Constraints over Lambda Structures, Antecedent Contained Deletion, and Quantifier Identities

Markus Egg, Alexander Koller, Joachim Niehren*, Peter Ruhrberg
Department of Computational Linguistics / *Programming Systems Lab
Universität des Saarlandes, Saarbrücken, Germany
{egg,koller,peru}@coli.uni-sb.de, niehren@ps.uni-sb.de

Abstract

The constraint language for lambda-structures (CLLS) allows for a simple, integrated, and underspecified treatment of scope, ellipses, anaphora, and their interaction. CLLS features constraints for dominance, lambda binding, parallelism, and anaphoric links. In the case of antecedent contained deletion (ACD), the definition of parallelism in the original version of CLLS is slightly too restrictive due to an overly weak notion of quantifier identity. We show how to extend CLLS with an appropriate notion of quantifier identity such that ACD can be naturally analysed. This sheds some light on conflicting requirements on quantifier representations as needed for ACD and Hirschbühler sentences.

Keywords: Natural Language Semantics, Underspecification, Scope, Parallelism, Constraints

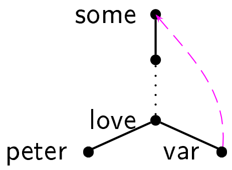
1 Introduction

The constraint language over lambda-structures CLLS is introduced in Egg et al. (1998) as a uniform framework for semantic underspecification. As argued there, CLLS allows for a simple, integrated, and underspecified treatment of scope, ellipses, anaphora, and their interaction. The linguistic coverage of CLLS is compatible with more traditional approaches based on higher-order unification (Dalrymple et al. 1991; Shieber et al. 1996) or Quasi Logical Form (Alshawi 1990; Crouch 1995) and the more recent glue language approach (Dalrymple et al. 1997; Gupta and Lamping 1998). It makes possible an adequate account of quantifier parallelism, as in Hirschbühler sentences, but also of pronoun puzzles raised by strict-sloppy readings of intrasentential anaphora.

CLLS is the most expressive language in the tradition of USDL (Pinkal 1996) and context unification (CU, Niehren et al. 1997b; Niehren et al. 1997a). The common characteristic of all these languages is that they all support the idea of "radical underspecification" (Pinkal 1996) – i.e. underspecification of arbitrary structures – by providing place holders within trees. For example, the following term of CU expresses that Peter loves someone or something, but leaves space open, by means of a place-holder C, for additional intervening material between the quantifier and the inner predication. Here, some_x corresponds to the existential quantifier $\exists x$.

$$\mathsf{some}_x(C(\mathsf{love}(\mathsf{peter}, \mathsf{var}_x)))$$

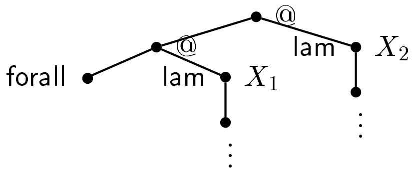
In CLLS, the same example would look like this:



The place holders are expressed in CLLS by dominance relations between nodes of the tree, as known from Marcus et al. (1983) or Rogers and Vijay-Shanker (1994), which are drawn as dotted lines in a picture. One of the main novelties of CLLS is the mechanism it provides for the description of variable binding. Variable occurrences are related to their binders by way of a *binding relation* between nodes labeled var and binding nodes labeled, e.g., with some or lam (for a lambda binder). This binding relation, drawn as dashed arrows, can be thought of as a "rubber band" between the binder and the variable; even though the space between these two nodes may be stretched by filling in more intervening material, they "never break". CLLS thus solves the capturing problem of some previous approaches to scope underspecification, where binding is expressed by coordination of variable names; for example, the CU example above does not express that the var_x is bound by the some_x because another some_x could be introduced within C. It also has computational advantages over CU, as discussed in Niehren and Koller (1998) and Koller (1998).

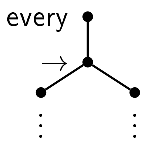
More traditional approaches to underspecification, such as UDRT (Reyle 1993) or QLF (Alshawi 1990; Crouch 1995), model quantifier scope by equipping quantifiers with a unique *identity* (called index or label). The position in a fully specified semantic structure to which a quantifier is raised is then associated with its identity. This mapping from quantifier identities to positions is described in an underspecified manner.

But this quantifier identity is not appropriately modeled in CLLS as it stands, which leads to difficulties with certain empirical phenomena, most notably antecedent contained deletion (ACD). Consider, for instance, the following higher-order representation of the universal quantifier in CLLS. Here, forall corresponds to the semantics of *every*, $\lambda P \lambda Q \forall x.P(x) \rightarrow Q(x)$.



In a sense, the quantifier is distributed over several nodes – not only the forall subtree, but also the two @ (application) nodes and the two lam (lambda abstraction) nodes; they

are only a tool that allows a more compositional notation, are introduced along with the forall, and disappear in a few steps of beta reduction. In other words, quantifier identity is a properly weaker relation than the identity of nodes that is considered in the original version of CLLS (Egg et al. 1998). In particular, the information that X_1 and X_2 are binders of the same quantifier (and, after beta reduction, are applied to the same variable, namely, x) is not represented. As we will show below, this is why this version cannot deal with ACD; an aim of this paper is to adapt the definition of CLLS to allow a proper treatment of ACD and introduce a notion of quantifier identity.
A first stab at the problem would be to simply use a first-order representation of quantifiers, where one (quantifying) binder is employed, rather than two lambda binders. Here, every corresponds to the universal quantifier $\forall x$.



But with this representation, we run into another problem: NPs like *every man* would no longer be analysed as subtrees. This feature is a necessary prerequisite for a proper treatment of the so-called Hirschbühler sentences in CLLS. Hence, the challenge is to define a quantifier representation in CLLS which satisfies the different requirements needed for ACD and Hirschbühler sentences:

1. quantifier identity must be expressed, and

2. NP interpretations must be represented as subtrees.

It seems impossible to capture quantifier identity by means of binding relations only (e.g. as it is the case in first-order logic) and satisfy condition (2) at the same time, if we maintain a natural condition for logical representations, namely that

3. variables are dominated by their binders.

We show how to introduce the concept of quantifier identity into CLLS so that ACD can be naturally analysed. We propose a notion of binding equivalence, which relates e.g. the nodes X_1 and X_2 in the above higher-order analysis of the universal quantifier. Binding equivalence is our way to spell out the notion of quantifier identity. We finally compare the approach to ones based on *Logical Forms* (May 1977), where the standard account of binding does not uphold condition (3).

2 CLLS

We introduce the basic notions of λ-structures and CLLS as far as required for the discussion in this paper. The full system of CLLS, as presented in Egg et al. (1998), also covers anaphora and their interaction with parallelism, but we will ignore these aspects here.

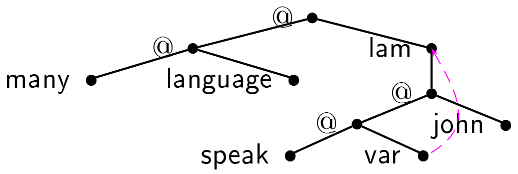
2.1 Lambda structures

A λ-structure is a tree structure that is extended by an additional relation on its nodes, the *binding* relation. Its nodes are labeled with symbols from a ranked signature, and the arity of each label determines the number of daughter nodes. Graphically, we represent a λ-structure as a graph with two types of edges: the underlying tree structure is drawn in the usual way with solid edges, and the binding relation is represented by drawing a dashed arrow from the *bound* node (that must be labeled with the special nullary symbol var) to the *binder* (a node labeled with the special unary symbol lam).

λ-structures are related to λ-terms very much like (constructor) trees to ground terms. Consider e.g. the higher-order logic formula (1) and its corresponding λ-structure (2).

(1) (many(language))(λx.speak(x)(john))

(2)



A constant term like many is represented by a tree with a single node labeled with many. An application MN corresponds to a λ-structure that has the binary symbol @ as its root node, the representation of the functor M as its left subtree, and the representation of the argument N as its right subtree. An abstraction $\lambda x.M$ is represented as a λ-structure whose root is labeled with lam and whose daughter is the representation of M. Variables x are represented by nodes labedeled with var. Furthermore, λ-structures provide a binary relation between nodes, which expresses the binding relation between λ-binders λx and the occurrences of the variables x that they bind. Thus, λ-structures correspond to λ-terms up to the consistent renaming of variables (α-equivalence).

2.2 Labeling and Binding Constraints

For an underspecified semantic description, we describe λ-structures with constraints of the language CLLS (constraint language for λ-structures), as introduced in Egg et al. (1998). These constraints are conjunctions of atomic formulae of several types which we will introduce one by one. A λ-structure L is *described* by such a constraint φ if there is a mapping of variables of φ (written in upper case) to nodes of L such that all the atomic conjuncts of φ are satisfied.

The most basic types of constraints in CLLS are labeling and binding constraints. *Labeling constraints* are of the form $X{:}f(X_1, \ldots, X_n)$, where f is an n-ary symbol from the signature; they express that the node denoted by X is labeled with f and has as children exactly the nodes denoted by $X_1, \ldots, X_n$. *Binding constraints* are written as $\lambda(X){=}Y$, saying that the node denoted by X is bound by the node Y (which must be a lambda binder). For example, a constraint describing the structure in (2) would be $X_1{:}@(X_2, X_3) \wedge X_3{:}\mathsf{lam}(X_4) \wedge X_5{:}\mathsf{many} \wedge \lambda(X_6){=}X_3$. Of course, this constraint is satisfied by many other structures as well; it can be extended by more conjuncts to obtain a complete description of the λ-structure in the example.

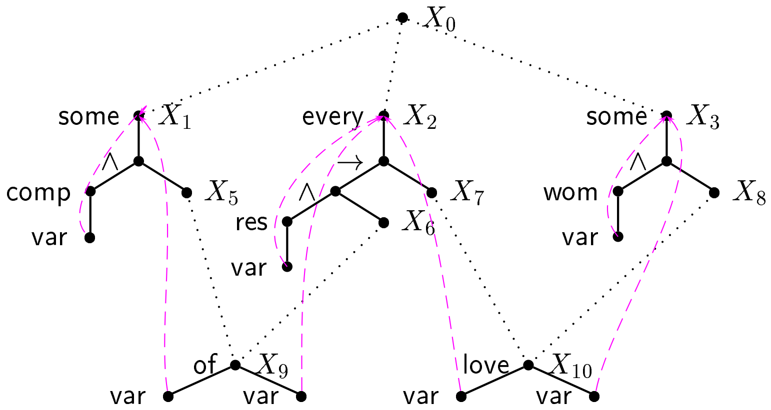
2.3 Dominance Constraints

For the treatment of scope underspecification, CLLS employs *dominance constraints* $X \triangleleft^* Y$, expressing that the node X dominates the node Y. As the dominance need not be immediate, more material could be filled in between nodes that dominate each other, which means that we can use this constraint as the "place holder" mentioned in the introduction. We will illustrate dominance constraints in their full power by giving a CLLS representation of the five-reading sentence (3).

(3) Every researcher of a company saw a woman.

In early approaches to scope underspecification, NP modifications as in this example were a problem. One cannot simply enumerate all permutations of the quantifiers at the sentence node; this would predict $n!$ readings for a sentence with n quantifiers, but instead of the predicted six ($= 3!$) readings, (3) has only five.
CLLS solves this problem easily: The constraint for the semantics of (3) is shown as a "constraint graph" in (4).

(4)



Several points of explanation about this picture are in order. First of all, the constraint graph is a graphical representation of a CLLS constraint. The nodes of the graph stand for variables in the constraint; sometimes, they are annotated with variable names. Labelings and solid lines represent labeling constraints, as discussed in the previous paragraph; dashed arrows represent binding constraints. The newly introduced dominance constraints are drawn as dotted lines. It is important to distinguish graphical representations of λ-structures from constraint graphs. They look very similar (in fact, (2) could not only be interpreted as the picture of a λ-structure, but also as the picture of a constraint that describes it), but the latter represent CLLS constraints, and the former, structures that can be described by them.
Furthermore, we have deviated from the higher-order analysis of the sentence we would normally employ; we are using first-order quantifiers as variable binders and deep atomic first-order formulae. In this example, this doesn't really make a difference to the higher-order case, but is more easily readable. In a higher-order analysis, we would use the usual

encoding of quantifiers via higher-order predicates with fixed interpretation instead (as in (2)) and still use lam nodes as binders, and instead of the deep atomic formulae, we would use trees of applications.

The constraint in (4) has a very specific structure that makes it easy to see its solutions. Due to the left "dominance diamond" (whose points are the root node X_0, the left and the middle quantifier X_1 and X_2, respectively, and X_9, the tree below which models the genitive relation), there are essentially two places for the left quantifier in a λ-structure that solves this constraint: above the universal quantifier, or below the universal quantifier's left daughter. The requirement that both dominate X_9 leaves no other options. Likewise, the right "dominance diamond" (whose bottom node is X_{10}, the tree below which models the love relation) allows two different classes of solutions, one where the right quantifier X_3 dominates the universal quantifier X_2, and one where it occurs below the universal quantifier's right daughter. It is crucial that there are two independent "dominance diamonds", each with its own "nucleus" at the bottom.

This yields five structurally different solutions to the constraint. If the existential quantifier X_1 dominates the universal quantifier X_2, the existential quantifier X_3 may be above, between, or below the other quantifiers, which gives three options. If, on the other hand, X_1 is dominated by X_2, there are only two different positions for the existential quantifier X_3, viz., either above or below the universal quantifier: The implication below the universal quantifier is a binary constructor, which makes sure that if the existential quantifiers are both dominated by the universal quantifier, they don't dominate each other.[1] These five solutions correspond to the five different readings of the sentence.

In some recent formalisms, such as Jaspars and van Eijck's Ambiguous Predicate Logic in van Eijck (1996), this expressive power is not available. Note that our approach preserves the intuitions of the analysis of Hobbs and Shieber (1987) but can do without restrictions on the order of resolution steps. It works essentially in the same way as the UDRT analysis (Reyle 1993); there is a correspondence between making the partial order on UDRSs more explicit and adding more dominance constraints to disambiguate the underspecified representation.

2.4 Parallelism Constraints

An important design goal of CLLS was to give an account of parallelism within a uniform semantic framework for underspecification, following (Niehren, Pinkal, and Ruhrberg 1997b). A benchmark case for their interaction are the so-called *Hirschbühler sentences* (Hirschbühler 1982) as in (5).

(5) a. Every linguist speaks two Asian languages.

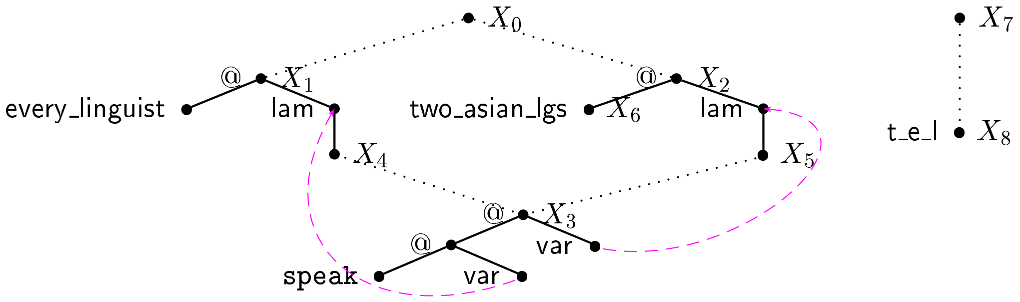
b. Two European ones, too.

The source sentence (5a) exhibits a scope ambiguity which can be captured by a CLLS constraint as indicated by the left part of constraint graph in (6); note that we have

[1]This is exactly what goes wrong in a six-reading analysis: if two quantifiers dominate each other, there are always two different readings, one for each relative order of the two.

returned to the higher-order notation, but have compressed small subtrees into single nodes for easier readability. The right part of the graph expresses everything we initially know about the semantics of the target sentence (5b): Somewhere inside, it contains the semantics of *two European languages* (abbreviated as node with the label t_e_l).

(6)



(a) (b)

The elliptical construction gives us the additional information that the structure of the elliptical target sentence must be parallel to that of the ambiguous source sentence; in particular, it contains an analogous scope ambiguity. But although both sentences allow for two scopings of their quantifiers, the parallelism requires that the relative positions of parallel quantifiers are the same in both sentences. As a consequence, the two sentences don't have a total of four different readings, but only two – although each of them is ambiguous in two ways.

In CLLS, we analyze elliptical constructions by means of *parallelism constraints*. These constraints have the form

$$X_s/X_p{\sim}Y_t/Y_p,$$

which means that the trees under the nodes denoted by X_s and Y_t must be equal *up to* the exception positions X_p and Y_p; the trees under these exception nodes can be (and normally are) different. One requirement of using these constraints to analyze ellipses is that the parallel elements in elliptical constructions are given as subtrees (for gapping, as sequences of subtrees) of the overall structure, which restricts our choice of the semantic representation language.

In the example (6), we require that the following constraint must be satisfied:

(7) $X_0/X_6{\sim}X_7/X_8$

Intuitively, this means that the semantics of both sentences are the same up to the meaning of the parallel elements (here, the object NPs); these must be plugged into the rest of the semantics in the same place.

This treatment of Hirschbühler sentences is descriptively equivalent to the one presented in (Niehren, Pinkal, and Ruhrberg 1997b). Our parallelism constraints and their *context constraints* have been shown to be (non-trivially) intertranslatable (Niehren and Koller 1998; Niehren, Pinkal, and Ruhrberg 1997a).

2.5 Syntax and Semantics of CLLS

We now give a more precise definition of the syntax and semantics of CLLS. As mentioned above, we omit anaphoric links because they are irrelevant for the scope of this paper; they are described in detail in Egg et al. (1998).
We start with a ranked signature $\Sigma = \{@^2, \mathsf{lam}^1, \mathsf{var}^0, \mathsf{greet}^0 \ldots\}$, ranged over by f^i; the arities i may be omitted. The syntax of CLLS constraints φ is given by

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

Constraints are interpreted over λ-structures L, which are formally tree structures with an additional binding relation. Each relation symbol $\zeta \in \{:f \mid f \in \Sigma\} \cup \{\triangleleft^*, \lambda(\cdot)=\cdot, \cdot/\cdot{\sim}\cdot/\cdot\}$ is interpreted as a relation ζ_L over the nodes; the interpretations of the ζ_L are fixed.
A (finite) tree structure is given by a set of *nodes* $\nu, \nu', \ldots$ that are connected by *paths* $\pi, \pi', \ldots$ (possibly empty words over positive integers), and a *labeling function* l from nodes to labels. The number of daughters of a node matches the arity of its label. This determines the interpretation of constraints over this structure. For each symbol $f^i \in \Sigma$, the labeling constraint $X : f(X_1, \ldots, X_i)$ is interpreted as true iff the denotations of the variables are in the relation $:f_L$. $\nu{:}f_L(\nu_1, \ldots, \nu_n)$ holds iff $l(\nu)=f$ and $\nu.i = \nu_i$ for $i = 1..n$, where $\nu.\pi$ stands for the node that is reached from ν by following the path π (if defined). To express that a path π is defined on a node ν in L we write $\nu.\pi{\downarrow}_L$. We write $\pi \leq \pi'$ for π being an initial segment of π'. Dominance constraints are interpreted into the *dominance* relation: $\nu \triangleleft_L^* \nu'$ holds if $\exists \pi\ \nu.\pi = \nu'$. If π is non-empty we have *proper dominance* $\nu \triangleleft_L^+ \nu'$.
A *λ-structure* L is a tree structure that additionally satisfies the following conditions on the *λ-binding* relation $\lambda_L(\cdot)=\cdot$:

1. binding only holds between variables and λ-binders:
   $\forall\nu\forall\nu' \lambda_L(\nu)=\nu' \Rightarrow (\nu{:}\mathsf{var}_L \wedge \exists\nu''\ \nu'{:}\mathsf{lam}_L\nu'')$
2. every variable has a binder:
   $\forall\nu\ \nu{:}\mathsf{var}_L \Rightarrow \exists\nu'\ \lambda_L(\nu)=\nu'$
3. every variable has at most one binder:
   $\forall\nu\forall\nu'\forall\nu''(\lambda_L(\nu)=\nu' \wedge \lambda_L(\nu)=\nu'') \Rightarrow \nu'=\nu''$
4. variables are dominated by their binders:
   $\forall\nu\forall\nu' \lambda_L(\nu)=\nu' \Rightarrow \nu' \triangleleft_L^* \nu$

The most involved definition is that of parallelism. The parallelism constraint

$$\nu_1/\nu_1'{\sim_L}\nu_2/\nu_2'$$

is interpreted as true iff there is a path π_0 such that the following conditions hold:

1. π_0 is the "exception path" from the top node of the parallel structures to the two exception positions: $\nu_1'{=}\nu_1.\pi_0 \wedge \nu_2'{=}\nu_2.\pi_0$

2. the two *contexts*, which are the trees below ν_1 and ν_2 up-to the trees below the exception positions ν_1' and ν_2', must have the same structure and labels:
$\forall\pi\ \neg\pi_0{\leq}\pi \Rightarrow ((\nu_1.\pi{\downarrow_L} \Leftrightarrow \nu_2.\pi{\downarrow_L}) \wedge (\nu_1.\pi{\downarrow_L} \Rightarrow l(\nu_1.\pi) = l(\nu_2.\pi)))$

3. there are no 'hanging' binders from the contexts to variables outside them:
$\forall\nu\forall\nu'\ \neg(\nu_1{\lhd_L^*}\nu{\lhd_L^+}\nu_1'{\lhd_L^*}\nu' \wedge \lambda_L(\nu'){=}\nu)$

4. binding is structurally isomorphic within the two contexts:
$\forall\pi\forall\pi'\ \neg\pi_0{\leq}\pi \wedge \nu_1.\pi{\downarrow_L} \wedge \neg\pi_0{\leq}\pi' \wedge \nu_1.\pi'{\downarrow_L}\ \Rightarrow\ (\lambda_L(\nu_1.\pi){=}\nu_1.\pi' \Leftrightarrow \lambda_L(\nu_2.\pi){=}\nu_2.\pi')$

5. two variables in identical positions within their context and bound outside their context must be bound by binding equivalent binders:
$\forall\nu\forall\nu'\forall\pi\ (\nu{\lhd_L^+}\nu_1 \wedge \nu'{\lhd_L^+}\nu_2 \wedge \neg\pi_0{\leq}\pi \wedge \nu_1.\pi{\downarrow_L} \wedge \lambda_L(\nu_1.\pi){=}\nu \wedge \lambda_L(\nu_2.\pi){=}\nu') \Rightarrow$
$\nu{\approx_L}\nu'$.

The last condition of this definition makes reference to the relation $\approx_L$. This relation is called the *binding equivalence* relation, and we will use it to model quantifier identity. In the next section, we will present some of the considerations for what nodes should be binding equivalent and give a definition of this relation, but for now think of it simply as node identity; doing so restores the semantics of parallelism from the original definition of CLLS, where parallel variables that are bound outside of the parallel construction must be bound by the *same* binder. This is certainly safe, but as we will see, too restrictive in the case of ACD.

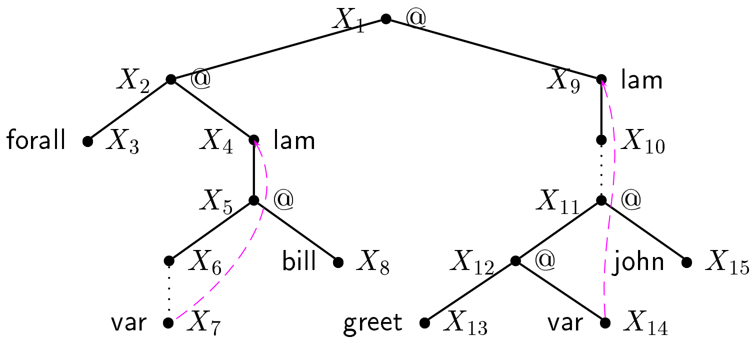
3 ACD in CLLS

Consider (8), a simple case of antecedent contained deletion. For expository purposes we assume that the relative clause semantics constitutes the restriction of the universal quantifier and ignore further complications that arise from the treatment of relative clauses.

(8) John greeted everyone that Bill did.

The constraint we obtain for (8) is given in figure (9).

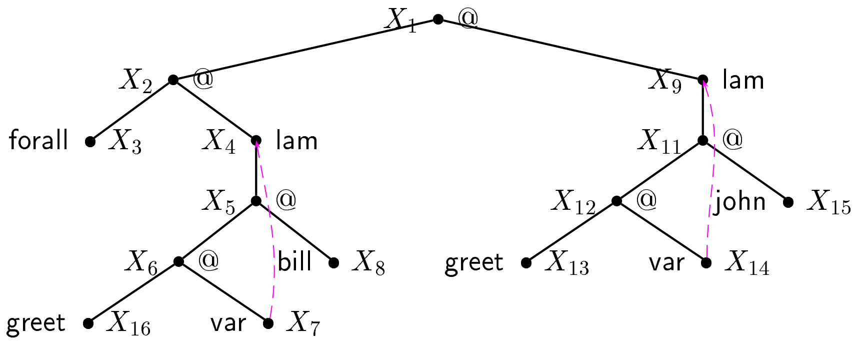
(9)



$X_5/X_8{\sim}X_{11}/X_{15}$

The intended solution is displayed in figure (10).

(10)



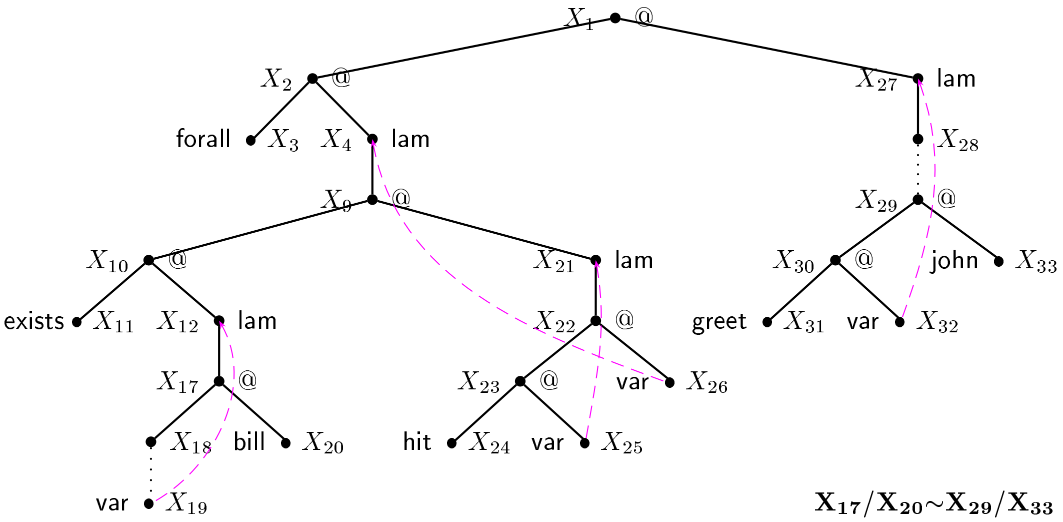
This λ-structure satisfies all the requirements of the parallelism constraint $X_5/X_8{\sim}X_{11}/X_{15}$ except for the critical question of the parallelism of the two variable nodes X_7 and X_{14} (clause 5 of the definition). The preliminary idea of taking binding equivalence to be node identity would require the binders of X_{14} and its parallel copy, X_7, to be the same, but here they are different nodes. Clearly, this interpretation of binding equivalence is too narrow for a case like (8).[2]

On the other hand, neither can we simply drop any restrictions on variables bound outside the parallel contexts altogether, as this would allow for the construction of non-existent readings for sentences like (11) with its (simplified) semantic constraint (12).

(11) John greeted everyone who hit someone that Bill did.

[2]An ad hoc remedy for this case would be to raise the level of parallelism slightly in order bring the two binders into the parallel contexts, but this does not work in general and diminishes the chances to find a reasonable algorithm for identifying domains of parallelism.

(12)



A sentence like (11) has no possible reading according to Fiengo and May (1994), in particular not one which can be represented as

(13) John greeted everyone who hit someone$_i$ that Bill greeted $\mathbf{e}_i$.

where the quantifier *someone* – and not the quantifier *everyone* – binds the variable $\mathbf{e}_i$ in the elided context.[3] But an unrestricted notion of binding equivalence, where all nodes are considered binding equivalent, would admit such a reading.

3.1 Binding equivalence

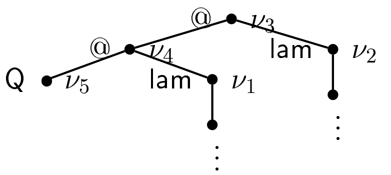
The correct definition of binding equivalence is the one that corresponds most closely to the concept of quantifier identity that we discussed earlier. This becomes evident when you consider that in (10), the binding nodes are the binders of the scope and restriction of the same quantifier, whereas in (12), that is not the case.

Looking at the problem from a different perspective, two binder nodes ν_1, ν_2 should be binding equivalent if the variables they bind can be mapped to the same variables by a series of beta reductions. In all common natural-language quantifiers, this is true; for example, in the higher-order analysis of universal quantification, $\lambda F \lambda G \forall x.(Fx \to Gx)$, both arguments are applied to the variable x, which is replaced for the variables bound both by ν_1 and ν_2. This situation is displayed graphically in (a) below; in the picture, Q stands for a quantifier. So in a way, what gives a quantifier its identity is that it mediates an identity between the binders of its arguments.

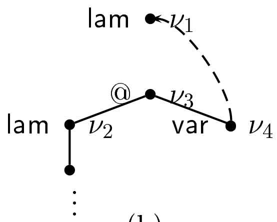
Unfortunately, checking this condition is computationally infeasible because it requires the computation of implicit beta reductions on the constraint graph. Even worse, beta reduction on underspecified lambda terms is not even always defined; only if there is enough information on the structure of the redex can we reduce. It is ongoing research to determine the exact conditions. Thus, we restrict ourselves to the most frequent source

[3] This example is modeled after example (18) of Fiengo and May (1994, p.245).

of this effect, quantifiers and application/abstraction chains as they are introduced by more complex cases of semantic construction. This more modest approach circumvents the computational and logical problems of beta reduction while remaining sufficiently powerful for the current task.
The case of application/abstraction chains is shown in the picture (b). Clearly, all variables bound by the node ν_2 will be replaced by variables bound by ν_1 when we reduce at ν_3. Hence, ν_1 and ν_2 must be considered binding equivalent.



(a)



(b)

Formally, we fix a finite set $\mathcal{Q} = \{\mathsf{every}, \mathsf{some}, \mathsf{most}, \ldots\}$ of quantifiers and introduce *binding equivalence constraints* $X{\approx}Y$ as a new type of constraint that is satisfied iff the nodes denoted by X and Y are in the binding equivalence relation $\approx_L$. We define this relation to be the smallest relation that satisfies the following conditions:

1. $\nu_1{\approx_L}\nu_2$ holds if there are nodes $\nu_1', \nu_2', \nu_3, \nu_4, \nu_5$ such that the following relations are true: $\nu_1{:}\mathsf{lam}_L(\nu_1')$, $\nu_2{:}\mathsf{lam}_L(\nu_2')$, $\nu_3{:}@_L(\nu_4, \nu_2)$, $\nu_4{:}@_L(\nu_5, \nu_1)$, and $\nu_5{:}Q_L$, where $Q \in \mathcal{Q}$. (Picture (a).)

2. $\nu_1{\approx_L}\nu_2$ holds if there are nodes $\nu_1', \nu_2', \nu_3, \nu_4$ such that the following relations are true: $\nu_3{:}@_L(\nu_4, \nu_2)$, $\nu_1{:}\mathsf{lam}_L(\nu_1')$, $\nu_2{:}\mathsf{lam}_L(\nu_2')$, $\nu_4{:}\mathsf{var}_L$, $\lambda(\nu_4) = \nu_1$. (Picture (b).)

3. $\approx_L$ is reflexive, transitive, and symmetric.

With this new definition for $\approx_L$, we have changed the semantics of parallelism constraints from Section 2.5, where we simply used node identity for this relation. Applying the new definition, it is easy to see that in example (9), X_4 and X_9 are binding equivalent, and therefore, (10) is a valid solution. On the other hand, in example (12), X_{12} and X_{27} are not binding equivalent; this means that the parallelism constraint cannot be resolved, and the constraint is unsatisfiable.

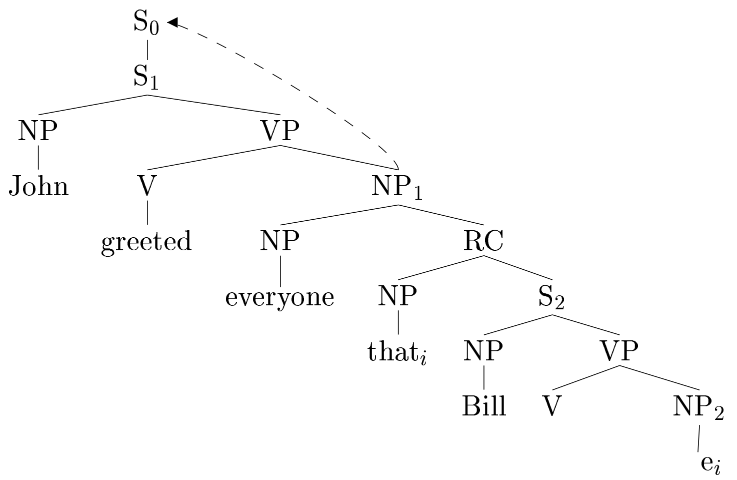
3.2 Lambda Structures and Logical Form

We conclude the paper with some remarks on related approaches. Our approach has obvious affinities to work that uses Logical Forms (LF) of GB theory (May 1977; May 1985). E.g., Muskens (1998) proposes to treat underspecification by means of first order constraints over tree structures that are highly reminiscent of expressions of LF structures. This resembles our use of first order constraints over lambda structures. In particular, Muskens' descriptions of LFs crucially use dominance constraints. His LFs differ from descriptions of surface syntactic structures in that there are additional (non-surface) sentence nodes to each of which one quantifier NP bears a linking relation. The position of these additional nodes indicates the semantic scope of the quantifier. LFs themselves are unambiguous. Underspecification enters in that a description of a surface syntactic structure can be satisfied by different LFs, just as different lambda structures may satisfy one CLLS constraint.

Muskens does not discuss ellipsis, but it is clear that our parallelism constraints look like a promising approach in his setting. But a naive extension of his approach as it stands would not work for ACD cases. The problem with ACD is that one may easily end up with structures in which the reconstructed constituent itself contains an elided phrase that once more calls for reconstruction. To avoid this danger of an infinite recursion, the crucial idea of Sag (1976) was to reconstruct an elided phrase by copying a constituent that contains only a variable (or trace) rather than a whole quantifier. This copying happens at LF, since, to this end, the quantifier must be moved outside the constituent that is to be copied.

Consider a representation for example (8) in Muskens' version of LF:

(14)



Muskens does not move quantified NPs, but instead links them to additional S nodes. In (8), for example, NP_1 is linked to S_0. Hence, if reconstruction was to be expressed (naively) on Muskens-like structures in the spirit of our approach, viz., in terms of an isomorphy requirement between the nodes S_1 and S_2, up to John and Bill, the regress problem would occur immediately.

This suggests that if one wants to use CLLS-like parallelism constraints, one must really raise quantifiers, as it is done in more standard styles of LF, such as the one employed by Fiengo and May (1994) in their discussion of parallelism. LFs are, of course, tree structures which have a closer affinity to syntax than lambda structures have. One important difference to our approach is that in LF, the binding relation from variables/traces to their binders is not constrained by dominance but by c-command relations between the "moved" NP and its "trace".[4] Binding is usually expressed as coindexing, but an explicit

[4] A node α c-commands another node β in a tree structure iff every maximal projection dominating α

relation between nodes would do as well. This difference makes it possible to express quantifier identity by a unique index associated with each quantifier, while at the same time representing NP structures as subtrees.

Note that the CLLS requirement of binding equivalence between binders for the scope and the restriction of a quantifier is implemented in LF as the requirement of coindexing a quantified NP with a relative pronoun that is part of the restricting $\bar{\mathrm{N}}$ constituent of the NP. For instance, Fiengo & May's representation of (8) is (15); boldface indicates copied constituents.

(15) [everyone that$_i$ Bill **greet** $\mathbf{e}_i$]$_i$ John greet e_i.

Here, the coindexing requirement of the moved NP [*everyone that...*]$_i$ with its internal relative pronoun *that*$_i$ is precisely the sort of equivalence requirement between two binders that we have found necessary for CLLS too, in order to insure that the two traces/variables can be seen as parallel.

A comparison of our analysis (12) of the example (11) to the one of Fiengo and May (1994) shows similarities and differences of the two approaches. Both our approach and the one of Fiengo and May (1994) put the ungrammaticality of (11) down to a conflict between two requirements: first, the relation between the elliptical constituent and its parallel element, and second, the binding relations introduced by the embedded quantifier *someone* and its relative pronoun within the parallel element (the restriction of this quantifier).

The difference in derivation between the two analyses is that in our approach, semantic construction first establishes the bindings for the variables within the elliptical constituent, which consequently fails to enter the required parallelism relation with its parallel constituent. Fiengo and May (1994), on the other hand, directly copy the parallel element with its traces *and their indices* into the position of the elliptical element, which results in a structure that falls prey to a general restriction on binding relations, viz., the requirement for *wh*-phrases to bind a trace. Hence their approach would not even consider the non-reading (13) for our example (12).

Their (simplified) analysis of the LF of (12) is given in (16):

(16) [$_{\mathrm{IP}}$ [$_{\mathrm{NP}}$ everyone who$_1$ [$_{\mathrm{IP}}$ e$_1$ [$_{\mathrm{VP}}$ [$_{\mathrm{NP}}$ someone that$_2$ Bill [$_{\mathrm{VP}}$ **greeted** $\mathbf{e}_1$]]$_2$ [$_{\mathrm{VP}}$ hit e$_2$]]]]$_1$ [$_{\mathrm{IP}}$ Bill [$_{\mathrm{VP}}$ greeted e$_1$]]]

Both quantifiers have been raised at LF; the universal quantifier has been adjoined to IP, the existential one, to the VP of the relative clause within the restriction of the universal quantifier. Due to the reconstruction of the elliptical VP (by the copying of **greeted** $\mathbf{e}_i$) in the above structure, *that*$_2$ fails to bind a trace, which renders (16) ungrammatical.

In sum, in this section we have shown that it is possible to integrate in CLLS both the requirement of quantifier identity and the need to assign to quantifiers a representation that constitutes a subtree in the chosen representation language. A comparison to other, more syntax-based approaches shows that our approach shares with them the same intuition about why examples like (12) are ungrammatical.

dominates β, and α does not dominate β (May 1985).

4 Conclusion

In this paper, we presented an extension of the constraint language for lambda-structures (CLLS) that allows for an adequate treatment of ACD cases. The solution addresses an issue which must be overcome in any approach to an underspecified treatment of ACD: Seemingly conflicting constraints on the representation on quantifiers in a logical formalism, namely the need for expressing quantifier identities and the representation of quantifiers as subtrees, must be reconciled. We also compared our approach to other, more syntax-oriented but otherwise very similar approaches.

Acknowledgements

This work was supported by the SFB 378 (project CHORUS) at the Universität des Saarlandes. The authors wish to thank Feiyu Xu, who first pointed out the difficulty with ACD in CLLS, and Manfred Pinkal for helpful discussions.

References

Alshawi, H. (1990). Resolving quasi logical form. *Computational Linguistics 16*, 133–144.

Crouch, R. (1995). Ellipsis and quantification: A substitutional approach. In *Proceedings EACL'95*, Dublin, 229–236.

Dalrymple, M., J. Lamping, F. Pereira, and V. Saraswat (1997). Quantifiers, anaphora, and intensionality. *Journal of Logic, Language, and Information 6*, 219–273.

Dalrymple, M., S. Shieber, and F. Pereira (1991). Ellipsis and higher-order unification. *Linguistics & Philosophy 14*, 399–452.

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

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

Gupta, V. and J. Lamping (1998). Efficient linear logic meaning assembly. In *Proceedings ACL'98*, Montreal.

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

Hobbs, J. and S. Shieber (1987). An algorithm for generating quantifier scoping. *Computational Linguistics 13*, 47–63.

Koller, A. (1998). Evaluating context unification for semantic underspecification. In *Proceedings of the Third ESSLLI Student Session.*

Marcus, M., D. Hindle, and M. Fleck (1983). D-theory: Talking about talking about trees. In *Proceedings of the 21st ACL*, 129–136.

May, R. (1977). *The Grammar of Quantification.* Doctoral dissertation, MIT, Cambridge.

May, R. (1985). *Logical Form. Its structure and derivation.* Cambridge: MIT Press.

Muskens, R. (1998). Underspecified semantics. Technical Report 95, Universität des Saarlandes, Saarbrücken. To appear.

Niehren, J. and A. Koller (1998). Dominance Constraints in Context Unification. *Logical Aspects of Computational Linguistics*, Grenoble. `http://www.ps.uni-sb.de/Papers/abstracts/Dominance.html`.

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, 410–417.

Pinkal, M. (1996). Radical underspecification. In *Proceed. 10th Amsterdam Colloquium*, 587–606.

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

Rogers, J. and K. Vijay-Shanker (1994). Extracting trees from their descriptions: an application to tree-adjoining grammars. *Computational Intelligence 10*, 401–421.

Sag, I. (1976). *Deletion and Logical Form.* Ph. D. thesis, MIT, Cambridge.

Shieber, S., F. Pereira, and M. Dalrymple (1996). Interaction of scope and ellipsis. *Linguistics and Philosophy 19*, 527–552.

van Eijck, J. (1996). [chapter 10.2 of FraCaS Report D15]. In FraCaS Consortium (ed.), *Building the Framework.* University of Edinburgh.