Well-Nested Parallelism Constraints for Ellipsis Resolution

Katrin Erk and Joachim Niehren
Saarland University, Saarbrücken, Germany
erk@coli.uni-sb.de / niehren@ps.uni-sb.de

Abstract

The Constraint Language for Lambda Structures (CLLS) is an expressive tree description language. It provides a uniform framework for underspecified semantics, covering scope, ellipsis, and anaphora. Efficient algorithms exist for the sublanguage that models scope. But so far no terminating algorithm exists for sublanguages that model ellipsis. We introduce *well-nested parallelism constraints* and show that they solve this problem.

1 Introduction

Ellipsis phenomena are ubiquitous in natural language, e.g. in VP ellipsis, answers to questions, and corrections. They have been studied extensively (Sag, 1976; Williams, 1977; Fiengo and May, 1994; Dalrymple et al., 1991; Hardt, 1993; Kehler, 1995; Lappin and Shih, 1996) but remain difficult to handle. Among the problems to solve in connection with ellipsis are: determining the ellipsis antecedent, constructing a description of the ellipsis meaning, and resolving the ellipsis (i.e. actually determining its meaning). In this paper we focus on the problem of *resolving ellipsis*. We assume an analysis of its structure (source, target, and parallel elements) in the Constraint Language for Lambda Structures (CLLS) (Egg et al., 2001).

CLLS is an expressive tree description language that provides a uniform framework for semantic underspecification covering scope, ellipsis, and anaphora. CLLS offers dominance constraints for modeling scope ambiguity in a similar way as previous approaches (Reyle, 1993; Pinkal, 1995; Bos, 1996), parallelism constraints for modeling ellipsis, and anaphoric links for modeling coreference. The interaction of ellipsis with scope (quantifier parallelism) is handled in a modular fashion. Enumerating scope readings becomes solving dominance constraints, while ellipsis resolution is reduced to solving parallelism constraints.

Constraint solving subsumes satisfiability checking. Satisfiability of dominance constraints is NP-complete (Koller et al., 2001). But for modeling scope underspecification a sublanguage of constraints suffices. These constraints can be solved in low polynomial time (Althaus et al., 2002). Parallelism constraints are as expressive as the language of Context Unification, the satisfiability problem of which is prominent but still open (Comon, 1992). A lower bound is given by string unification (Makanin, 1977), for which the best known algorithm runs in PSPACE.

So far, no terminating algorithm exists for sublanguages of CLLS that model ellipsis. The sound and complete semi-decision procedure for CLLS (Erk et al., 2002) can be used for this purpose but is slow in practice and not guaranteed to terminate.

In the current paper we introduce *well-nested parallelism constraints* and so solve this problem for the first time. We argue that well-nested parallelism constraints are powerful enough to model ellipsis, in particular VP-ellipsis. We present a solver for well-nested parallelism constraints which decides satisfiability in nondeterministic polynomial time, and hence proves the NP-completeness of this problem, as dominance constraints are subsumed.

2 CLLS

We represent the meaning of sentences by lambda terms, which are seen as trees and then described

by formulas of CLLS. The most basic formulas of CLLS are dominance constraints (Marcus et al., 1983). They model scope ambiguity in an underspecified way such that the solved forms of a constraint correspond precisely to the readings of a scopally ambiguous sentence.

Next we look at a simple example to see how ellipsis is modeled in this setting.

(1) Mary sleeps, and John does, too.

Fig. 1 (a) shows the meaning of sentence (1) as a tree. The source *Mary sleeps* has the same meaning as the target *John does, too* except that the contribution of the source parallel element *Mary* is replaced by the one of the target parallel element *John*. In the tree in Fig. 1, this is reflected in the two shaded *tree segments* having the same structure.

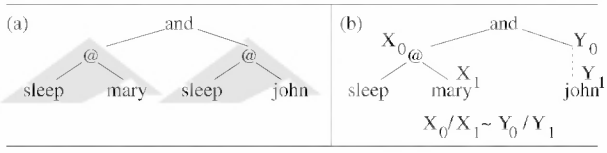


Figure 1: (a) The semantics of sentence (1), and (b) a CLLS description

Next we look at an idealized CLLS constraint that a syntax/semantics interface could produce for the above sentence. The graph of this constraint is given in Fig. 1 (b).

The semantics of the source starts at node X_0, the semantics of the target at Y_0. The source parallel element starts at X_1 and the target parallel element at Y_1. The graph contains an explicit description of the source semantics, but leaves the semantics of the target (mostly) unspecified. However the target semantics is described by the parallelism constraint $X_0/X_1 \sim Y_0/Y_1$, which states that the tree segment X_0/X_1 has the same structure as the tree segment Y_0/Y_1.

CLLS models coreference by anaphoric links. The interaction of ellipsis and anaphora (strict/sloppy ambiguity) is modeled by copying rules, which result in link chains equivalent to Kehler's (1995) analysis.

For modeling more complex classes of ellipsis, generalizations of parallelism constraints are needed: parallelism segments with more than one hole, and jigsaw parallelism, (Erk and Koller, 2001), which is used for cases where the excluded semantic contributions are not subtrees, as in "John went to the station, and every student did too, on a bike." The approach we describe in this paper extends canonically to segments with more than one hole. For jigsaw parallelism the extension remains a topic of further research.

3 Parallelism Constraints

In the following sections we restrict ourself to the language of parallelism constraints: CLLS without anaphoric links. However our results extend to the whole language of CLLS. We comment on this further in Sec. 7.

We first briefly recall the definition of parallelism constraints.

Trees. We assume a signature $\Sigma = \{f, g, \ldots\}$ of function symbols, each equipped with an arity $\mathsf{ar}(f) \geq 0$. A *tree* is a ground term over Σ. A *node* of a tree can be identified with its *path* from the root down, expressed by a word over $\mathbb{N}$. We use the letters u, v for paths. We write ε for the empty path and uv for the concatenation of two paths u and v. A tree τ consists of a finite set of nodes $u \in D_\tau$, each of which is labeled by a symbol $L_\tau(u) \in \Sigma$. Each node u has a sequence of children $u1, \ldots, un \in D_\tau$ where $n = \mathsf{ar}(L_\tau(u))$ is the arity of the label of u. A single node ε, the *root* of τ, is not the child of any other node.

A tree defines the following relations. The labeling relation $u{:}f(u_1, \ldots, u_n)$ holds in τ if $L_\tau(u) = f$ and $u_i = ui$ for all $1 \leq i \leq n$. The *dominance* relation $u \triangleleft^* v$ holds iff there is a path u' such that $uu' = v$. *Inequality* $\neq$ is simply inequality of nodes; *disjointness* $u \perp v$ holds iff neither $u \triangleleft^* v$ nor $v \triangleleft^* u$. We combine dominance and inequality into *strict dominance* $u \triangleleft^+ v$, which holds iff both $u \triangleleft^* v$ and $u \neq v$.

Parallelism. Intuitively, a segment is an occurrence of a subtree from which another subtree has been cut out.

Definition 3.1 (Segments). *A segment α of a tree τ is a pair u_0/u_1 of nodes in D_τ such that $u_0 \triangleleft^* u_1$ holds in τ. The* root *of the segment is u_0, and its*

$$
\begin{array}{rcl}
P, Q & ::= & X{\lhd^*}Y \mid X \perp Y \mid X{\neq}Y \\
& \mid & X{:}f(X_1, \ldots, X_n) \quad (\mathsf{ar}(f) = n) \\
& \mid & A{\sim}B \mid P \wedge Q \\
A, B, C & ::= & X/Y
\end{array}
$$

Figure 2: The language of parallelism constraints

hole *is* u_1. *The set* $\mathsf{b}_\tau(\alpha)$ *of* inner nodes *of* α *is:*

$$\mathsf{b}_\tau(\alpha) = \{v \in D_\tau \mid u_0 {\lhd^*} v, \neg(u_1 {\lhd^+} v)\}$$

The proper inner $\mathsf{b}^-_\tau(\alpha) = \mathsf{b}_\tau(\alpha) - \{u_1\}$ *excludes the hole* u_1. *A segment* α *is* empty *iff* $u_0 = u_1$.

A correspondence function is an isomorphism between two segments of some tree that have the same structure.

Definition 3.2 (Correspondence function). *A* correspondence function *between two segments* α, β *is a bijective mapping* $c : \mathsf{b}_\tau(\alpha) \to \mathsf{b}_\tau(\beta)$ *such that* c *maps the root of* α *to the root of* β *and the hole of* α *to the hole of* β, *and for every* $u \in \mathsf{b}^-_\tau(\alpha)$ *and every label* f, $u{:}f(u1, \ldots, un) \Leftrightarrow c(u){:}f(c(u1), \ldots c(un))$.

Corresponding nodes bear the same labels and have corresponding children, except for the holes.

Definition 3.3 (Parallelism relation). *A* parallelism relation *in a tree* τ *is a two-place relation* $\alpha \sim \beta$ *on segments of* τ *such that* $\alpha \sim \beta$ *implies the existence of a correspondence function between* α *and* β.

Constraint Language. We assume an infinite set of node variables X, Y, Z. Figure 2 shows the language of parallelism constraints.

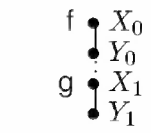
A constraint P is a conjunction of *literals* (for dominance, labeling, parallelism etc). We use the abbreviations $X{\lhd^+}Y$ for $X{\lhd^*}Y \wedge X{\neq}Y$ and $X = Y$ for $X{\lhd^*}Y \wedge Y{\lhd^*}X$. For simplicity, we view inequality ($\neq$) and disjointness ($\perp$) literals as symmetric. A *segment term* A is a pair of node variables X/Y. A parallelism literal relates two segment terms. We write $V(P)$ for the set of variables of P. The *dominance part* of P is P without its parallelism literals.

A tuple $(\tau, \sim, \sigma)$ of a tree τ, a parallelism relation $\sim$ and a variable assignment σ *satisfies* a constraint P iff it satisfies each literal, in the obvious way. In that case, $(\tau, \sim, \sigma)$ is a solution, and $(\tau, \sim)$ a model of P.

Dominance constraints can be drawn as constraint graphs, like in Fig. 1 (b). The nodes of the constraint graph are the variables of the constraint. Labels and solid lines indicate labeling literals, dotted lines represent dominance.

4 Well-Nested Parallelism

Parallelism constraints are very expressive – more expressive than is necessary for modeling ellipsis. In particular, overlapping parallel segments seem useless, but are difficult to resolve. Consider the example on the right. The parallel segments X_0/X_1 and Y_0/Y_1 must overlap but this is impossible. If one tries to build a solution, one quickly runs into an infinite repetition caused by the overlap.



$X_0/X_1 {\sim} Y_0/Y_1$

4.1 Well-Nested Parallelism Relations

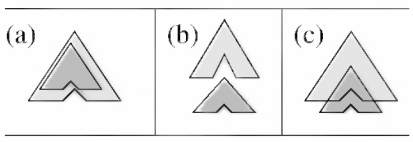


Figure 3: (a) inside, (b) outside, (c) overlap

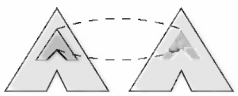
The idea behind well-nested parallelism constraints is to exclude overlap between all parallel segments in a solution.

Definition 4.1 (inside, outside, overlap). *Let* α, β *be segments of a tree* τ, $\alpha = u_0/u_1$, $\beta = v_0/v_1$. *Then* inside(α, β) *holds in* τ *iff*

- *either* $v_0 {\lhd^*} u_0 {\lhd^*} u_1 {\lhd^*} v_1$,
- *or* $v_0 {\lhd^*} u_0 \perp v_1$.

outside(α, β) *holds in* τ *iff* $\mathsf{b}^-_\tau(\alpha) \cap \mathsf{b}^-_\tau(\beta) = \emptyset$. *Otherwise,* overlap$(\alpha, \beta)$ *holds in* τ.

The *image* of a segment is its copy within another segment, as illustrated to the right:



Definition 4.2 (Image). *Let* $c : \alpha \to \beta$ *be a correspondence function and let* $\gamma = u/v$ *be inside* α. *Then* $c(\gamma) = c(u)/c(v)$ *is the* image *of* γ *under* c.

We have to prohibit overlap between images as well as "original" segments.

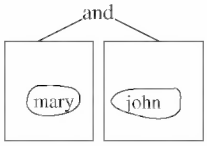
Definition 4.3 (Image closure). *A parallelism relation $\sim$ is* image-closed *if for all correspondence functions c relating segments $\alpha \sim \beta$, and all γ inside α: $\gamma \sim c(\gamma)$.*

Definition 4.4 (Well-nested Models). *Let $\sim$ be an image-closed parallelism relation in the tree τ. Then $(\tau, \sim)$ is a* well-nested *model iff for all segments $\alpha \sim \beta$ either* inside(α, β) *or* inside(β, α) *or* outside(α, β) *holds in τ.*

Definition 4.5 (Well-nested Constraints). *A parallelism constraint is well-nested if it is unsatisfiable or permits a well-nested model.*

4.2 Application to Ellipsis

Well-nesting seems to be a sufficiently weak condition to ensure that we can still model ellipsis. We now show a few examples. In Fig. 1 (b), the two segments involved do not overlap, in fact, they have to lie in disjoint positions in any tree that matches the description. If we outline segments as boxes, the situation of Fig. 1 (b) can be sketched as the picture to the right.



In a similar way, the following elliptical sentences can be modeled with CLLS constraints in which segment terms are properly nested:

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

(3) Mary can't go to Princeton in the fall, but she can in the spring, although if she does, those that expect her in fall will be very disappointed. (Sag, 1976)

Sentence (2) is a famous many-pronouns puzzle. Figure 4 (a) shows a sketch of the two parallelisms that model the two ellipses. Both segments of the first parallelism are nested in the same segment of the second. The situation for sentence (3) is sketched in Fig. 4 (b). The right segment of the first parallelism is nested in the left segment of the second parallelism. So in both cases, the parallelism segments are either nonoverlapping or properly nested.

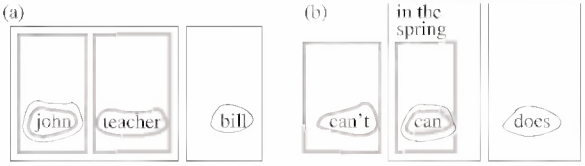


Figure 4: Nesting sketches for (2) and (3)

These examples are typical of the constellations we found. It seems that many cases of ellipsis can be modeled without overlapping parallelism. Corrections may be problematic, although we have not yet managed to construct a definitive counterexample.

5 Solved Forms

In this and the following section, we describe our algorithm for well-nested parallelism constraints. It makes a constraint *dominance-solved*, then solves one parallelism literal, then makes the constraint dominance-solved again, etc. In the current section we define the *dominance solved forms* that all dominance constraint solvers compute, and the *well-nested solved forms* that will be constructed by our solver.

Dominance solved forms and constraint graphs. In Sec. 2 we have introduced constraint graphs informally. We now make this notion formal. The *graph* $G(P)$ of a dominance constraint P is a directed graph $(V(P), \triangleleft^* \uplus \triangleleft_1 \uplus \triangleleft_2 \uplus \ldots)$. Its nodes are the variables of P, and it has two kinds of directed edges:

$$\begin{array}{lll} (X,Y) \in \triangleleft^* & \text{iff} & X \triangleleft^* Y \in P \\ (X,Y) \in \triangleleft_i & \text{iff} & X{:}f(\ldots,Y,\ldots) \in P, \\ & & Y \text{ i-th child of } X \end{array}$$

We draw dominance edges $(X,Y) \in \triangleleft^*$ by dashed lines and children edges $(X,Y) \in \triangleleft_i$ by solid lines. (We leave out node labels as they are not essential here.) We write $P \vdash X \triangleleft^* Y$ if there exists a directed path from X to Y in the graph $G(P)$.

A *dominance solved form* is a dominance constraint P with the following properties for all $X, Y \in V(P)$:

1. The constraint graph $G(P)$ is a tree (no two incoming edges, acyclic, exactly one root).
2. No variable is labeled twice in P.

3. Labeled variables in P don't have outgoing dominance edges in the graph $G(P)$.
4. If $X \perp Y \in P$ then neither $P \vdash X \triangleleft^* Y$ nor $P \vdash Y \triangleleft^* X$.
5. Not $X \neq X \in P$ and not $X = Y \in P$.

Proposition 5.1. *A dominance solved form is satisfiable.*

Segment relations. Fig. 5 defines the possible relationships between two tree segments. The formula $\mathrm{seg}(A)$ that we use there states that the segment term $A = X/X'$ denotes a segment:

$$\mathrm{seg}(A) =_{\mathrm{df}} X \triangleleft^* X'$$

The inside and outside relations are nonproper so that the formulas $\mathrm{inside}(A,B) \wedge \mathrm{inside}(B,A)$ and $\mathrm{inside}(A,B) \wedge \mathrm{outside}(A,B)$ remain satisfiable. In the first case, $\mathrm{equal}(A,B)$ follows, in the second case A must denote the empty segment. The overlap relation, however, is proper:

$$\mathsf{inside}\,(A,B) \models \neg\mathrm{overlap}(A,B)$$

We also use "inside" and "outside" to describe the relation between a segment term and a variable:

$$\mathrm{inside}(Z,A) =_{\mathrm{df}} \mathrm{inside}(Z/Z,A)$$
$$\mathrm{outside}(Z,A) =_{\mathrm{df}} \mathrm{outside}(Z/Z,A)$$

Predecision. In a *predecided* constraint, the relative positions of segment terms are decided. (A dominance-solved form need not be predecided.) A constraint P is *predecided* if any two segment terms A, B in P satisfy the following conditions:

D1 Different segment terms denote different segments: $P \models \neg\mathrm{equal}(A,B)$ if $A \neq B$.

D2 Segment inclusion is decided: $P \models \mathrm{inside}(A,B)$ or $P \models \neg\mathrm{inside}(A,B)$.

D3 No overlap: $P \models \neg\mathrm{overlap}(A,B)$.

D4 Variable inclusion is decided: For all $Z \in V(P)$, $P \models \mathrm{inside}(Z,A)$ or $P \models \neg\mathrm{inside}(Z,A)$.

D5 Equality to holes is decided: For $A = X/X'$ and all Z in $V(P)$, $P \models Z \neq X'$ or $P \models Z = X'$.

Proposition 5.2. *Every well-nested parallelism constraint is satisfaction equivalent to a finite disjunction of predecided constraints.*

Blank segment terms. If for a parallelism literal $A \sim B$, the segment term B is *blank*, i.e. contains no information, then it is easy to read off the solutions of this parallelism literal. We call a segment term $B = X/Y$ *blank* in P if it fulfills three conditions:

B1 Variables $Z \in V(P) - V(B)$ cannot take values inside B, i.e., $P \models \neg\mathrm{inside}(Z,B)$.

B2 B is a segment term X/Y with distinct variables and $X \triangleleft^* Y$ is the only literal of the dominance part of P containing X and Y.

B3 No literal $X{:}f(\ldots)$ or $Z{:}f(\ldots,Y,\ldots)$ belongs to P for any f and Z.

Nesting graphs. In a predecided parallelism constraint, we can study the *nesting* of segment terms: The *nesting graph* $N(P)$ of a constraint P is a directed graph whose nodes are the segment terms of P. The edges of $N(P)$ are given by the relation $<$ that we define recursively:

$$\begin{array}{lll} A < B & \text{if} & P \models \mathrm{inside}(A,B) \wedge \neg\mathrm{equal}(A,B) \\ & \text{or} & A < B' \text{ and } B' \sim B \in P \\ & \text{or} & A' < B \text{ and } A' \sim A \in P \end{array}$$

Proposition 5.3. *If P is satisfiable then the nesting graph $N(P)$ is acyclic.*

Proof. Let $(\tau, \sim, \sigma) \models P$ be a solution of P. If $A < B$ holds in $N(P)$ then the inner $\mathsf{b}_\tau(\sigma(A))$ has properly less nodes than $\mathsf{b}_\tau(\sigma(B))$. So if there existed a cycle $A < \ldots < A$ in $N(P)$ then $\mathsf{b}_\tau(\sigma(A))$ would contain strictly less nodes than itself. □

The segment term A is *outermost* in P if A has no outgoing edges in the nesting graph $N(P)$.

Well-nested solved forms. Now we have all the notation we need to define *well-nested solved forms*, constraints from which a well-nested solution can be directly read off. We call P a *well-nested solved form* iff:

S1 The dominance part of P is satisfiable.

S2 P is predecided.

$$
\begin{array}{rcl}
\text{inside}(A,B) & =_{\text{df}} & \text{seg}(A) \wedge \text{seg}(B) \wedge Y \lhd^* X \wedge (X' \lhd^* Y' \vee X \perp Y') \\
\text{outside}(A,B) & =_{\text{df}} & \text{seg}(A) \wedge \text{seg}(B) \wedge Y' \lhd^* X \vee X' \lhd^* Y \vee X \perp Y \\
\text{equal}(A,B) & =_{\text{df}} & \text{seg}(A) \wedge \text{seg}(B) \wedge X{=}Y \wedge X'{=}Y' \\
\text{overlap}(A,B) & =_{\text{df}} & \text{seg}(A) \wedge \text{seg}(B) \wedge (X \lhd^+ Y \lhd^+ X' \lhd^+ Y' \vee Y \lhd^+ X \lhd^+ Y' \lhd^+ X' \vee \\
 & & \quad X \lhd^* Y \lhd^* X' \perp Y' \quad \vee Y \lhd^* X \lhd^* Y' \perp X')
\end{array}
$$

Figure 5: Segment relations where $A = X/X'$ and $B = Y/Y'$

$\text{cap}(P, B, A) =$
  % invariant: $P \wedge A{\sim}B$ is predecided
  % cut
  let $P_1 = P - \text{cut}(B, P) - \text{para}(P)$
  let $P_2 = P_1 \wedge X \lhd^* Y$ where $X/Y = B$
  % paste
  let $r : V(B, P) \to V$ be some variable renaming
   with $r(B) = A$ and $r(Z)$ fresh for all $Z \notin V(B)$
  let $P_3 = P_2 \wedge r(\text{cut}(B, P)) \wedge \text{s}(r)(\text{para}(P))$
  return $\text{predecide}(P_3)$

Figure 6: Cut and paste simplification

S3 The nesting graph $N(P)$ is acyclic.

S4 If $P = P' \wedge A{\sim}B$ then B is blank in P'.

Proposition 5.4. *Every well-nested solved form has a well-nested solution.*

6 Constraint Solving

In this section we present a constraint solver for well-nested parallelism constraints: Given a parallelism constraint P, it computes a finite set of well-nested solved forms with the same well-nested solutions as P.

Dominance constraint solving and predecision. To compute predecide(P):

- first compute dominance solved forms of P.
- In each dominance solved form P', guess relative positions of variables with respect to the roots and holes of segment terms, unless they are implied by P' already. Discard P' if it contains overlapping segments. Substitute variables if necessary to fulfill condition D1.
- Again compute dominance solved forms to detect inconsistencies.

Cut-and-paste simplification. Given a dominance solved and predecided constraint, we apply *cut-and-paste* to an outermost parallelism literal. The goal is to make one segment term blank.

We need some notation. Given a constraint P with segment B let $V(P, B)$ be the set of variables of B that must take their value inside B.

$$V(P, B) = \{Z \mid P \models \text{inside}(Z, B)\}$$

The constraint $\text{cut}(B, P)$ consists of all literals of P with variables in $V(P, B)$, with the exception of constant labelings of the hole of B:

$$
\begin{array}{rcl}
\text{cut}(B,P) & = & P_{|V(P,B)} - \text{lab(B, P)} \\
\text{lab(X/Y, P)} & = & \{Z{:}a \mid P \models Z{=}Y, a \in \Sigma\}
\end{array}
$$

Let $\text{para}(P)$ be the conjunction of parallelism literals in P. Finally, we lift substitutions $r : V' \to V$ with $V' \subseteq V$ to a substitution $\text{s}(r)$ on segment terms which only alters segment terms with variables solely in V':

$$
\text{s}(r)(C) \quad = \quad \begin{cases} r(C) & \text{if } V(C) \subseteq V' \\ C & \text{else} \end{cases}
$$

The cut-and-paste simplification $\text{cap}(P, B, A)$ is shown in Fig. 6. It requires that $P \wedge A{\sim}B$ is predecided. It first cuts out the contents of B, $\text{cut}(B, P)$, from P and removes all parallelism literals. Then it makes B blank. In P_3, two things happen: First, the contents of B are pasted over those of A. This is done by renaming the variables in $\text{cut}(B, P)$ apart but mapping root and hole of B to those of A. Second, the parallelism literals are adapted by mapping segment terms inside B to segment terms inside A. Finally, the resulting constraint gets dominance-solved and predecided.

Lemma 6.1. *A predecided constraint $P' = P \wedge A{\sim}B$ where A, B are outermost in $N(P')$ has the same models as $A{\sim}B \wedge \bigvee \text{cap}(P, B, A)$.*

```
solve(P) =
  % invariant: P is predecided
  if P contains no parallelism literals then return {P}B.
  elseif N(P) is cyclic then return ∅
  else let P = A~B ∧ P' with A outermost in N(P)
    let S1 = cap(P',B,A) % cut-and-paste
    let S2 = ⋃{ solve(Q) | Q ∈ S1}
    return { Q ∧ A~B | Q ∈ S2 }
```

Figure 7: Constraint solver

This holds because parallel segments have the same structure, so in any model, the segment denoting A contains the structure described by A *and* the structure described by B.

The complete algorithm. The solver for well-nested parallelism constraints is shown in Fig. 6. It applies cut-and-paste simplification exhaustively to parallel segment terms in P, always choosing an outermost parallelism literal next. Constraints with cyclic nesting graphs are discarded as they have no solution.

Proposition 6.2 (Complexity). *The computation of* solve(P) *terminates for all predecided P; emptiness of* solve(P) *can be checked in nondeterministic polynomial time.*

Recursive calls during solve(P) apply to constraints P' with properly fewer parallelism literals than P. All used subroutines terminate, and thus, the computation of solve(P) terminates.

Emptiness of solve(P) can be decided by computing the elements of solve(P) nondeterministically: Whenever solve(P) works with sets of constraints, we choose a single element and continue it alone. The remaining deterministic steps require at most polynomial time.

Proposition 6.3 (Correctness). *If P is predecided then* solve(P) *is a finite set of well-nested solved forms that has the same well-nested models as P.*

The dominance solver and predecision algorithm see to it that solve(P) is predecided and has a satisfiable dominance part. Cutting and pasting leaves the right segment term of a parallelism literal blank, and nothing can move into a blank segment term later because we work from the outside in: B is outermost at the point in time that we solve the literal, and afterwards we only change parts of the constraint that are deeper nested than B. For the same reason, the acyclicity of the nesting graph is guaranteed. Well-nested models are preserved in spite of the changed parallelism literals because well-nestedness presumes image-closedness (Def. 4.3).

Theorem 6.4. *Satisfiability of well-nested parallelism constraints is NP-complete.*

Propositions 6.2 and 6.3 prove satisfiability in nondeterministic polynomial time. NP-hardness already holds for dominance constraints (Koller et al., 2001) which are clearly well-nested.

7 An Example

We demonstrate the algorithm on sentence (2), and we also show how ellipsis resolution and anaphora resolution may be integrated. Figure 8 shows the constraint for that sentence. The coreference is represented by the arrow from "ana" to "john". We have abbreviated "revise" and "the teacher" for better readability.

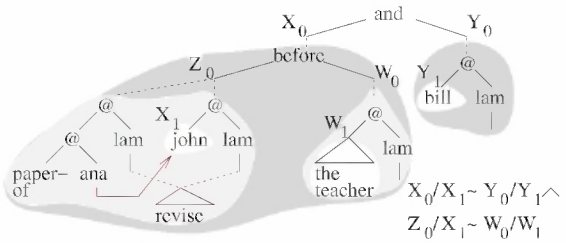


Figure 8: "John revised his paper before the teacher did, and so did Bill."

Constructing a dominance-solved form includes resolving the scope of "john" and "his paper". We pursue the case where "john" takes wide scope. The resulting constraint is already predecided: It entails that the segment terms do not overlap, and it is clear for all variables whether they are inside the segment terms or outside. This is typical for constraints from the linguistic application. So although the problem is NP-hard in theory, in practice it is not necessary to guess relative positions.

We first resolve the ellipses, ignoring the anaphora. We start by solving the outer parallelism $X_0/X_1 \sim Y_0/Y_1$ by "cut-and-paste". The result is shown in Fig. 9 (a). For better readability we have abbreviated "his paper" to "ana".

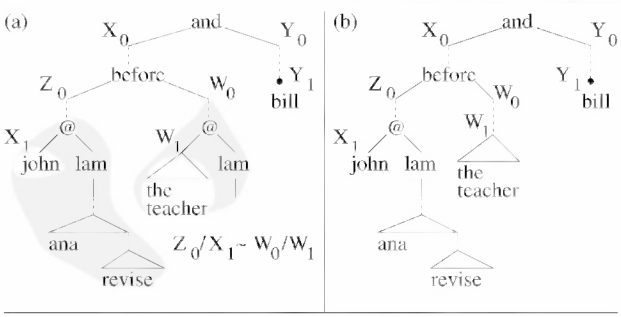
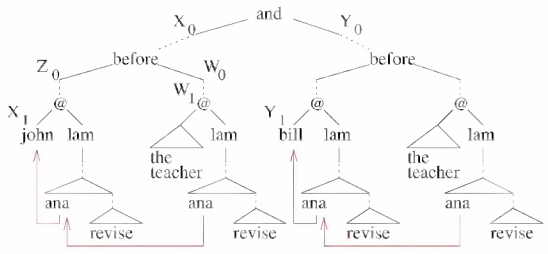


Figure 9: After solving (a) the outer parallelism, (b) the inner parallelism

Now $Z_0/X_1 \sim W_0/W_1$ is outermost. The result of applying "cut-and-paste" to it is shown in Fig. 9 (b). As no parallelism literals are left, solve(P) is this constraint plus $X_0/X_1 \sim Y_0/Y_1 \wedge Z_0/X_1 \sim W_0/W_1$, a well-nested solved form.

To read off a solution from the well-nested solved form, we take each parallelism literal and copy the contents of the left segment term to the right, this time working from the inside out. Finally we enumerate the anaphora readings, using the CLLS rules for the interaction of parallelism and anaphoric links. Figure 10 shows one of the 5 readings (Egg et al., 2001) that this yields.



John revised John's paper before the teacher revised John's paper, and Bill revised Bill's paper before the teacher revised Bill's paper.

Figure 10: Reading off the results

8 Conclusion and Outlook

We have introduced *well-nested parallelism constraints*, a fragment of CLLS for which satisfiability is decidable in nondeterministic polynomial time. We have presented an algorithm for computing well-nested solved forms, and we have shown how well-nested parallelism constraints can be used to model ellipsis.

An interesting question to pursue is whether we can use an even less expressive fragment of parallelism constraints to model ellipsis.

References

E. Althaus, D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. 2002. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*. To appear.

J. Bos. 1996. Predicate logic unplugged. In *Proc. of the 10th Amsterdam Colloquium*.

H. Comon. 1992. Completion of rewrite systems with membership constraints. In *Proc. of ICALP '92*.

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

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

Katrin Erk and Alexander Koller. 2001. VP ellipsis by tree surgery. In *Proc. of the 13th Amsterdam Colloquium*.

K. Erk, A. Koller, and J. Niehren. 2002. Processing underspecified semantic representations in the Constraint Language for Lambda Structures. *Journal of Language and Computation*. To appear.

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

D. Hardt. 1993. *Verb Phrase Ellipsis: Form, Meaning, and Processing*. Ph.D. thesis, University of Pennsylvania.

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

A. Koller, J. Niehren, and R.Treinen. 2001. Dominance constraints: Algorithms and complexity. In *Proc. of LACL'01*.

S. Lappin and H. Shih. 1996. A generalized reconstruction algorithm for ellipsis resolution. In *Proc. of COLING'96*.

G. S. Makanin. 1977. The problem of solvability of equations in a free semigroup. *Mat. Sbornik.*, 103(2):147–236.

M. P. Marcus, D. Hindle, and M. M. Fleck. 1983. D-theory: Talking about talking about trees. In *Proc. ACL'83*.

M. Pinkal. 1995. Radical underspecification. In *Proc. of the 10th Amsterdam Colloquium*. University of Amsterdam.

U. Reyle. 1993. Dealing with ambiguities by underspecification: Construction, representation, and deduction. *Journal of Semantics*, 10(2).

I. Sag. 1976. *Deletion and logical form*. Ph.D. thesis, MIT, Cambridge.

E. Williams. 1977. Discourse and logical form. *Linguistic Inquiry*, 8(1):101–139.