Bridging the Gap Between Underspecification Formalisms: Hole Semantics as Dominance Constraints

Alexander Koller koller@coli.uni-sb.de
Joachim Niehren niehren@ps.uni-sb.de
Stefan Thater stth@coli.uni-sb.de
Saarland University, Saarbrücken, Germany

Abstract

We define a back-and-forth translation between Hole Semantics and dominance constraints, two formalisms used in underspecified semantics. There are fundamental differences between the two, but we show that they disappear on practically useful descriptions. Our encoding bridges a gap between two underspecification formalisms, and speeds up the processing of Hole Semantics.

1 Introduction

In the past few years there has been considerable activity in the development of formalisms for *underspecified semantics* (Alshawi and Crouch, 1992; Reyle, 1993; Bos, 1996; Copestake et al., 1999; Egg et al., 2001). These approaches all aim at controlling the combinatorial explosion of readings of sentences with multiple ambiguities. The common idea is to delay the enumeration of all readings for as long as possible. Instead, they work with a compact *underspecified representation* for as long as possible, only enumerating readings from this representation by need.

At first glance, many of these formalisms seem to be very similar to each other. Now the question arises how deep this similarity is – are all underspecification formalisms basically the same? This paper answers this question for Hole Semantics and normal dominance constraints, two logical formalisms used in scope underspecification, by defining a back-and-forth translation between the two. Due to fundamental differences in the way the two formalisms interpret underspecified descriptions, this encoding is only correct in a nonstandard sense. However, we identify a class of *chain-connected* underspecified representations for which these differences disappear, and the encoding becomes correct. We conjecture that all linguistically useful descriptions are chain-connected. To support this claim, we prove that all descriptions generated by a nontrivial grammar we define are indeed chain-connected.

Our results are interesting because it is the first time in the literature that two practically relevant underspecification formalisms are formally related to each other. In addition, the satisfiability problems of Hole Semantics and normal dominance constraints coincide on their chain-connected fragments. This means that satisfiability of Hole Semantics, which is NP-complete in general (Althaus et al., 2003), becomes polynomial in practice, and can be checked using the efficient algorithms available for normal dominance constraints (Erk et al., 2002). Enumeration of readings becomes much more efficient accordingly.

2 Some Intuitions

The similarity of Hole Semantics and dominance constraints is illustrated in Fig. 1. The pictures graphically represent the underspecified representations of all five readings of the sentence "Every researcher of a company saw a sample" in Hole Semantics (Bos, 1996) and as a dominance constraint (Egg et al., 2001). The underspecified representations specify the material that every reading is made up of and constraints on the way in which they can be combined in obviously similar ways.

However, the interpretations of these underspecified representations differ. In Hole Semantics, the interpretation is given by means of *pluggings*, where holes (the h_i) and labels (l_k) are identified. In contrast, dominance constraints are interpreted by *embedding* descriptions into trees that may contain more material. This difference comes

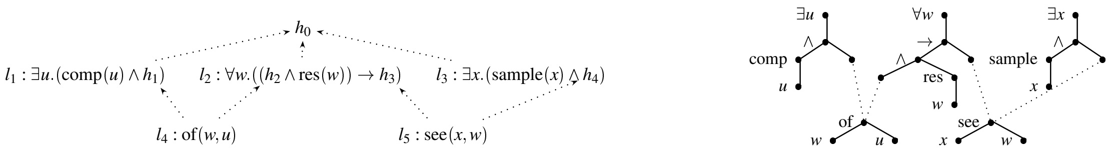


Figure 1: Graphical representations of the Hole Semantics USR (left) and the normal dominance constraint (right) for the sentence "Every researcher of a company saw a sample."

out especially clearly in a description like in Fig. 2. It has no plugging in Hole Semantics, as two different things would have to be plugged into one hole, but it is satisfiable as a dominance constraint. It is this fundamental difference that restricts our result in §5, and that we avoid by using chain-connected descriptions.

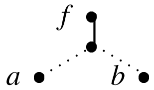


Figure 2: A description on which Hole Semantics and dominance constraints disagree.

3 Dominance Constraints

Dominance constraints are a general framework for the partial description of trees. They have been used in various areas of computational linguistics (Rogers and Vijay-Shanker, 1994; Gardent and Webber, 1998). For underspecified semantics, we consider semantic representations like higher-order formulas as trees.

Dominance constraints can be extended to CLLS (Egg et al., 2001), which adds parallelism constraints to model ellipsis and binding constraints to account for variable binding without using variable names. We do not use these extensions here, for simplicity, but all results remain true if we allow binding constraints.

3.1 Syntax and Semantics

We assume a signature Σ of function symbols ranged over by f, g, each of which is equipped with an arity $\mathrm{ar}(f) \geq 0$, and an infinite set Vars of variables ranged over by X, Y, Z. A dominance constraint φ is a conjunction of dominance, inequality, and labeling literals of the following form:

$$\varphi ::= X \triangleleft^* Y \mid X \neq Y \mid X{:}f(X_1, \ldots, X_n) \mid \varphi \wedge \varphi',$$

where $\mathrm{ar}(f) = n$.

Dominance constraints are interpreted over finite constructor trees, and their variables denote nodes of a tree. We define an *unlabeled tree* to be a finite directed acyclic graph (V, E), where V is the set of nodes and $E \subseteq V \times V$ the set of edges. The indegree of each node is at most 1. Each tree has exactly one node (the *root*) with indegree 0. Nodes with outdegree 0 are called the leaves of the tree.

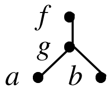
A finite *constructor tree* τ is a triple (T, L_V, L_E) consisting of an unlabeled tree $T = (V, E)$, a node labeling $L_V : V \to \Sigma$, and an edge labeling $L_E : E \to \mathbb{N}$, s. t. for each node $u \in V$ there is an edge $(u, v) \in E$ with $L_E((u, v)) = k$ iff $1 \leq k \leq \mathrm{ar}(L_V(u))$.

Now we are ready to define tree structures, the models of dominance constraints:

Definition 1 (Tree Structure). The *tree structure* $\mathcal{M}^\tau$ of a constructor tree $\tau = ((V, E), L_V, L_E)$ is a first-order structure with domain V interpreting dominance and labeling.

Let $u, v, v_1, \ldots v_n \in V$. The dominance relationship $u \triangleleft^{*\tau} v$ holds iff there is a path from u to v in E and the labeling relationship $u{:}f^\tau(v_1, \ldots, v_n)$ holds iff u is labeled by the n-ary symbol f and has the children $v_1, \ldots, v_n$ in this order; that is, $L_V(u) = f$, $\mathrm{ar}(f) = n$, $\{(u, v_1), \ldots, (u, v_n)\} \subseteq E$, and $L_E((u, v_i)) = i$ for all $1 \leq i \leq n$.

Let φ be a dominance constraint and $\mathsf{Var}(\varphi)$ be the set of variables of φ. A pair of a tree structure $\mathcal{M}^\tau$ and a *variable assignment* $\alpha : \mathsf{Var}(\varphi) \to V_\tau$ *satisfies* φ iff it satisfies each literal in the obvious way. We say that $(\mathcal{M}^\tau, \alpha)$ is a *solution* of φ in this case; φ is *satisfiable* if it has a solution. *Entailment* $\varphi \models \varphi'$ holds between two constraints iff every solution of φ is also a solution of φ'.



We often represent dominance constraints as (directed) *constraint graphs*; for example, the graph in Fig. 2 stands for the constraint $X{:}f(Y) \wedge Y \triangleleft^* Z \wedge Y \triangleleft^* Z' \wedge Z{:}a \wedge Z'{:}b$. This constraint is satisfied e.g. by the tree structure displayed here. Note the added g.

3.2 Solving Dominance Constraints

The satisfiability problem of dominance constraints (i.e. deciding whether a constraint has a solution) is NP-complete (Erk et al., 2002). However, Althaus et al. (2003) show that satisfiability becomes polynomial if the constraint φ is *normal*, i.e. satisfies the following very natural conditions:

(N1) Every variable occurs in a labeling constraint.

(N2) Every variable occurs at most once on the right-hand side and at most once on the left-hand side of a labeling constraint. Variables that don't occur on a left-hand side are called *holes*; variables that don't occur on a right-hand side are called *roots*.

(N3) If $X \triangleleft^* Y$ occurs in φ, X is a hole and Y is a root.

(N4) If X and Y are different variables that are not holes, there is a constraint $X \neq Y$ in φ.

The graph of a normal constraint (e.g. the one in in Fig. 1) consists of solid tree fragments (N1, N2) that are connected by dominance edges (N3); these fragments may not overlap in a solution (N4).

Because every satisfiable dominance constraint φ has an infinite number of solutions, algorithms typically enumerate its *solved forms* instead (Erk et al., 2002). A solved form is a constraint that differs from φ only in its dominance literals. Its graph must be a tree, and the reachability relation on the graph must include the reachability in the graph of φ. Every solved form of φ has a solution, and every solution of φ satisfies one of its solved forms; so we can see solved forms as representing classes of solutions that only differ in irrelevant details (e.g. unnecessary extra material).

Another way to avoid infinite solutions sets is to consider *constructive solutions* only. A solution $(\mathcal{M}, \alpha)$ of a constraint φ is constructive if every node in $\mathcal{M}$ is denoted by a variable in $\mathsf{Var}(\varphi)$ on the left-hand side of a labeling constraint. Intuitively, this means that the solution consists only of the material mentioned in the labeling constraints. Not all solutions are constructive; for example, Fig. 2 is a solved form but has no constructive solutions. The problem of deciding whether a normal dominance constraint does have constructive solutions is again NP-complete (Althaus et al., 2003).

4 Hole Semantics

Hole Semantics (Bos, 1996) is a framework that defines underspecified representations over arbitrary object languages. We use the version of (Bos, 2002) because it repairs some severe flaws in the original definition of admissible pluggings.

Hole Semantics configures formulas of an object language (such as FOL or DRT) with *holes*, into which other formulas can be plugged. Formally, a formula with n holes is a complex function symbol of arity n as above. The equivalent of a dominance constraint is an *underspecified representations* (USR). An USR U consists of a finite set L_U of labeled formulas $l{:}F(h_1, \ldots, h_n)$, where l is a label and F is an object-language formula with holes $h_1, \ldots, h_n$, and a finite set C_U of constraints. Constraints are of the form $l \leq h$, where l is a label and h a hole; this constraint means that h *outscopes* l. Like for dominance constraints, there is a natural way of writing USRs as graphs (Fig. 1).

An USR U is called *proper* if it has the following properties:

(P1) U has a unique *top element*, from which all other nodes in the graph can be reached.

(P2) The graph of U is acyclic.

(P3) Every label and every hole except for the top hole occurs exactly once in L_U.[1]

For example, the USR shown in Fig. 1 is proper; its top element is h_0.

The solutions of underspecified representations are called *admissible pluggings*. A plugging is a bijection from the holes to the labels of an USR. Intuitively, we "plug" every hole with a formula (named by its label), and a plugging is admissible if it respects the constraints on the order of holes and labels.

Definition 2 (**P-domination**)**.** Let k, k' be holes or labels of some underspecified representation U, and P a plugging on U. Then k *P-dominates* k' iff one of the following conditions holds:

1. $k : F \in L_U$ and k' occurs in F, or
2. $P(k) = k'$ if k is a hole, or
3. There is a hole or label k'' such that k P-dominates k'' and k'' P-dominates k'.

[1] The restriction on hole occurrences is missing in (Bos, 2002), but is necessary to rule out counterintuitive USRs.

Definition 3 (Admissible Plugging). A plugging P is *admissible* for a proper USR U iff $k \leq k' \in C_U$ implies that k' P-dominates k.

5 Hole Semantics as Dominance Constraints

Now we have the formal machinery to make the intuitive similarity between Hole Semantics and dominance constraints described in Section 2 precise. We shall define encodings from Hole Semantics to normal dominance constraints and back, and show that this preserves models in a certain sense.

To keep things simple, the results in this sections will only speak about *compact* normal dominance constraints. A dominance constraint is compact if no variable occurs in two different labeling constraints. A very nice property (which we need below) of compact normal constraints is that every variable is either a root or a hole. However, any normal constraint can be made compact by an operation called *compactification*, which compresses conjunctions of labeling constraints into single labeling constraints with more complex labels. So the encodings and results are more more generally correct for arbitrary normal dominance constraints (with acyclic graphs).

From Hole Semantics to Dominance Constraints. Assume $U = (L_U, C_U)$ is a proper USR. To obtain a compact dominance constraint φ_U that encodes the same information, we first encode every labeled formula $l{:}F(h_1, \ldots, h_n)$ as the labeling constraint $l{:}F(h_1, \ldots, h_n)$. We encode every constraint $l \leq h$ in C_U as a dominance constraint $h \triangleleft^* l$ – except if h is the unique top hole and does not occur as a hole in a labeled formula. Finally, we add a constraint $l \neq l'$ for every label l.

This encoding maps labels and holes to variables; labels end up as roots, and holes become holes. This means φ_U satisfies axiom (N3). (N2) follows from (P3). (N1) and (N4) are obvious from the construction. Hence φ_U is normal.

From Dominance Constraints to Hole Semantics. Assume φ is a compact normal dominance constraint whose graph is acyclic. To obtain a proper USR U_φ encoding the same information, we first split the variables $\mathsf{Var}(\varphi)$ into holes and labels: roots become labels, and holes become holes. Then we encode every labeling constraint $X{:}f(X_1, \ldots, X_n)$ as the labeled formula $X{:}f(X_1, \ldots, X_n)$, and we encode every dominance constraint $X \triangleleft^* Y$ as the constraint $Y \leq X$. Finally, we add a top hole h_0 and a constraint $l \leq h_0$ for every label l in U_φ.

U_φ is a well-defined USR because of (N3). (P1) is obvious: h_0 is the unique top element. The graph is acyclic because the graph of φ is acyclic, so (P2) holds. (P3) holds because every label names at least one formula by construction, and no more than one by (N2).

This back-and-forth encoding has the following property:

Theorem 4. *Compact normal dominance constraints φ with acyclic graphs and proper USRs U can be encoded into each other, in such a way that the pluggings of U and the constructive solutions of φ correspond.*

Proof. We only show that the solutions of an USR U and its encoding φ_U correspond; the other direction is analogous.

Assume first that we have a plugging P of U. We build a tree which satisfies φ_U constructively and has one node for every label l of U. The node label of this node is the formula that l addresses. Starting at the top element, we work our way down the USR; whenever we find a hole h, we continue at the label $P(h)$.

Conversely, assume we have a constructive solution $\mathcal{M}$ of φ. Every node in $\mathcal{M}$ is denoted by a variable. Because holes have no labeling constraints, every hole h must denote the same node as a root $P(h)$. Further, every root that is not the root of the entire tree denotes the child of another root, i.e. denotes the same node as a hole. We obtain an admissible plugging by mapping each hole h to the label $P(h)$ in the USR, and mapping the new top hole h_0 to the label denoting the root of the tree. □

6 From Solved Forms to Constructive Solutions

Theorem 4 establishes a very strong connection between Hole Semantics and normal dominance constraints. But it is not quite what we want: Normal dominance constraints are almost always considered with respect to arbitrary solutions (or solved forms), and not constructive solutions. Constraints such as Fig. 2 are solved forms, but have no constructive solutions. The efficient algorithms available for normal constraints check for solved forms, and aren't necessarily correct for constructive satisfiability.

In this section, we establish that for normal dominance constraints which are *chain-connected* and *leaf-labeled* (to be defined below), satisfiability and constructive satisfiability are equivalent; i.e. such a constraint has a constructive solution if only it is satisfiable. The proof proceeds in three steps: First we show that all solved forms of a normal constraint are *simple* iff the constraint *branches constructively*. Then we show that if a constraint is chain-connected, it branches constructively. Finally, every simple solved form of a leaf-labeled constraint has a constructive solution.

6.1 Constructive Branching

We call a solved form *simple* if its graph has no node with two outgoing dominance edges (i.e. Fig. 2 is *not* simple). This means that we can decide for any two variables how they will be situated in a solution of the solved form. They can either dominate each other in either direction, or they can be *disjoint*. But if they are disjoint, we also know the lowest node that dominates them both, and this *branching point* is necessarily also denoted by a variable on the left-hand side of a labeling constraint.

This motivates the following definition. We locally allow disjunctions of constraints and use an auxiliary constraint, the *disjointness constraint* $X \perp Y$ at O, where O is a set of variables. It is satisfied if X and Y denote disjoint nodes whose branching point is denoted by a member of O.

Definition 5. A normal dominance constraint φ *branches constructively* if for any two variables $X, Y \in \mathsf{Var}(\varphi)$,

$$\varphi \models X \triangleleft^* Y \ \vee\ Y \triangleleft^* X \ \vee\ X \perp Y \text{ at } L(\varphi),$$

where $L(\varphi)$ is the set of variables that occur on the left-hand side of a labeling constraint in φ.

Lemma 6. *Let φ be a normal dominance constraint. φ branches constructively iff all solved forms of φ are simple.*

Proof. Assume first that all solved forms of φ are simple; let $\{\varphi_1, \ldots, \varphi_n\}$ be the set of all solved forms. Now because they are simple solved forms, each φ_i entails the right-hand side of Def. 5. But φ entails the disjunction of all of its solved forms, and hence branches constructively.

Conversely, assume that φ has a non-simple solved form φ'. Then φ' must contain a variable X with two outgoing dominance edges (to Y and Z). But this means that φ' has a solution in which Y and Z are different children of X, and hence their lowest common ancestor is not in $L(\varphi)$. □

6.2 Chain-Connectedness

Constructive branching is a semantic property that can't conveniently be proved for a grammar. We shall now relate it to a more easily checkable property called *chain-connectedness*. We will first define chains, then chain-connectedness, and then prove the relation of the two concepts.

Definition 7 (Fragments). A *fragment* in φ is a nonempty subset $F \subseteq \mathsf{Var}(\varphi)$ that is connected by labeling constraints in φ. We call the fragment *maximal* if it has no proper superset that is also a fragment of φ. Exactly one variable in every fragment is a root; we write $\mathcal{R}(F)$ for this root.

Definition 8 (Chains). Let φ be a normal dominance constraint, and let $F_1, \ldots, F_n$ ($n \geq 1$) be disjoint fragments of φ. $\mathcal{C} = (F_1, \ldots, F_n)$ is called a *chain* of φ iff there is a disjoint partition $\mathcal{O} \cup \mathcal{U} = \{F_1, \ldots, F_n\}$ with the following properties:

1. $\mathcal{O}$ is nonempty.
2. For each $1 \leq i < n$, either
   (a) $F_i \in \mathcal{O}$ and $F_{i+1} \in \mathcal{U}$, and there is a hole $X_{i,r}$ of F_i s.t. $\varphi \models X_{i,r} \triangleleft^* \mathcal{R}(F_{i+1})$; or
   (b) $F_i \in \mathcal{U}$ and $F_{i+1} \in \mathcal{O}$, and there is a hole $X_{i+1,l}$ of F_{i+1} s.t. $\varphi \models X_{i+1,l} \triangleleft^* \mathcal{R}(F_i)$.
3. For $1 < i < n$ s.t. $F_i \in \mathcal{O}$, the holes $X_{i,l}$ and $X_{i,r}$ are different.

$\mathcal{O}$ is called the set of *upper fragments* of the chain, and $\mathcal{U}$ is the set of *lower fragments*. We call all the $X_{i,l}$ and $X_{i,r}$ *connecting holes* of $\mathcal{C}$, and all other holes in any of its fragments *open holes*.

A schematic picture of a chain is shown in Fig. 3. Note that although the definition of a chain involves the rather abstract condition that dominance between to variables is entailed by the con-

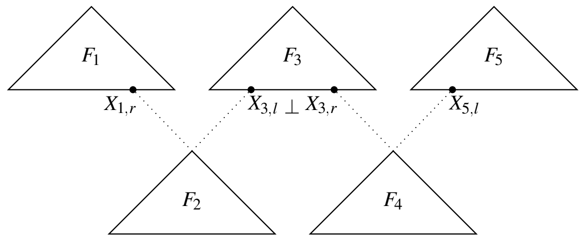


Figure 3: A schematic picture of a chain.

straint, this condition can often be established syntactically – for example in Fig. 3 by the explicit dominance edges. Chains were originally introduced by Koller et al. (2000) because they have very useful structural properties. A particularly useful one is the following.

Lemma 9 (Structural Properties of Chains). *Let φ be a normal dominance constraint, and let $\mathcal{C}$ be a chain that contains all variables of φ. Let $\mathcal{V}_O$ be the set of all variables in upper fragments of $\mathcal{C}$ that are not holes. Then if X, Y are variables in different fragments of $\mathcal{C}$, the following structural property holds:*

$$\varphi \models X \triangleleft^* Y \vee Y \triangleleft^* X \vee X \perp Y \text{ at } \mathcal{V}_O$$

Using this lemma, it is easy to show that whenever a constraint is chain-connected, it also branches constructively.

Definition 10. Two variables X, Y of a normal dominance constraint φ are *chain-connected in* φ iff there is a chain $\mathcal{C}$ in φ that contains both X and Y. A constraint is *chain-connected* iff every pair of variables is chain-connected.

Proposition 11. *Every chain-connected dominance constraint φ branches constructively.*

Proof. Let X, Y be two arbitrary variables in φ. If X and Y belong to the same fragment, there is obviously a connecting chain containing just this fragment. Otherwise, constructive branching for X and Y follows from Lemma 9. □

For the last step of the proof, we define that a normal dominance constraint is *leaf-labeled* if every variable occurs on the left-hand side of a labeling or dominance literal. Such constraints have the following property:

Lemma 12. *Every simple solved form of a leaf-labeled constraint has a constructive solution.*

Putting it all together, we obtain the intended result:

Theorem 13. *Every solved form of a chain-connected, leaf-labeled normal dominance constraint has a constructive solution.*

We can transfer the notions of chain-connectedness and leaf-labeledness to USRs either by a direct definition or by defining that U is chain-connected or leaf-labeled iff φ_U is. Then we can state the following theorem:

Corollary 14 (Processing of Hole Semantics). *The problem whether a chain-connected, leaf-labeled proper USR has a plugging is polynomial.*

Proof. Simply check the corresponding dominance constraint for satisfiability. Althaus et al. (2003) give a quadratic satisfiability algorithm; Thiel (2002) improves this to linear. □

7 Connectedness in a Grammar

Finally, we claim that chain-connectedness and leaf-labeledness are very weak assumptions to make about a normal dominance constraint, and conjecture that all linguistically useful constraints satisfy them. We define a nontrivial grammar for a fragment of English and show that it only generates dominance constraints with these properties. The argument we use to establish chain-connectedness (the less obvious property) is fairly general, and should be applicable to other grammar fragments as well.

The grammar we use is a variant of the one presented in (Egg et al., 2001). Its syntax-semantics interface produces dominance constraints describing formulas of higher-order logic; the symbol @ stands for functional application, and abstraction and variables are written as 'lam_x' and 'var_x'. We use dominance constraints because this gives us the logical tools we need in the proof; but by Theorem 4, we can translate all results back into proper USRs, and those USRs will also be chain-connected.

7.1 The Grammar

The syntactic component of the grammar consists of the following phrase structure rules.

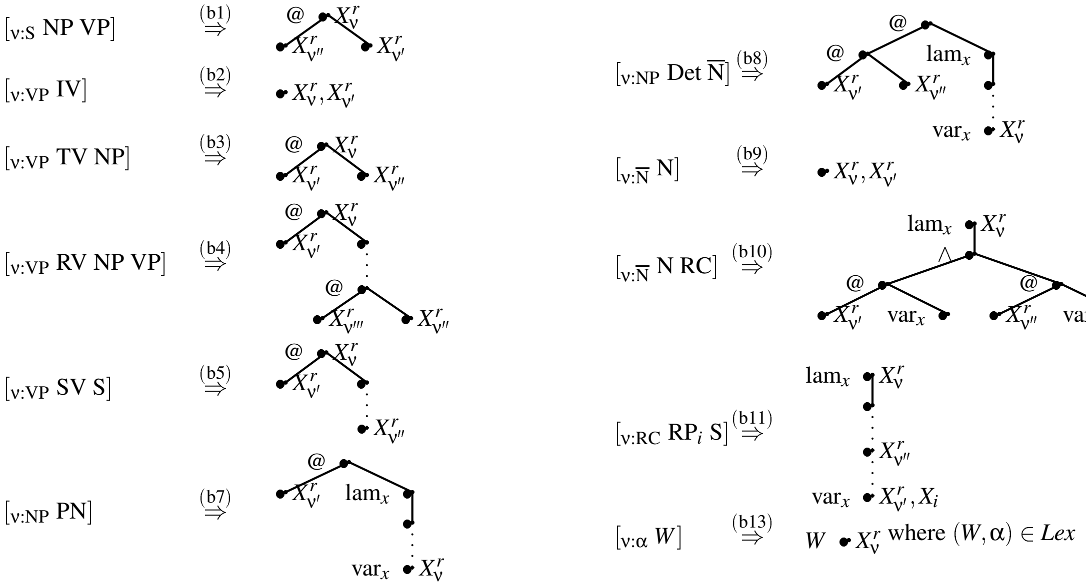


Figure 4: The syntax-semantics interface

(a1)	S $\to$ NP VP	(a8)	NP $\to$ Det $\overline{\text{N}}$
(a2)	VP $\to$ IV	(a9)	$\overline{\text{N}}$ $\to$ N
(a3)	VP $\to$ TV NP	(a10)	$\overline{\text{N}}$ $\to$ N RC
(a4)	VP $\to$ RV NP VP	(a11)	RC $\to$ RP S
(a5)	VP $\to$ SV S	(a13)	$\alpha \to W$
(a7)	NP $\to$ PN		if $(W, \alpha) \in Lex$

Most category labels are self-explanatory, perhaps except for SV, which refers to verbs taking sentence complements such as *say*, and RV, which refers to (object) raising verbs such as *expect*.

The lexicon is defined by a relation *Lex* relating words and lexical categories. Rule (a13) expands lexical categories to words of the category.

7.2 The Syntax-Semantics Interface

Every node ν in a syntax tree contributes a constraint φ_ν; the variable X^r_ν is intuitively the "root" of this contribution. We assume that the syntax provides for a coindexation of relative pronouns and their corresponding traces, and associate each NP with index i with a corresponding variable X_i.

The variables are related by the rules in Fig. 4. Each syntactic production rule corresponds to one semantic construction rule, which defines the semantic contribution of a syntactic node. A construction rule of the form $[_{\nu:P}\ Q\ R] \Rightarrow \varphi_\nu$ means that the node ν in the syntax tree is labeled with P, and its two daughter nodes ν' and ν'' are labeled with Q and R, respectively. The semantic contribution of ν is the constraint φ_ν, with fresh instances of 'lam$_\text{x}$' and 'var$_\text{x}$' where necessary.

The complete constraint of a syntax tree with root ν is the conjunction of the $\varphi_{\nu'}$ for all nodes ν' dominated by ν, and inequalities that are needed to make the constraint normal.[2]

7.3 Connectedness of Constraints

The proof that all constraints generated by this grammar are connected proceeds by structural induction over parse trees. The semantic contributions of leaves are trivially chains, and hence connected. What we show in the rest of this section is that if t is any subtree of the syntax tree, and all the semantics of all immediate subtrees of t are connected, then so is the semantics of t. We ignore the globally introduced inequality constraints because they have no effect on chain-connectedness.

The central property of the construction rules that we exploit is the following:

Proposition 15. *Let $\varphi_0, \ldots, \varphi_n$ be chain-connected constraints such that*

1. $\text{Var}(\varphi_i) \cap \text{Var}(\varphi_j) = \emptyset$*, for* $1 \le i < j \le n$*,*
2. $\text{Var}(\varphi_0) \cap \text{Var}(\varphi_i) = \{X_i\}$*, for* $1 \le i \le n$*,*

where $X_1, \ldots, X_n$ are open holes in all connecting chains in φ_0. Then the constraint $\varphi_0 \wedge \cdots \wedge \varphi_n$ is chain-connected.

[2]The original grammar accounts for scope island constraints by means of additional dominance literals. We ignore them here, as they do not affect chain-connectedness.

This can be proved by induction. The base case $n = 0$ is trivial, and for the induction step we combine a connecting chain within $\varphi_0 \wedge \cdots \wedge \varphi_{n-1}$ from an arbitrary X to X_n with a connecting chain within φ_n from X_n to an arbitrary Y. Chains are combined by taking all the fragments of the two smaller chains together. The assumption that the X_i are open holes in the connecting chains is needed for the problematic case in which the fragment containing X_n is an upper fragment in both chains.

All constraints introduced by a semantics construction rule other than (b11) are of this form: φ_0 corresponds to the constraint introduced by the rule, and $\varphi_1, \ldots, \varphi_n$ to the constraints associated with the daughter nodes. Hence, all constraints generated using only these rules are chain-connected. For the case of (b11), observe that the relative pronoun is coindexed with its trace. This means that the variable $X^r_{\nu'}$ occurs in the same fragment as $X^r_{\nu''}$, so (b11) also satisfies the general scheme. An easier structural induction shows that the constraints are also leaf-labeled. Hence:

Corollary 16. *All constraints generated by the grammar are chain-connected and leaf-labeled.*

8 Conclusion

We have established the equivalence of Hole Semantics and normal acyclic dominance constraints with constructive solutions. They are equivalent to normal acyclic dominance constraints with standard solutions if the constraints are chain-connected and leaf-labeled. All constraints generated by our grammar have these properties; we conjecture this is true more generally.

This bridges a gap between the two underspecification formalisms, which means that we can now combine the simplicity of hole semantics with the efficient algorithms, powerful metatheory, and extensibility of dominance constraints. A first practically useful result is a polynomial satisfiability algorithm for chain-connected, leaf-labeled USRs.

Conversely, chain-connected dominance constraints inherit some of Hole Semantics' resource-sensitivity: Additional material *need* never be added to satisfy the constraint; but to model e.g. reinterpretation (Koller et al., 2000), this is still possible. This resource-sensitivity was the crucial difference between the two formalisms. In the future, it will be interesting to see how our results extend to other resource-sensitive underspecification formalisms – for example, to MRS (Copestake et al., 1999), whose naive encoding into dominance constraints is less obviously normal, and which adds a more powerful outscopes relation.

References

H. Alshawi and R. Crouch. 1992. Monotonic semantic interpretation. In *Proc. 30th ACL*, pages 32–39.

E. Althaus, D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*. In press.

Johan Bos. 1996. Predicate logic unplugged. In *Proc. 10th Amsterdam Colloquium*, pages 133–143.

J. Bos. 2002. *Underspecification and resolution in discourse semantics*. Ph.D. thesis, Saarland University.

A. Copestake, D. Flickinger, and I. Sag. 1999. Minimal Recursion Semantics. An Introduction. Unpublished manuscript.

M. Egg, A. Koller, and J. Niehren. 2001. The constraint language for lambda structures. *Journal of Logic, Language, and Information*, 10:457–485.

K. Erk, A. Koller, and J. Niehren. 2002. Processing underspecified semantic representations in the constraint language for lambda structures. *Research in Language and Computation*, 1(1). In Press.

Claire Gardent and Bonnie Webber. 1998. Describing discourse semantics. In *Proceedings of the 4th TAG+ Workshop*, Philadelphia.

A. Koller, J. Niehren, and K. Striegnitz. 2000. Relaxing underspecified semantic representations for reinterpretation. *Grammars*, 3(2-3).

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

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

Sven Thiel. 2002. A linear time algorithm for the configuration problem of dominance graphs. Submitted.