of μ) has one too few children. Based on this intuition, we call π the *root* of the context and μ its *hole.*

Intuitively, the parallelism $\pi_1/\mu_1 \sim \pi_2/\mu_2$ holds for a λ-structure Λ if Λ imposes parallel structures on the contexts between π_1 and μ_1 and between π_2 and μ_2. As it must also specify what it means for binding and anaphoric linking to be parallel, however, the following definition of the relation is rather involved.

DEFINITION 3.3. *Let Λ be a λ-structure over Σ and $\pi_1, \mu_1, \pi_2, \mu_2$ nodes of its domain Δ.* Parallelism $\pi_1/\mu_1 \sim \pi_2/\mu_2$ *holds in Λ iff there are subsets $\Delta_1, \Delta_2 \subseteq \Delta$ and a bijective correspondence function c : $\Delta_1 \cup \{\mu_1\} \to \Delta_2 \cup \{\mu_2\}$ such that* **P1**–**P6** *are satisfied.*

P1 *For $i \in \{1, 2\}$, the set Δ_i is the context μ_i below π_i in Λ:*

$$\pi_i \lhd^* \mu_i$$
$$\Delta_i = \{\pi \in \Delta \mid \pi_i \lhd^* \pi \text{ and not } \mu_i \lhd^* \pi\}$$

P2 *Λ imposes parallel tree relations on Δ_1 and Δ_2: For all $f^n \in \Sigma$,*

$$\forall \pi \in \Delta_1 : \ \pi{:}f(\pi 1, \ldots, \pi n) \Leftrightarrow c(\pi){:}f(c(\pi)1, \ldots, c(\pi)n)$$

In particular, this enforces that $c(\mu_1) = \mu_2$ (that is, the holes correspond to each other) and that for all $\pi \in I\!N^*$, $\pi_1\pi \in \Delta_1 \cup \{\mu_1\} \Rightarrow c(\pi_1\pi) = \pi_2\pi$ (that is, corresponding nodes can be reached over the same path from the roots of their respective contexts).

P3 *Binding within Δ_1 and Δ_2 is parallel:*

$$\forall \pi, \mu \in \Delta_1 : \ \lambda(\mu){=}\pi \Leftrightarrow \lambda(c(\mu)){=}c(\pi)$$

P4 *For $i \in \{1,2\}$, no node below μ_i is bound by a node inside Δ_i:*

$$\forall \pi : \ \mu_i \triangleleft^* \pi \Rightarrow \lambda(\pi) \notin \Delta_i$$

P5 *If Λ binds a variable node within Δ_1 properly above π_1 or the corresponding variable node in Δ_2 properly above π_2, then the two binder nodes are* binding equivalent *('$\triangleleft^+$' denotes proper dominance).*

$$\forall \pi \in \Delta_1 : \ (\lambda(\pi) \triangleleft^+ \pi_1 \vee \lambda(c(\pi)) \triangleleft^+ \pi_2) \Rightarrow \lambda(\pi) \approx \lambda(c(\pi))$$

For the time being, we assume that binding equivalence $\approx$ is simply the node identity. We will modify this definition in Section 4.5.

P6 *If π is an anaphoric node in Δ_1, then the corresponding anaphoric node in Δ_2 must be linked either to π (strict) or, if it exists, to the correspondence of* $\mathsf{ante}(\pi)$ *(sloppy).*

$$\forall \pi \in \Delta_1 : \ \mathsf{ante}(c(\pi)) = \pi \ \vee \ \mathsf{ante}(c(\pi)) = c(\mathsf{ante}(\pi))$$

P1 specifies the parallel contexts and the correspondence function between them. **P2** requires parallelism of tree structure. Conditions **P3**–**P5** are concerned with the interaction of parallelism and variable binding. Condition **P6** deals with the interaction of parallelism and anaphora, i.e. with strict/sloppy ambiguities.

The parallelism relation is almost symmetric; the only exception is that the strict variant of the rule **P6** only allows to link the anaphoric node in Δ_2 to the corresponding node in Δ_1, but not vice versa.

3.4. The Constraint Language CLLS

Now that λ-structures and their relations are defined, it is straightforward to fix a constraint language to go with them. We assume an infinite set of *node variables*, ranged over by X, X_i, Y, etc., and pick relation symbols for all relations defined so far. To keep the presentation simple, we use the same symbols as for the relations themselves. The constraint language over lambda structures (CLLS) is given by the following abstract syntax:

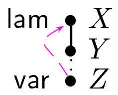
$$\begin{array}{lll} \varphi & ::= & X{:}f(X_1,\ldots,X_n) \qquad\qquad (f^n \in \Sigma) \\ & | & X \triangleleft^* Y \\ & | & X \neq Y \\ & | & \lambda(X)=Y \\ & | & \mathsf{ante}(X)=Y \\ & | & X/X' {\sim} Y/Y' \\ & | & \varphi \wedge \varphi' \end{array}$$

The semantics of CLLS is defined by interpretation of constraints over the class of lambda structures in the usual Tarski style. A constraint φ of CLLS can be evaluated to a truth value if a λ-structure Λ and a variable assignment α into the domain of Λ are given. A solution of φ is a pair (Λ, α) which satisfies φ.

3.5. Constraint Graphs

CLLS constraints as just defined can become unreadable rather quickly. Throughout the paper, we will use *constraint graphs* instead. Constraint graphs are an alternative, graphical syntax for CLLS; a constraint graph corresponding to a constraint φ contains one node for each variable in φ and labels and edges for the atomic constraints that appear in φ. A simple example is given in (18).

(18)



$$\xrightarrow{\text{represents}} \quad \begin{array}{l} X{:}\mathsf{lam}(Y) \wedge Y \triangleleft^* Z \wedge \\ Z{:}\mathsf{var} \wedge \lambda(Z)=X \wedge \\ X \neq Z \end{array}$$

For every dominance constraint $X \triangleleft^* Y$, we draw a dotted line from X to Y. For every labelling constraint $X{:}f(X_1, \ldots, X_n)$, we label the graph node for X with f and draw solid lines to the nodes for $X_1, \ldots, X_n$. For every λ-binding constraint $\lambda(X)=Y$, we draw a curved dashed arrow from X to Y; and for every anaphoric binding constraint $\mathsf{ante}(X)=Y$, we draw a dashed arrow from X to Y that only uses straight lines. Parallelism constraints cannot be represented easily in constraint graphs; we will simply write them next to the graph.

Finally, a constraint graph represents inequality constraints $X_1 \neq X_2$ for each two labelled nodes in the graph. This assumption nicely prevents 'overlaps' between the 'fragments' of the constraint graph.

Now we can interpret the intuitive diagrams from Section 2 as CLLS constraint graphs. This gives them a formal meaning as CLLS constraints, which can be satisfied by lambda structures. These lambda structures, in turn, represent λ-terms which stand for the actual meaning(s) of the sentence.

4. Interaction of Quantifiers, Anaphora, and Ellipsis

Now that we know how to deal with simple examples, we turn to some well-known, more complex challenges in the area of quantifiers, anaphora, and ellipses, in particular to involving their interactions. We look into quantifier parallelism, strict/sloppy ambiguities, nested ellipses, a sentence involving interaction of all three, and antecedent-contained deletion (ACD). There are other interesting things that the analysis can do; e.g., to deal with gapping, parallelism could be extended to allow *two* exception positions rather than one: Niehren et al. (1997a) showed that this can be done in a different setting.

Throughout, we assume a fixed signature:

$$\Sigma = \{@^2, \mathsf{lam}^1, \mathsf{var}^0, \mathsf{ana}^0, \mathsf{before}^2, \mathsf{mary}^0, \mathsf{read}^0, \ldots\}$$

We follow the convention that proper nouns are always analyzed as constants of type e, except as contrasting elements in ellipses where the other contrasting element is a quantifier. We indicate the first analysis by labelling the corresponding nodes in lowercase letters (john), the second one, by using labels with a capitalized first letter (John). Furthermore, we draw dominance edges wherever dominance is entailed by λ-binding constraints. This is logically redundant, but may improve readability.

4.1. Quantifier Parallelism

The first phenomenon we consider is *quantifier parallelism*, an interaction of scope and ellipsis. By way of example, we continue the sentence (3), repeated here as (19), with the elliptical sentence (20).

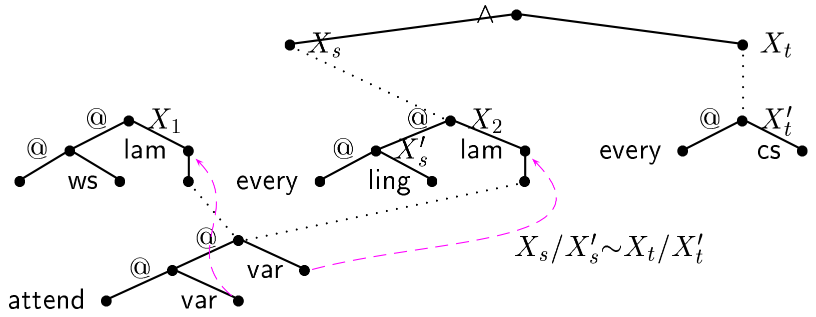
(19) Every linguist attends a workshop.

(20) Every computer scientist does, too.

This pair of sentences, modelled after an example from Hirschbühler (1982), has three readings: Either there is a single workshop which both all linguists and all computer scientist attend; or there is one workshop for all linguists and one for all computer scientists (but not necessarily the same one); or every linguist and computer scientist can attend a different workshop. This is surprising; if the target sentence means 'Every computer scientist attends a workshop', each sentence is two-way ambiguous, for a total of at least four readings.

In CLLS, parallelism enforces that the quantifier scopes in both sentences must be analogous; the ambiguities cannot be resolved independently. This is reflected by the following constraint (abbreviating workshop to ws, linguist to ling, and computer_scientist to cs):

(21)



X_s is the root of the semantics of the source sentence; X_t is the root of the target sentence. The contrasting elements, *every linguist* and *every computer scientist*, are below the variables X_s' and X_t', respectively. There is a dominance constraint between X_s and X_2, but not between X_s and X_1, since, like most quantifiers, universal quantifiers must obey a 'scope island constraint' that prevents them from escaping from a sentence. This restriction does not hold for indefinites, such as *a workshop*.

There are three places where the fragment below X_1 can go in a lambda structure satisfying the constraint: below the fragment starting at X_2, between X_s and X_2, and above the conjunction. In each case, the parallelism constraint enforces that whatever is below X_s is also below X_t, replacing what is below X_s' with what is below X_t'. This produces exactly the three readings of the sentence. Note that in the third reading, the indefinite takes scope over the entire conjunction; it is not part of the parallel contexts.

A particular consequence of the parallelism constraint enforcing structural parallelism of the sub-lambda structures below X_s and X_t is that the relative scopes of the quantifiers must be identical in both the source and the target sentence (if the indefinite is below X_s, of course).

4.2. Strict/sloppy Ambiguities

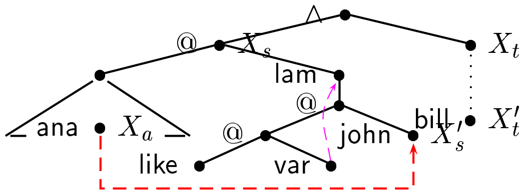
Strict/sloppy ambiguities are ambiguities of anaphoric reference that occur in the context of VP ellipses. They are an important benchmark for analyses of ellipsis. The prototypical example looks as follows:

(22) John likes his mother, and Bill does too.

This sentence has two readings: Either Bill likes John's mother (this is the *strict* reading), or Bill likes his own mother (the *sloppy* reading).

We describe the meaning of (22) as in (23), using parallelism and anaphoric linking constraints. We have hidden the sub-lambda structure for *his mother* and only drawn an empty triangle, as the only node we are interested in is the anaphor.

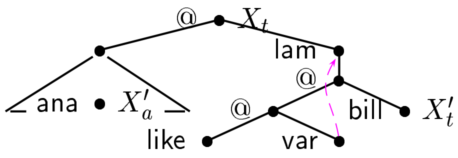
(23)



$X_s/X_s'{\sim}X_t/X_t'$

Except for the fact that the source clause contains an anaphor, this constraint is totally analogous to the simple ellipses we considered above. The parallelism constraint first enforces that the tree part of the lambda structure below X_t is the same as the tree part of the one below X_s, except for the contrasting elements, as follows:

(24)



(24) is not a complete lambda structure because the anaphor at X_a' does not have an antecedent. According to P6 of Definition 3.3, there are two legal antecedents that this anaphor can be linked to. It can either be linked to the anaphor in the source context, i.e. to the node denoted by X_a. In this case, the link chain starting at X_a' ends at X_s, which is labelled with john; so this is the strict reading. Or it can be linked to the node corresponding to the antecedent of X_a. As X_a is linked to the hole of the source context, X_a' must be linked to the hole of the target context; this is X_t', which produces the sloppy reading.

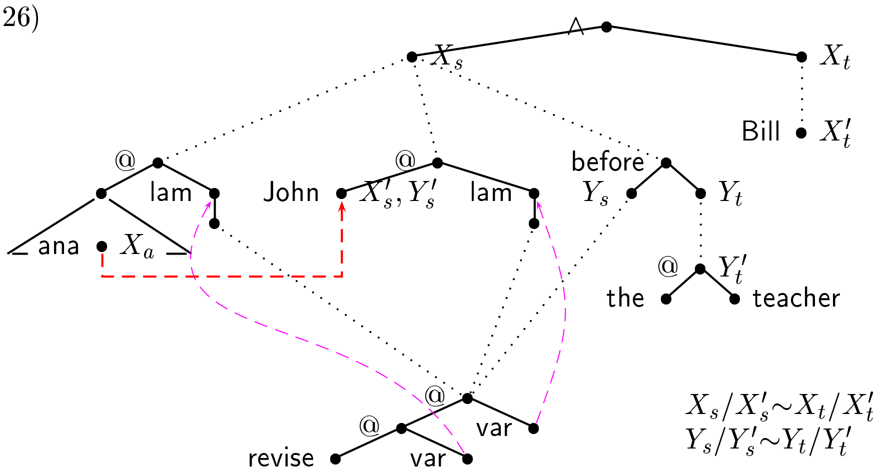
4.3. Nested Ellipses

This analysis of strict/sloppy ambiguity carries over to more sophisticated cases like the Gawron and Peters (1990) example

(25) John revised his paper before the teacher did, and so did Bill.

This sentence comprises nested ellipsis: the source clause *John revised his paper before the teacher did* of the ellipsis is elliptical itself; ellipsis resolution is made even more interesting by the presence of an anaphor. We follow Dalrymple et al. (1991) in assuming five readings (listed in (27)) for (25). These five readings are described by the constraint in (26).

(26)



For readability, we again compressed the subconstraint describing *his paper* into an empty triangle; the only visible (and interesting) node is X_a, the anaphor. The constraint contains *two* parallelism constraints, one for each ellipsis. Note that the source sentence seems to contain a scope ambiguity between the three fragments corresponding to *his paper*, *John*, and *before*. However, the constraint entails that *before* takes scope over *John* because $Y_s/Y_s' \sim Y_t/Y_t'$ entails $Y_s \triangleleft^* Y_s'$. The remaining scope ambiguity between *his paper* and the other two fragments is spurious; we chose to leave it in the constraint to be closer to what the syntax/semantics interface (Section 5) would produce.

To understand how the five (constructive) solutions come about, let us try to 'solve' the constraint by mimicking first resolution of the inner and then of the outer ellipsis. An actual solution algorithm might behave differently; in particular, it could interleave resolution steps for the inner and outer parallelism constraint and other constraints such as the dominance constraints for a scope ambiguity, as the formalism is totally declarative. We call the correspondence function on the parallel contexts that is induced by the inner parallelism constraint c_i, and that of the outer one, c_o.

Resolution of the parallelism constraint for the inner ellipsis will make the description of the sentence *the teacher revised his paper* more explicit. Let us say that the node denoted by the variable for the original anaphor is π_a; then its copy via this parallelism constraint is $c_i(\pi_a)$. There are two options where $c_i(\pi_a)$ can be linked: either to π_a (strict – this produces the first three readings in (27)) or to μ_t', the denotation of Y_t' (sloppy – this produces the other two readings).

If we now resolve the parallelism constraint for the outer ellipsis, we produce copies of π_a and of $c_i(\pi_a)$: $c_o(\pi_a)$ and $c_o(c_i(\pi_a))$. The node $c_o(\pi_a)$ can be linked either to π_a (strict) or to π_t', the denotation of X_t'

(sloppy). Likewise, $c_o(c_i(\pi_a))$ can be linked either to $c_i(\pi_a)$ (strict), or it can be linked to the respective value of $c_o(\mathsf{ante}(c_i(\pi_a)))$ (sloppy).

So there are five possible combinations of linking relations:

(27)
1. $\mathsf{ante}(c_i(\pi_a))=\pi_a$, $\mathsf{ante}(c_o(\pi_a))=\pi_a$, $\mathsf{ante}(c_o(c_i(\pi_a)))=c_i(\pi_a)$ and $\mathsf{ante}(c_i(\pi_a))=\pi_a$, $\mathsf{ante}(c_o(\pi_a))=\pi_a$, $\mathsf{ante}(c_o(c_i(\pi_a)))=c_o(\pi_a)$:
   John revised *John's* paper before the teacher revised *John's* paper,
   Bill revised *John's* paper before the teacher revised *John's* paper
2. $\mathsf{ante}(c_i(\pi_a))=\pi_a$, $\mathsf{ante}(c_o(\pi_a))=\pi_t'$, $\mathsf{ante}(c_o(c_i(\pi_a)))=c_i(\pi_a)$:
   John revised *John's* paper before the teacher revised *John's* paper,
   Bill revised *Bill's* paper before the teacher revised *John's* paper
3. $\mathsf{ante}(c_i(\pi_a))=\pi_a$, $\mathsf{ante}(c_o(\pi_a))=\pi_t'$, $\mathsf{ante}(c_o(c_i(\pi_a)))=c_o(\pi_a)$:
   John revised *John's* paper before the teacher revised *John's* paper,
   Bill revised *Bill's* paper before the teacher revised *Bill's* paper
4. $\mathsf{ante}(c_i(\pi_a))=\mu_t'$, $\mathsf{ante}(c_o(\pi_a))=\pi_a$, $\mathsf{ante}(c_o(c_i(\pi_a)))=c_i(\pi_a)$ and $\mathsf{ante}(c_i(\pi_a))=\mu_t'$, $\mathsf{ante}(c_o(\pi_a))=\pi_a$, $\mathsf{ante}(c_o(c_i(\pi_a)))=c_o(\mu_t')$:
   John revised *John's* paper before the teacher revised *the teacher's* paper,
   Bill revised *John's* paper before the teacher revised *the teacher's* paper
5. $\mathsf{ante}(c_i(\pi_a))=\mu_t'$, $\mathsf{ante}(c_o(\pi_a))=\pi_t'$, $\mathsf{ante}(c_o(c_i(\pi_a)))=c_i(\pi_a)$ and $\mathsf{ante}(c_i(\pi_a))=\mu_t'$, $\mathsf{ante}(c_o(\pi_a))=\pi_t'$, $\mathsf{ante}(c_o(c_i(\pi_a)))=c_o(\mu_t')$:
   John revised *John's* paper before the teacher revised *the teacher's* paper,
   Bill revised *Bill's* paper before the teacher revised *the teacher's* paper

A reading where π_a, $c_i(\pi_a)$, and $c_o(\pi_a)$ are all linked to John and $c_o(c_i(\pi_a))$ is linked to Bill, which is particularly difficult to exclude in other approaches, is impossible here, as $c_o(c_i(\pi_a))$ must be linked either to $c_i(\pi_a)$ or to $c_o(\pi_a)$.

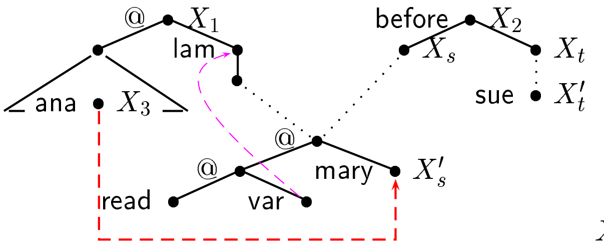
4.4. A Complex Interaction

Now we consider an example involving quantification, anaphora, and ellipsis where all three phenomena interact.

(28) Mary read a book she liked before Sue did.

This sentence, modelled after a Gawron and Peters (1990) example, has three readings. In the first, the indefinite NP *a book she liked* takes wide scope over both clauses (a particular book liked by Mary is read by both Mary and Sue). The second and third arise from the strict/sloppy ambiguity that occurs if the operator *before* outscopes the indefinite: either both read a book that Mary liked, or each read a book she liked herself. A constraint describing these three readings is shown in (29).

(29)

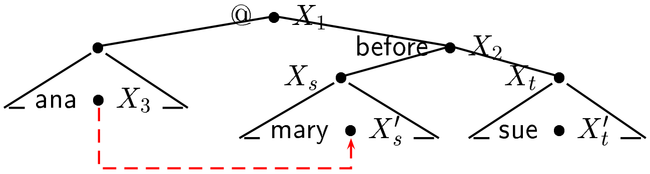


$X_s/X'_s{\sim}X_t/X'_t$

For the sake of readability, we once again simplify the semantics of *a book she liked* to a mere empty triangle with the anaphoric node X_3.

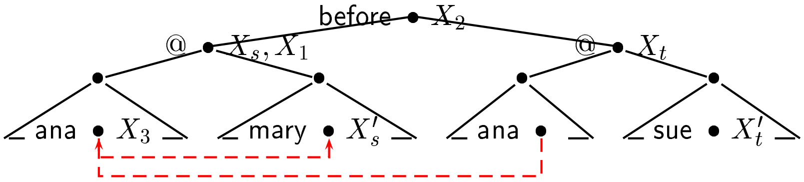
The first reading, with the NP taking wide scope, results when the relative scope of X_1 and X_2 is resolved such that X_1 dominates X_2. In this case, the anaphoric node is not part of the parallel subdomain, which means that there is no strict/sloppy ambiguity. A schematic representation of this solution (in which we have compressed even more sub-lambda structures into empty triangles) is shown in (30).

(30)

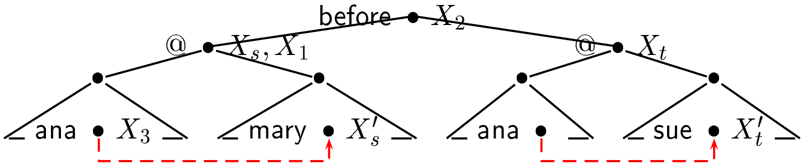


The other possible scoping is for X_2 to dominate X_1. The two solutions this gives rise to are displayed in (31) and (32). Now the interpretation of the indefinite, below X_1, is dominated by X_s and is copied by resolving the parallelism constraint. In particular, the anaphoric node, denoted by X_3, is copied; the strict/sloppy ambiguity of its copy is dealt with in the now familiar way.

(31)



(32)



4.5. Antecedent-Contained Ellipsis

We conclude this section with a notoriously difficult case of ellipsis, *antecedent-contained deletion* (ACD), as in (33).

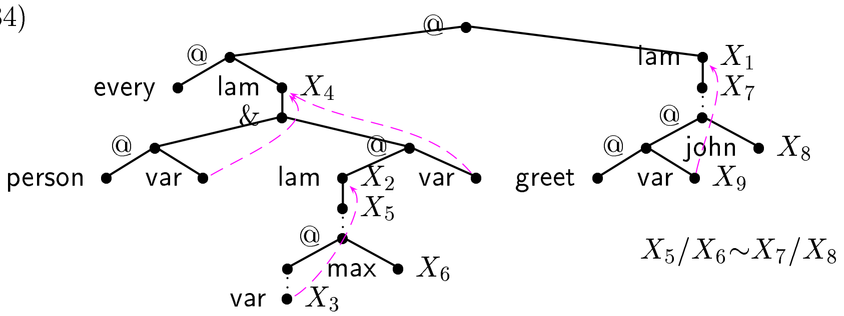
(33) John greeted every person that Max did.

The problem here is is that the ellipsis is contained in the VP that it refers to (its 'antecedent'). A naive approach to ellipsis that simply copies phonetic material from the source to the target sentence runs into an infinite loop as there is always an elliptic sub-sentence left.

Generative syntax solves this problem by raising the universal quantifier out of the sentence in LF (Sag, 1980; Fiengo and May, 1994); this has the effect that the ellipsis is no longer contained in the copied material. Dalrymple et al. (1991) and Crouch (1995) avoid a similar kind of recursion by an 'occurs check' of higher-order unification.

In CLLS, the meaning of (33) can be described as follows.

(34)



Like in the other analyses, the universal quantifier is not contained in the source context of the parallelism constraint.[1] As already discussed in Section 4.3, the constraint also entails that the universal quantifier must take scope over *John*.

There is one subtle problem, however. Condition P5 in Definition 3.3 requires that if π is labelled with var, the binders of π and $c(\pi)$ must be *binding equivalent*, which we have taken to mean 'equal' so far. However, ACD forces one to soften this restriction: The var-node X_3 is bound by X_2, but X_2 cannot bind the copy of X_3 because it does not dominate it. Intuitively, it should be permitted for the copy (X_9) to be bound by X_1 because if it is, a sequence of β-reductions will map

[1] If the universal quantifier *was* contained in the context, the only solutions of the constraint would be infinite lambda structures. This means that it would be unsatisfiable according to Section 3, as we are interpreting CLLS over finite lambda structures. This is reminiscent of the occurs check, as the function of an occurs check is to avoid infinite solutions to unification problems.

X_3 and its copy to the same variable. The separation of the two binders is merely an artifact of the type-raised analysis of *every*.

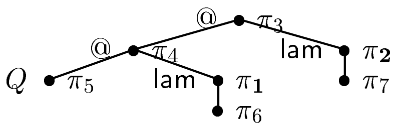
However, checking for $\beta\eta$-equivalence while processing the constraint is unfeasible: It is computationally expensive, and it is unclear if it can be done at all before all readings have been enumerated. Instead, we use the following revised definition of binding equivalence, which is easier to check and works for the examples:

DEFINITION 4.1. *Let* Λ *be a lambda structure.* Binding equivalence in Λ *is the smallest equivalence relation* $\approx$ *on nodes in the domain of* Λ *such that for all* $\pi_1, \ldots, \pi_7$,

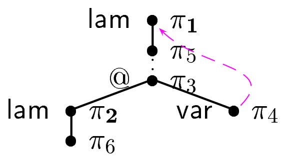
1. *if* $\pi_1{:}\mathsf{lam}(\pi_6)$, $\pi_2{:}\mathsf{lam}(\pi_7)$, $\pi_3{:}@(\pi_4, \pi_2)$, $\pi_4{:}@(\pi_5, \pi_1)$, *and* $\pi_5{:}Q$, *where* Q *is a quantifier label, i.e. a member of* $\{\mathsf{every}, \mathsf{a}, \ldots\}$, *then* $\pi_1 \approx \pi_2$;
2. *if* $\pi_3{:}@(\pi_2, \pi_4)$, $\pi_1{:}\mathsf{lam}(\pi_5)$, $\pi_2{:}\mathsf{lam}(\pi_6)$, $\pi_4{:}\mathsf{var}$, *and* $\lambda(\pi_4) = \pi_1$, *then* $\pi_1 \approx \pi_2$.

The first branch models β-equivalence; the precondition is shown in (35a). The second branch models η-equivalence; its precondition is shown in (35b). If we modify our definition of the parallelism relation in this way, the constraint in (34) has exactly the correct solution.

(35) (a)



(b)



Note, by way of conclusion, that our analysis can also account for the difference between (36) and (37) (Sag, 1980), the first of which lacks one of the two readings of the second one (where the matrix verb *want* outscopes the universal quantifier).

(36) John wants Bill to read everything that Max does.

(37) John wants Bill to read everything Max wants him to read.

This blocking is achieved in the CLLS analysis because, as discussed above, *John* must be in the scope of the universal quantifier (it is the contrasting element). As every NP argument outscopes the verb of which it is a syntactic argument, *John* must outscope *want*; so the relative scope of the universal quantifier and *want* is fixed. If the elliptic material of (36) is spelt out like in (37), there is no parallelism constraint to enforce wide scope of the universal quantifier over *John*.

5. The Syntax-Semantics Interface

This section defines a grammar with a syntax/semantics interface which allows the derivation of the CLLS constraints for the example sentences discussed in this paper. As we have used a mixture of type-raised and first-order representations of proper names in the examples to improve readability, we will define an interface for each version.

We will first define the grammar and then the rules for semantic construction, starting with the type-raised version and then showing what needs to be changed to get the first-order representations. Finally, we will go through an example to show how the interface works.

5.1. Grammar

The grammar is defined by the phrase structure rules in Fig. 1.

(a1) $\mathtt{S} \to \mathtt{NP\ VP}$	**(a8)** $\mathtt{NP} \to \mathtt{Det}\ \overline{\mathtt{N}}$
(a2) $\mathtt{VP} \to \mathtt{IV}$	**(a9)** $\overline{\mathtt{N}} \to \mathtt{N}$
(a3) $\mathtt{VP} \to \mathtt{TV\ NP}$	**(a10)** $\overline{\mathtt{N}} \to \mathtt{N\ RCl}$
(a4) $\mathtt{VP} \to \mathtt{AV\ NP\ VP}$	**(a11)** $\mathtt{RCl} \to \mathtt{RP\ S}$
(a5) $\mathtt{VP} \to \mathtt{SV\ S}$	**(a12)** $\mathtt{S} \to \mathtt{S\ Coord\ S}$
(a6) $\mathtt{VP} \to \mathtt{VP\ Conj\ S}$	**(a13)** $\alpha \ \to\ W$ if $(W, \alpha) \in Lex$

Figure 1. The grammar

Most labels for syntactic categories are self-explanatory, perhaps except for the following: `Conj` refers to the category of conjunctions like *before*, `Coord`, to the one of sentence coordinators like *and*, and `RP` and `RCl`, to the categories of relative pronoun and relative clause, respectively. Beyond transitive and intransitive verbs, we also allow 'AcI verbs' `AV` (such as *expect*) and sentence complement verbs `SV` (such as *say*). `Conj` and `Coord` are distinguished because they percolate scope domains in a different fashion.

The lexicon is defined by a relation Lex, which relates words W and lexical categories $\alpha \in \{\mathtt{Det}, \mathtt{N}, \mathtt{IV}, \mathtt{TV}, \mathtt{SV}, \mathtt{RP}, \ldots\}$. Terminal productions **(a13)** expand lexical categories to words of this category.

We make two types of simplifying assumptions about the grammar. First, we disregard all aspects of syntactic analysis that are inconvenient for a pure phrase structure grammar, such as agreement. Any serious NLP system would employ some type of unification grammar, in which these aspects could be taken care of easily. As a matter of fact, we have implemented an HPSG grammar that does this and produces essentially the same constraints as the one presented here.

Second, we assume that the syntax provides coindexation that relates anaphoric pronouns to their antecedents and relative pronouns to their corresponding traces, and that the parallelism constraints that model VP ellipsis are generated by an independent source. Determining elliptic elements is, to our knowledge, an open problem and therefore beyond the scope of this paper.

5.2. Semantic construction

The overall strategy of our syntax-semantics interface is to factor out as much of the constraint construction as possible into the interface rules. This allows us to keep the entries in the semantic lexicon extremely simple: most of them introduce just one labelling constraint. We generate a constraint for each node in the syntax tree; the constraint for the entire tree will then be simply the conjunction of these subconstraints.

Each node $\nu \in \mathbb{N}^*$ in the syntax tree is associated with two variables, X^s_ν (the *local scope domain* of ν) and X^r_ν (the *root* of the subconstraint for ν). Intuitively, X^r_ν denotes the root of the 'semantic contribution' of the syntactic constituent below ν. X^s_ν is referred to in scope island constraints; for each quantifier NP ν, except for indefinites, a constraint $X^s_\nu \lhd^* X^r_\nu$ keeps the quantifier within its scope domain. The values of X^s_ν are set in the construction rules for embedded sentences; otherwise, daughters inherit the scope domains of their mothers.

Furthermore, we associate with each NP index i that is used for coindexation in the syntax tree a variable X_i. We will use these variables to model anaphoric binding and relative clauses.

The (semantic) variables associated with syntactic nodes are related by the rules in Fig. 2, one for each syntactic rule from Fig. 1. Each construction rule is applied to syntactic nodes that have been expanded by a certain syntax rule. These rules are specified on the left hand sides of the arrows; $[_{\nu:P}\ Q\ R]$ means that node ν in the syntax tree is labelled with P, and its two daughter nodes ν' and ν'' are labelled with Q and R, respectively.

The complete constraint which the interface generates is the conjunction of all of these local constraints, plus the dominances $X^s_\nu \lhd^* X^r_\nu$ for scope islands, plus constraints for the lexical entries. Most of the latter constraints are simple labelling constraints; for instance, the constraint we add for the intransitive verb *sleep* would be $X^r_\nu{:}\mathsf{sleep}$.

There are some exceptions to this general rule. First, the elliptic *does (too)* does not add a labelling constraint; its semantics is

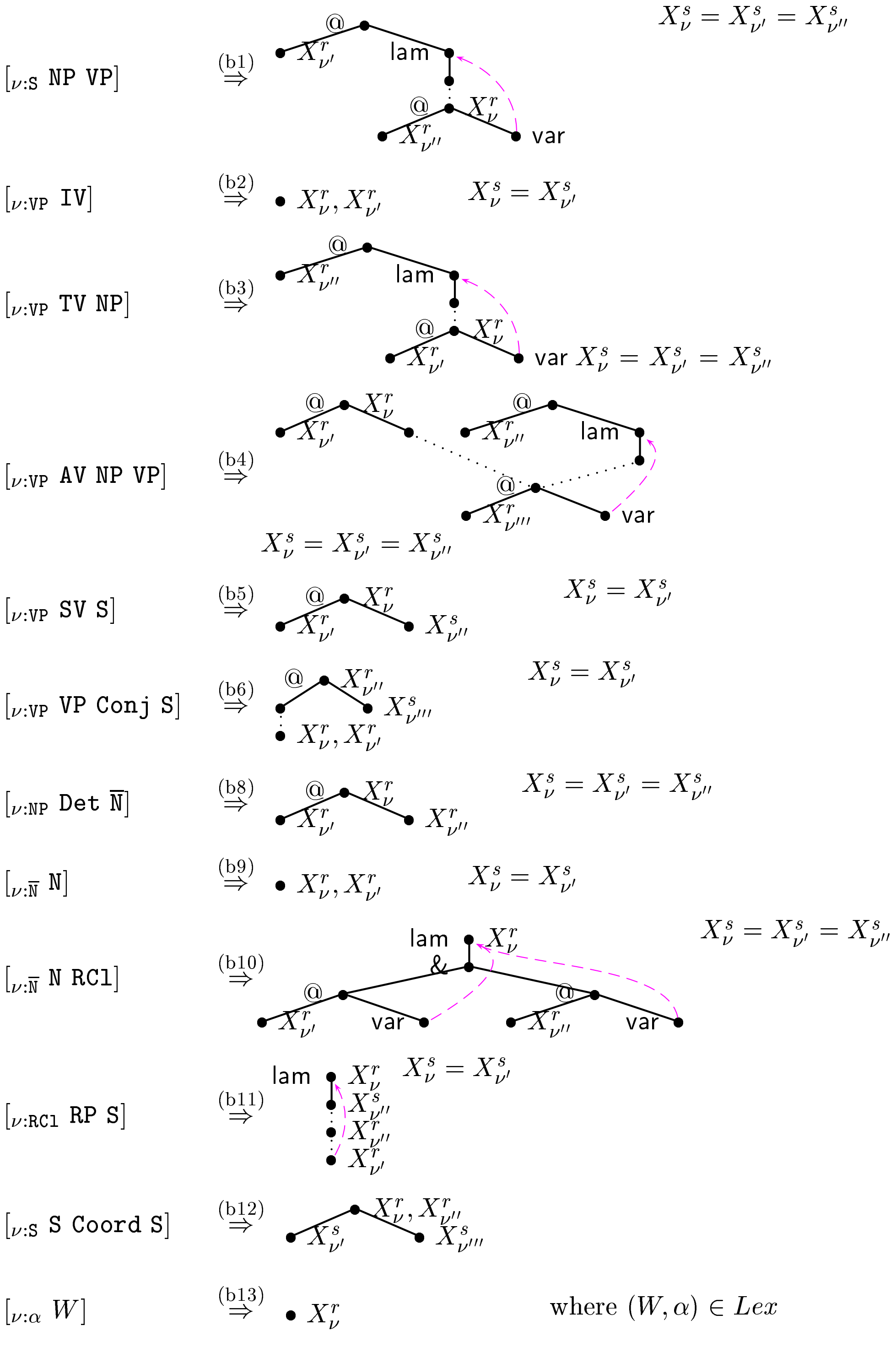
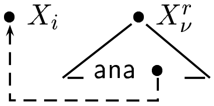


Figure 2. The syntax/semantics interface

determined via a parallelism constraint. Furthermore, whenever coindexation signifies a relation between an anaphor ν' and its antecedent ν, we add the constraint $X^r_\nu{=}X_i$ when we process ν and the constraint $X^r_{\nu'}{:}\mathsf{ana} \wedge \mathsf{ante}(X^r_{\nu'}){=}X_i$ when we process ν'. Similarly, the constraint for a relative pronoun with index i at ν is $X^r_\nu{=}X_i \wedge X_i{:}\mathsf{var}$, and the constraint for the corresponding trace (say, at ν') is $X^r_{\nu'}{=}X_i$. This, together with rule **(b11)**, enforces correct binding of the trace.

Finally, the constraints for possessive pronouns, such as *his*, are as in (38). Intuitively, we want to analyze the NP *his*$_i$ *book* just like the NP *the book of him*$_i$; the diagram in (38) hides most details of this construction for readability, but notice that there is an anaphoric linking constraint from the anaphoric node to the antecedent X_i, as specified by the syntactic coindexation.

(38)



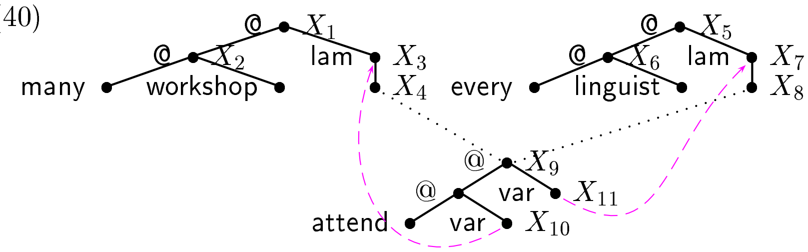
The interface as presented so far will always produce type-raised representations of all NPs. A version of the interface which produces first-order representations of proper names (i.e. as constants of type e) can be obtained by making exactly two changes to the interface. For one, the rule **(b7)** must be replaced by a rule that looks like **(b9)**; second, the lexicon must be changed to label the nodes with the constants of type e (i.e., mary instead of Mary).

5.3. An Example

To see how semantic construction works, we go through the derivation of (40), the semantic representation of (39), a variant of (5).[2] This example illustrates how our fragment directly builds underspecified representations of scope ambiguities.

(39) Every linguist attends many workshops.

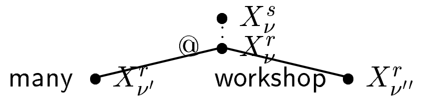
(40)



[2] This variant avoids the additional complexity that comes in by the fact that the existential quantifier is not scopally restricted.

First, the NP fragments are built from the constraints of the nouns and the determiners (appropriately labelled node variables) by the rules **(b9)** and **(b8)**. For *many workshops*, this returns the following constraint:

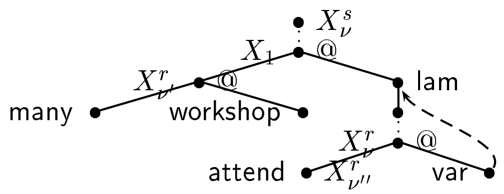
(41)



The semantics of the determiner carries the additional constraint $X^s_{\nu'} \triangleleft^* X^r_{\nu'}$, i.e., the topnode of the determiner is dominated by its scopenode, hence, also by the scopenode of the NP X^s_ν (recall that $X^s_\nu = X^s_{\nu'} = X^s_{\nu''}$). The derivation of the semantics of *every linguist* works analogously.

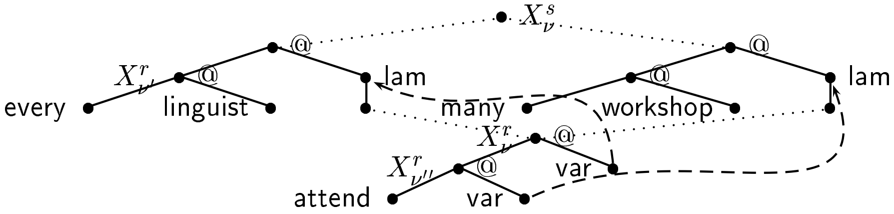
The rule **(b3)** combines the topnodes of verb and NP semantics into application trees (with a lam and a var node variable, respectively) and relates these new variables by the relation λ. The root of the second application tree is then the topnode of the VP. In addition, **(b3)** identifies the scopenodes of VP, verb, and NP. Accordingly, the first application tree (which comprises the NP semantics; this kind of tree fragment is referred to as *N-fragment*) is positioned between X^s_ν and X^r_ν. The scoping possibilities of an NP are represented in CLLS constraints by relating the root of its N-fragment (here, X_1) to X^s node variables in the constraint.

(42)



Finally, the sentence radical semantics is constructed from (42) and the constraint for *every linguist* by rule **(b1)**:

(43)



The semantic effect of this rule is pretty much the same as the one of rule **(b3)**: the topnode of the NP is combined in an application tree with a lam node variable (whose root is the topnode of S), the topnode of the VP, in an application tree with a var node variable. These new variables are related in terms of λ and the scopenodes of S, NP, and VP are identified. As a result, the N-fragment of the subject NP is placed somewhere between X^s_ν and X^r_ν. This directly introduces the quantifier scope underspecification into the semantic representation of (39): we only know that the two N-fragments dominate the topnode and are dominated by the scopenode, but it is unknown which fragment dominates the other one.

6. Efficient Disambiguation

A good underspecification formalism should not only allow to write down appropriate underspecified descriptions, but also to enumerate their readings efficiently. The most optimistic case is that they can be enumerated in time polynomial in the total number of readings.

In a framework that allows the analysis of ellipsis and anaphora in addition to scope, it is unlikely that such a low runtime can be obtained. A prime example is the Dalrymple et al. (1991) analysis of ellipsis: It uses higher-order unification, which is known to be undecidable.

Similarly, the problem of disambiguating arbitrary CLLS descriptions (that is, of enumerating all most general solved forms of a constraint) is very complex. It was shown in Niehren and Koller (1998) that CLLS is equivalent to context unification (Schmidt-Schauß and Schulz, 1998; Niehren et al., 1997a). There are semi-decision procedures for context unification, but its decidability is a prominent open problem in theoretical computer science (RTA, 1998).

For the sublanguage of dominance constraints (which are sufficient for scope underspecification alone), the situation is better: It was shown in Koller et al. (1998) that the satisfiability problem of dominance constraints is decidable, but NP-complete. This means that the most efficient solution procedure may take (deterministic) exponential time for enumerating all readings, even if the number of readings is polynomial.

Nevertheless, there are solvers for dominance constraints that perform very efficiently for all constraints from scope underspecification. These solvers are based on constraint programming with finite sets (Koller and Niehren, 1999); the general strategy is iterating deterministic *propagation* steps and non-deterministic choices. It turns out that despite the general NP-completeness, all choices made in the disambiguation of a constraint from scope underspecification are necessary to distinguish different readings; the search never fails. Recent research in this area focuses on obtaining formal complexity results that model this experience and on integrating a semi-decision procedure for the rest of CLLS with a dominance constraint solver. A demo system that derives dominance constraints describing a scope ambiguity and solves them efficiently is available on the WWW (CHORUS Project, 1999).

7. Conclusion and Outlook

In this paper, we have presented the Constraint Language for Lambda Structures (CLLS), a first-order language for semantic underspecification. CLLS allows the representation of scope ambiguities, anaphora, and ellipses in simple underspecified structures that are transparent and suitable for processing. It is a language of tree descriptions that conservatively extends the language of dominance constraints.

We have shown that CLLS correctly represents many notorious problems from the literature involving scope, anaphora, ellipses, and their interactions. An advantage over earlier approaches to this area is that CLLS analyses are completely order-independent, and the formal foundations are very transparent. It also provides a clean solution to a problem with variable binding in the context of scope underspecification that becomes most obvious in the presence of ellipses.

Although the linguistic coverage of CLLS includes various challenging problems, it still needs to be extended. The most obvious area for improvement is a more complete theory of ellipsis. Another interesting extension is to build dynamic accessibility relations into the definition of anaphoric links and make the effects of anaphora on quantifier scope explicit by reasoning on underspecified structures (Koller and Niehren, 2000). Finally, CLLS can be used to model reinterpretation (meaning shift) of aspect and NPs in an underspecified way (Striegnitz, 1999; Egg, 1999; Koller et al., 1999).

On the other hand, there are various more formal aspects of CLLS that can be pursued in the future. One essential problem for any underspecification formalism is to raise operations on the object level, such as deduction, to the level of underspecified descriptions. Although there are some first results (Jaspars and Koller, 1999), the entire field is still in its infancy. Furthermore, it is necessary to find tractable algorithms for processing parallelism constraints (Erk and Niehren, 2000).

Acknowledgements

This work was supported by the SFB 378 (project CHORUS) at the Universität des Saarlandes. The article is an updated and revised version of Egg et al. (1998) and Egg et al. (1999); we are indebted to

Peter Ruhrberg and Feiyu Xu for their contributions to these papers. The authors wish to thank Manfred Pinkal, Gert Smolka, all members of the SFB 378 projects CHORUS, LISA, NEGRA, and NEP, and the commentators and participants at the Bad Teinach workshop on underspecification.

References

Bos, J.: 1996, 'Predicate Logic Unplugged'. In: *Proceedings of the 10th Amsterdam Colloquium*. pp. 133–143.

CHORUS Project: 1999, 'The CHORUS Demo System'. `http://www.coli.uni-sb.de/cl/projects/chorus/demo.html`.

Crouch, R.: 1995, 'Ellipsis and Quantification: A Substitutional Approach'. In: *Proceedings of the 7th EACL*. Dublin, pp. 229–236.

Dalrymple, M., S. Shieber, and F. Pereira: 1991, 'Ellipsis and Higher-Order Unification'. *Linguistics & Philosophy* **14**, 399–452.

Egg, M.: 1999, 'Reinterpretation from a synchronic and diachronic point of view'. Arbeitspapiere der FG Sprachwissenschaft, Universität Konstanz. `http://www.coli.uni-sb.de/cl/projects/chorus/papers/egg99.html`.

Egg, M., A. Koller, J. Niehren, and P. Ruhrberg: 1999, 'Constraints over Lambda Structures, Antecedent Contained Deletion, and Quantifier Identities'. Manuscript. `http://www.coli.uni-sb.de/~koller/papers/acd.html`.

Egg, M., J. Niehren, P. Ruhrberg, and F. Xu: 1998, 'Constraints over Lambda-Structures in Semantic Underspecification'. In: *Proceedings COLING/ACL'98*. Montreal.

Erk, K. and J. Niehren: 2000, 'Parallelism constraints'. Submitted.

Fiengo, R. and R. May: 1994, *Indices and Identity*. Cambridge: MIT press.

Gawron, M. and S. Peters: 1990, *Anaphora and Quantification in Situation Semantics*, No. 19 in CSLI Lecture Notes. CSLI/University of Chicago Press.

Hirschbühler, P.: 1982, 'VP deletion and across the board quantifier scope'. In: J. Pustejovsky and P. Sells (eds.): *NELS 12*. Univ. of Massachusetts.

Jaspars, J. and A. Koller: 1999, 'A Calculus for Direct Deduction with Dominance Constraints'. In: *Proceedings of the Twelfth Amsterdam Colloquium*. Amsterdam.

Kehler, A.: 1993, 'A Discourse Copying Algorithm for Ellipsis and Anaphora Resolution'. In: *Proceedings of EACL*.

Kehler, A.: 1995, 'Interpreting Cohesive Forms in the Context of Discourse Inference'. Ph.D. thesis, Harvard University.

Koller, A. and J. Niehren: 1999, 'Constraint Programming Technology in Computational Linguistics'. Submitted. `http://www.coli.uni-sb.de/~koller/papers/cpcl.html`.

Koller, A. and J. Niehren: 2000, 'Underspecified Processing of Dynamic Semantics'. Submitted. `http://www.coli.uni-sb.de/~koller/papers/dynamic.html`.

Koller, A., J. Niehren, and K. Striegnitz: 1999, 'Relaxing Underspecified Semantic Representations for Reinterpretation'. In: *Proceedings of the Sixth Meeting on Mathematics of Language (MOL6)*. Orlando, Florida.

Koller, A., J. Niehren, and R. Treinen: 1998, 'Dominance Constraints: Algorithms and Complexity'. In: *Proceedings of the Third Conference on Logical Aspects of Computational Linguistics*. Grenoble. To appear in LNCS.

Muskens, R.: 1995, 'Order-Independence and Underspecification'. In: J. Groenendijk (ed.): *Ellipsis, Underspecification, Events and More in Dynamic Semantics.* DYANA Deliverable R.2.2.C.

Niehren, J. and A. Koller: 1998, 'Dominance Constraints in Context Unification'. In: *Proceedings of the Third Conference on Logical Aspects of Computational Linguistics.* Grenoble. To appear in LNCS.

Niehren, J., M. Pinkal, and P. Ruhrberg: 1997a, 'On Equality Up-to Constraints over Finite Trees, Context Unification, and One-Step Rewriting'. In: *Proceedings 14th CADE.* Townsville: Springer-Verlag.

Niehren, J., M. Pinkal, and P. Ruhrberg: 1997b, 'A Uniform Approach to Underspecification and Parallelism'. In: *Proceedings ACL'97.* Madrid, pp. 410–417.

Reyle, U.: 1993, 'Dealing with ambiguities by underspecification: construction, representation, and deduction'. *Journal of Semantics* **10**, 123–179.

RTA: 1998, 'Decidability of context unification'. The RTA list of open problems, number 90, `http://www.lri.fr/~rtaloop/`.

Sag, I. A.: 1980, *Deletion and Logical Form.* New York and London: Garland Publishing.

Schmidt-Schauß M. and K. Schulz: 1998, 'On the Exponent of Periodicity of Minimal Solutions of Context Equations'. In: T. Nipkow (ed.): *International Conference on Rewriting Techniques and Applications.* to appear.

Shieber, S., F. Pereira, and M. Dalrymple: 1996, 'Interaction of Scope and Ellipsis'. *Linguistics & Philosophy* **19**, 527–552.

Striegnitz, K.: 1999, 'On Modeling Meaning Shifts by Relaxing Underspecified Semantic Representations'. Diplom thesis, Universität des Saarlandes, Saarbrücken, Germany. `http://www.coli.uni-sb.de/cl/projects/chorus/papers/kris99.html`.

Xu, F.: 1998, 'Semantic Underspecification and Ellipsis Resolution'. Diplom thesis, University of the Saarland. `http://www.coli.uni-sb.de/cl/projects/chorus/papers/fei98a.html`.