A Notion of Semantic Coherence for Underspecified Semantic Representation

Mehdi Manshadi[*]
University of Rochester

Daniel Gildea[**]
University of Rochester

James F. Allen[†]
University of Rochester

The general problem of finding satisfying solutions to constraint-based underspecified representations of quantifier scope is NP-complete. Existing frameworks, including Dominance Graphs, Minimal Recursion Semantics, and Hole Semantics, have struggled to balance expressivity and tractability in order to cover real natural language sentences with efficient algorithms. We address this trade-off with a general principle of coherence, which requires that every variable introduced in the domain of discourse must contribute to the overall semantics of the sentence. We show that every underspecified representation meeting this criterion can be efficiently processed, and that our set of representations subsumes all previously identified tractable sets.

1. Introduction

Quantifier scope ambiguity is a big challenge in deep language understanding systems. Consider the following conversation:

> **Woman**: *I believe there is one true soulmate for every person.*
>
> **Man**: *He must be very busy.*[1]

Most people find the man's answer unusual (humorous, sarcastic, etc.). This is because one of the two scopings of the woman's sentence feels so obvious that the less likely

* Department of Computer Science, University of Rochester, Rochester, NY 14627. E-mail: mehdih@cs.rochester.edu.

** Department of Computer Science, University of Rochester, Rochester, NY 14627. E-mail: gildea@cs.rochester.edu.

† Department of Computer Science, University of Rochester, Rochester, NY 14627. E-mail: james@cs.rochester.edu.

1 dilbert.com/strip/2001-04-19.

Submission received: 2 June 2016; revised version received: 13 March 2017; accepted for publication: 1 August 2017.

doi:10.1162/COLI_a_00307

scoping is often missed at first glance. In the most likely interpretation, where there are many soulmates, the quantifier *every* has **wide scope**, and, in the second interpretation, where there is a unique soulmate, *every* has **narrow scope**. The following conversation is of a similar nature:

Bob: *How long have you and Opal been married now Earl?*

Earl: *I've lost track. But I can tell you this ... I don't regret one day of it.*

Bob: *Which day don't you regret?*[2]

The difference, however, is that, in this example, the scope ambiguity is not between two quantifiers, but between a quantifier (*one*) and a scopal operator (*negation*). **Underspecification**, that is, generating an unscoped semantic representation, has been the most common way of dealing with quantifier scope ambiguity since the early days of natural language processing. Underspecification is not adopted only because quantifier scope disambiguation is difficult, but also because, for most practical purposes, an underspecified representation (UR)[3] will do the job. Equations (1) and (2) show an unscoped logical form (LF) for the sentences *There is one soulmate for every person* and *I do not regret one day*, respectively.

$$\begin{array}{l}\langle One\ \ x\ \ Soulmate\rangle \\ \langle Every\ \ y\ \ Person\rangle \\ Of(x,\ y)\end{array} \qquad (1) \qquad\qquad \begin{array}{l}\langle One\ \ x\ \ Day\rangle \\ Not(Regret(I,\ x))\end{array} \qquad (2)$$

Equation (3) shows the two scopings of the unscoped LF in Equation (2):

$$\begin{array}{l} One(x, Day(x), Not(Regret(I, x))) \\ Not(One(x, Day(x), Regret(I, x))) \end{array} \qquad (3)$$

When the number of quantifiers increases, the number of possible scopings will increase exponentially. More recent underspecification formalisms are constraint-based—that is, they allow for constraints, restricting the order of quantifiers, to be added to filter out unwanted scopings. The constraints can come from different sources, including deeper processing steps, such as discourse or pragmatics. Several constraint-based underspecification frameworks have been developed over the past couple of decades. Minimal Recursion Semantics (MRS) (Copestake, Lascarides, and Flickinger 2001), Hole Semantics (Bos 2002), and Dominance Constraints (Koller, Niehren, and Thater 2003) are among those frameworks. Each framework is different in the type of constraints it allows, and has its own advantages and disadvantages. A constraint-based UR defines

2 From *Pickles* (Brian Crane, 2005).

3 In the literature, UR is used to refer to underspecified representation within the context of Hole Semantics. In this article, unless otherwise specified, we use UR as a blanket term for any scope-underspecified representation.

a computational problem that needs to be solved: Given a UR with a set of constraints, one needs to check if these constraints are consistent, that is, whether there is a scoping satisfying all the constraints. This is called the **satisfiability** problem. The satisfiability problem for all these three frameworks, in their general form, is intractable (Althaus et al. 2003). There has been some effort towards defining a notion of **well-formedness** within the context of these frameworks. The goal has been to define a subset of URs, the so-called well-formed UR, for which the satisfiability problem becomes tractable. For example, Niehren and Thater (2003) defined the notion of (weak) net to characterize such a subset. Well-formedness was also intended to bridge the gap between these underspecification formalisms; the hope was that the differences between these formalisms disappear and they become equivalent once restricted to well-formed structures.

As seen in Niehren and Thater (2003) and Fuchss et al. (2004), the problem with those efforts on defining a notion of well-formedness is that their satisfaction of both properties was only empirically supported, and hence the correctness of those statements has remained a conjecture. In better words, first, there was no mathematical proof to show that nets enforce the equivalence of qeq vs. dominance relations, the two different types of constraint used in MRS vs. Dominance Constraints/Hole Semantics, and second, although they were proved to be tractable, there was no convincing linguistic justification as to why nets cover all URs corresponding to coherent sentences. In fact, this claim was later falsified when Thater (2007) presented examples of coherent sentences that were unaccounted for by nets (Section 7.1). In summary, it has remained an open question whether there is a linguistically justified notion of well-formedness that not only (provably) bridges the gap between the above formalisms, but also guarantees tractability. In this article, we propose such a notion of semantic coherence that not only answers both of these open questions but also solves several other unanswered questions within the context of scope underspecification. The contribution of this work can be summarized as follows:

- We extend the previous tractable frameworks to cover those natural language sentences that were known to be unaccounted for without increasing the complexity of the algorithms.
- We go beyond those known unaccounted examples and, once and for all, prove that every semantically coherent natural language sentence (based on a linguistically justified notion of semantic coherence) can be solved in polynomial time, presenting a definitive answer to the open question of whether solving unscoped representations of real-life natural language sentences within the context of these formalisms is tractable.
- We prove that, under our notion of coherence, the two fundamentally different types of constraint, dominance and qeq, become equivalent, hence, the principal difference between these formalisms disappears.
- We further bridge the gap between the constraint-based formalisms by proving that binding constraints (constraints enforcing that quantified variables be bound within the scope of their quantifiers), which are

label-to-label dominance relations in nature, can be represented by hole-to-label dominance relations, as long as the URs are coherent. This explains how a formalism such as Hole Semantics, which does not incorporate label-to-label dominance relations, does not lack the power to model binding constraints.

- Finally, given that quantifier scoping has traditionally been treated as an ordering problem (i.e., predicting a permutation of quantifiers), whereas in the constraint-based formalisms it is defined as predicting a tree structure, our notion of coherence allows us to explain this discrepancy. We show that, for coherent URs, quantifier scoping is reduced from predicting a tree structure to finding a permutation.

Whereas we focus on finding solutions to underspecified representations with hard constraints, our results have implications for statistical systems based on soft constraints. Because weighted soft constraints generalize hard constraints, finding efficient algorithms for solving systems with hard constraints is a first step toward finding efficient algorithms for finding the highest-scoring solution under weighted constraints. We also show how our algorithm for finding solutions under hard constraints can be used to guide search for the highest scoring solution given a combination of hard and soft constraints.

Some of this article's results (or a weaker version of them) have been proved in our own previous work. In Manshadi, Allen, and Swift (2008b), we proved the equivalence of qeq and dominance for canonical form MRS, which motivated the notion of **completeness**. In Manshadi, Allen, and Swift (2009), we introduced the notion of heart-connectedness, which, in the current work, forms the basis of coherence. In another line of work (Manshadi and Allen 2012), we introduced a superset of nets, called supernets, that covered the known examples unaccounted for by nets.

In this article, however, we combine completeness and heart-connectedness to define a mathematically characterized notion of coherence. We show that this notion directly follows from a linguistic property of semantically coherent sentences, as discussed by Frege (1923), among others. Under such a linguistically justified and mathematically characterized notion of coherence, we prove that (i) the given formalisms become equivalent, (ii) coherence guarantees tractability, and (iii) quantifier scoping is reduced to an ordering problem. More importantly, to connect our work to the previous work within the context of Dominance Graph, we define the notion of **hypernet**, the largest tractable subset of Dominance Graph found so far. Hypernets, in particular, contain downward-connected nets (Koller and Thater 2007) as well as all coherent URs.

In order to build a mathematical infrastructure to be able to rigorously prove these properties, in particular, the equivalence of different formalisms within a newly defined notion of well-formedness (i.e., coherence), we define a framework, the **underspecification graph**, by simultaneously incorporating both qeq and dominance constraints. This allows us to compare URs represented within each of the individual formalisms under a universal framework (as shown schematically in Figure 1), and ultimately to formally prove equivalence. The structure of this paper is as follows. In Section 2, we

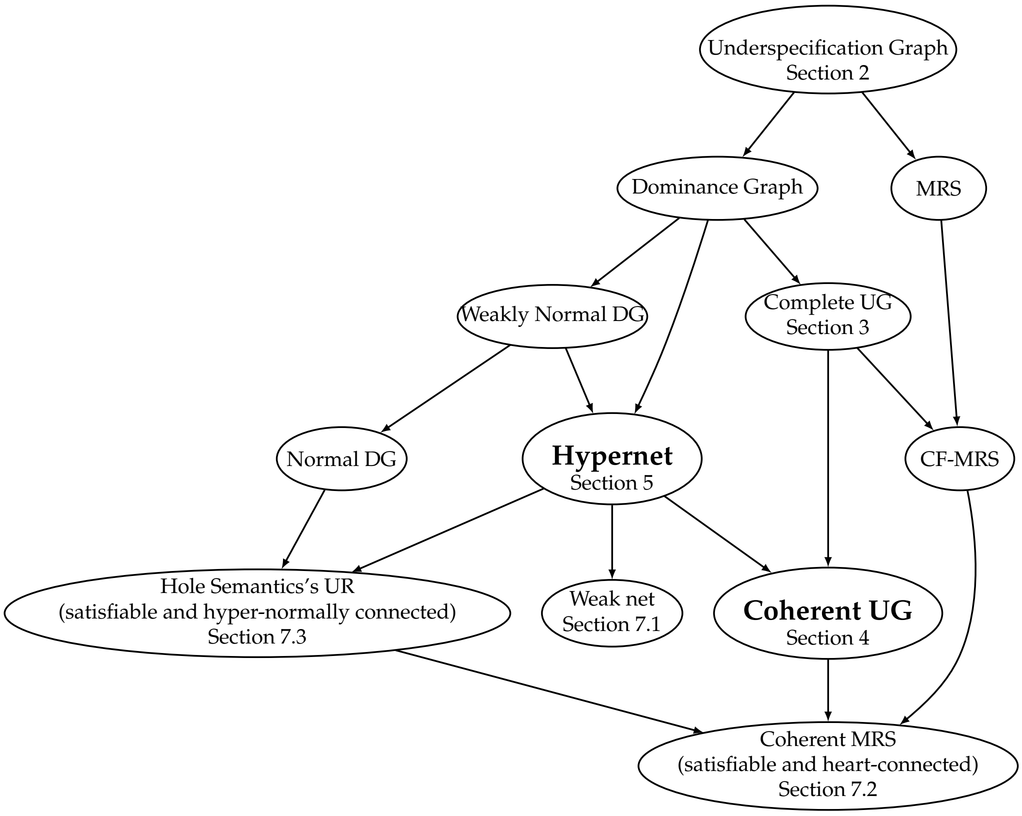


Figure 1
Classes of underspecification graph introduced in this article. We define hypernet, show that it is computationally tractable, and further show that it covers all coherent sentences, and that it subsumes previously identified tractable classes.

give a formal definition of our universal framework (i.e., underspecification graph). Section 3 defines a notion of completeness and proves that, under this notion, the two fundamentally different types of constraint used in underspecification formalisms (qeq and dominance) become equivalent. Section 4 defines a notion of coherence for a complete UR. Section 5 discusses the tractability issue. We propose a tractable subset of URs and prove that it is the largest tractable subset found so far. We then show that every coherent UR belongs to this set, and we discuss implications for systems of mixed hard and soft constraints. Section 6 shows that scope disambiguation can be treated as an ordering problem. Finally, Section 7 gives a detailed comparison of our framework with previous work, and Section 8 concludes.

2. Underspecification Graph

Consider the following example.

1. *Every child of a politician runs.*

Early systems (Schubert and Pelletier 1982; Hobbs and Shieber 1987; Allen 1995) represented the semantics of such a sentence using an unscoped LF of the following general form:

$$Every(x, Child(x,y), \bigcirc),\ A(y, Politician(y), \bigcirc),\ Run(x) \tag{4}$$

To scope this LF, at each step, a quantifier is picked and the main predication (i.e., $Run(x)$) or the partially scoped formula built so far is fused to its body hole:

$$\begin{aligned} &\text{Step 1. } Every(x, Child(x,y), Run(x)) \\ &\text{Step 2. } A(y, Politician(y), Every(x, Child(x,y), Run(x))) \end{aligned} \tag{5}$$

By picking quantifiers in different orders, different scopings are generated.

Next, the notion of constraints was introduced into the domain of scope underspecified semantics. For example, Quasi Logical Form (Alshawi and Crouch 1992) allows for constraints such as $A > Every$ to be used to force one quantifier to rest within the scope of another. By inventing some machinery that allows for Discourse Representation Theory (Kamp 1981) to support scope underspecification, Underspecified Discourse Representation Theory (or UDRT) (Reyle 1993) takes the notion of constraint-based underspecification to a new level. UDRT introduces a complex system of constraints, that, among other things, can define a maximum and a minimum range for the scope of a quantifier relative to other scope bearing elements. It is fair to say that Reyle's work inspired the next two decades of research on constraint-based scope underspecification, resulting in several underspecification formalisms such as Hole Semantics (HS; Bos 1996), MRS (Copestake, Lascarides, and Flickinger 2001), and Dominance Graph (DG; Thater 2007). Unlike Quasi Logical Form or UDRT, the new formalisms treat underspecification as an abstract algebraic framework which is independent of the target object language, whether it is first order predicate calculus, modal logic, Discourse Representation Theory, etc. A UR in these formalisms is a set of abstract labeled formulas with holes that can be filled in with other labeled formulas. The formulas come with a set of constraints between the labels and holes, and in order for a scoping to be considered valid, the set of constraints ought to be respected. Despite all the similarities, as will be seen shortly, HS/DG and MRS differ in how they interpret the constraints.

Figure 2 shows the graphical depiction of the UR of Example (1) as proposed by MRS. Solid nodes represent labeled formulas, and are called **label nodes**. The holes of the formulas are represented by nodes with hollow circles, called the **hole nodes**. Scopings of such a structure are built by fusing label nodes to hole nodes as shown in Figure 2(b). This UR leaves both the body and the restriction of the quantifiers underspecified. This is to allow for scopings such as Figure 2(c), in which quantifier *A* lies between quantifier *Every* and its restriction. The dotted line between the label node *Child(x,y)*, call it *l*, and the restriction hole of *Every*, call it *h*, is an example of a constraint. This constraint (also represented as $h =_q l$), requires that either *h* is directly filled by *l* or *h* is filled by a quantifier and the body hole of that quantifier is filled by *l* (or by another quantifier and the body hole of the latter is filled by *l*, and so on so forth). This type of constraint, first introduced by the MRS framework, is called a **qeq** constraint. It is easy

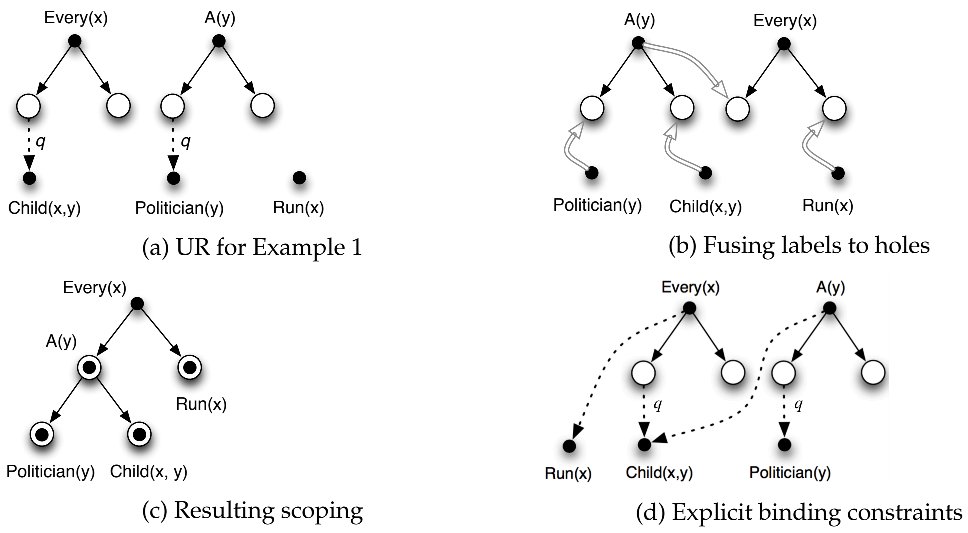


Figure 2
Using graphical notation to represent unscoped LF.

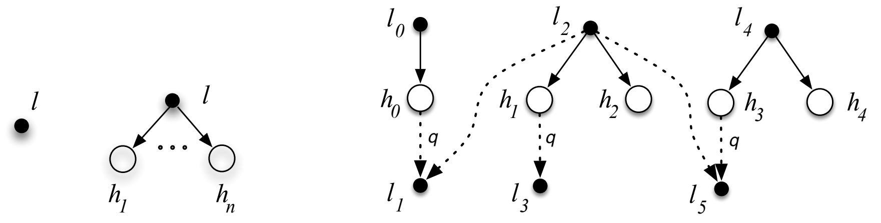
to see that qeq directly implements the idea of wrapping a quantifier around a formula as shown in Equation (5). It is easy to see that not every assignment of labels to holes in the UR is a valid scoping, even if it satisfies both qeq constraints. This is because, in addition to qeq constraints, the UR carries a group of implicit constraints, the so-called binding constraints. The binding constraints force every variable (x, y, etc.) to be in the scope of its quantifier. Unlike qeq, binding constraints enforce a mere outscoping (a.k.a. dominance) relation. That is, to satisfy a binding constraint from u to v, it is enough that u outscopes (a.k.a. dominates) v.

To simplify things, let us make the binding constraints explicit by using unlabeled dotted edges as shown in Figure 2(d). We call the resulting representation that incorporates both dominance and qeq constraints an underspecification graph (UG).

Before moving to the formal definition of UG, it should be emphasized that what is defined here as UG is nothing but the integration of the three existing concepts: MRS, HS, and DG. Defining a framework that subsumes all three has proved very useful in substantiating the results we have obtained and in providing rigorous proofs. Otherwise, from a practical standpoint (as proved later), UG has little to no advantage over DG. In addition, it serves us better to first define the general framework, and then introduce subsets of it, rather than the other way around.

2.1 The Formal Definition

Quantifiers and scopal and non-scopal predications are the building blocks of a UR. As seen in Figure 2, in graphical notation, they are depicted as single solid nodes or trees with solid edges. In DG terminology, these are called **fragments**. We alternatively use the term elementary tree, emphasizing that these are analogous to the MRS's elementary predications (Section 7.2).



(a) General form of ET (b) An example of UG

Figure 3
Elementary tree and underspecification graph.

Definition 1 (Elementary Tree)
An **elementary tree** or **ET** (a.k.a. fragment) is an *ordered*[4] tree of depth 0 or 1. The **roots** of all ETs are represented by small solid circles and are referred to as **label nodes** or simply **labels**. All the leaf nodes of the ETs of depth 1 are represented as big hollow circles and are referred to as **hole nodes** or simply **holes**. ETs with holes are called **scopal ETs**.[5]

Figure 3(a) shows the general form of an elementary tree for both cases of *depth* $= 0$ and *depth* $= 1$. When *depth* $= 0$, the elementary tree is a **singleton**.

Definition 2 (Underspecification Graph)
The 8-tuple $U = \langle L_U, H_U, E_U, Q_U, D_U, T_U, L_U^q, P_U \rangle$ is called a (scope) **Underspecification Graph** or in short **UG**, if it is a finite structure with the following properties:

- $\mathcal{F}_U = \langle L_U, H_U, E_U, P_U \rangle$ is a forest of elementary trees, with L_U being the set of label nodes, H_U the set of hole nodes, and E_U the set of directed solid edges, going from the root of ETs to their holes. The order of holes in each ET is defined by P_U.[6] In graphical notation, P_U is not explicitly given, as it is implicit in the left-to-right order by which the hole nodes of each ET are depicted.

- Q_U is a relation from H_U to L_U, that is, $Q_U \subset H_U \times L_U$. In graphical notation, each $(h, l) \in Q_U$ is represented as a directed *dotted* edge from h to l marked with a label q, called a **qeq constraint**.

4 Throughout this paper, unless otherwise specified, by tree we always mean a rooted ordered tree.
5 A mathematically precise definition requires ETs to be defined as graphs over pairs (u, s), where u is a node and s is a symbol of the value $\mathcal{L}$ (for label nodes) or $\mathcal{H}$ (for hole nodes). Such notation will be quite cumbersome and may become confusing, especially given that we later need to define two types of edges, and even two types of dotted edges. Therefore, although we understand that the underlying mathematical model is defined in this precise way, we avoid adopting the notation. Instead, we use l, l_1, and so forth, to denote labels and h, h_1, and so forth, to denote holes. The same applies to different types of edges defined later.
6 Therefore, P_U is a partial order over H_U that defines, for each ET ϵ, a total order over the set of holes of ϵ.

- D_U is a relation over $H_U \cup L_U$. Each $(u, v) \in D_U$ is called a **dominance constraint** and, in graphical notation, is represented as a directed *dotted* edge from node u to node v. The dominance constraints can go from any node to any node, except from holes to holes, therefore, $D_U \subset (L_U \cup H_U) \times (L_U \cup H_U) - H_U \times H_U$. Dominance (as opposed to qeq) is the default constraint type, therefore, the label d is dropped for the sake of brevity.
- T_U is a dummy ET of depth 1, with $l_0 \in L_U$, $h_0 \in H_U$, and $e_0 = (l_0, h_0) \in E_U$ being its root, its single hole, and its single edge. It is defined to be the designated root of every scoping, therefore, T_U, l_0, and h_0 are called **top ET**, **top label**, and **top hole**, respectively.[7] The set of all labels except l_0 is denoted by $\tilde{L}_U$, that is, $\tilde{L}_U =_{def} L_U - \{l_0\}$.
- $L_U^q \subset \tilde{L}_U$ is the set of **floating scopal nodes**. The ETs rooted at these nodes are called **floating scopal ETs**. Every scopal ET other than the floating scopal ETs is called a **fixed-scopal ET**. Floating scopal ETs are required to have a hole as the right-most child of the root. This hole and its connecting edge (sometimes distinguished with a label "b" for emphasis) are called the body hole, and the body edge, respectively. In practice, floating scopal ETs correspond to (generalized) quantifiers. Therefore, we use the terms *floating-scopal* and *(generalized) quantifier* interchangeably. Quite often, we do not explicitly define L_U^q, because (generalized) quantifiers can be recognized from the context. Because quantifiers have one restriction and one body preposition, throughout this article we only consider floating scopals with two holes.

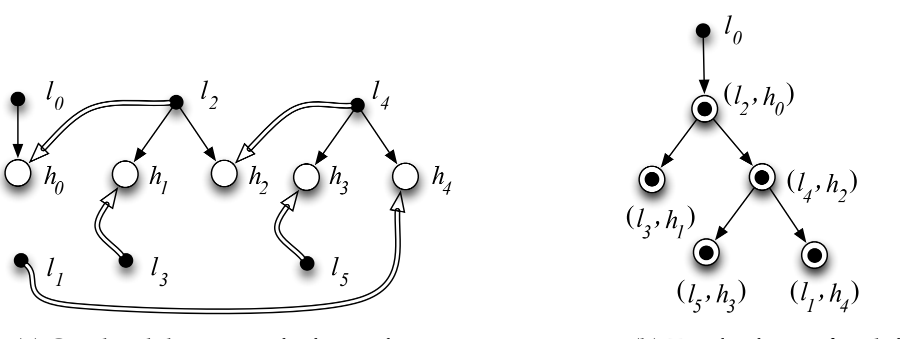
Scopings of a UG are built by fusing labels to holes. We call this a **fusion**.[8]

Definition 3 (Fusion)
Given a UG U, a (total) **fusion** f is a total function from $\tilde{L}_U$ to H_U. A partial **fusion** f is a partial function from $\tilde{L}_U$ to H_U.

Figure 4(a) demonstrates a fusion for the UG U in Figure 3(b). Given a fusion f, the corresponding scoping is denoted by $\mathcal{T}_{U,f}$. We construct the graph of $\mathcal{T}_{U,f}$ from U by removing all the constraint edges and fusing l to $f(l)$ for each $l \in \tilde{L}_U$, as illustrated in

7 Most underspecification frameworks, such as MRS or Hole Semantics, implement such an ET, which does not correspond to an actual predication of the sentence, but serves as the highest level predication of the sentence, encompassing the overall semantics. For example, in a typed feature structure formalism like MRS, this ET contains attribute-value pairs, encoding some global semantic properties of the sentence, such as the speech act, which is not particular to any individual elementary predication.

8 We intentionally avoid using the existing term *plugging* from Hole Semantics, because *plugging* is defined as a function from holes to labels (Bos 1996), while, in order to be able to fuse multiple labels to a single hole, we define fusion in the opposite direction. This gives UG the power to model MRS's operation of forming conjunctions by equating labels.



(a) Graphical depiction of a fusion function (b) $\mathcal{T}_{U,f}$ for fusion f on left

Figure 4
Building a solution for a UG.

Figure 4(b). Intuitively, we expect scopings to form a tree. Theorem 1 states the necessary and sufficient condition for $\mathcal{T}_{U,f}$ to be a tree.

Theorem 1
Given a UG U and a total fusion f of U, $\mathcal{T}_{U,f}$ is a rooted tree if and only if $\mathcal{T}_{U,f}$ is acyclic.

Proof. The "only if" direction is trivial. The "if" direction holds for the following reason. Because f is a function, every label is fused into at most one hole, hence, has at most one parent in $\mathcal{T}_{U,f}$. If f is total, then every label except l_0 is fused into exactly one hole, hence, has exactly one parent in $\mathcal{T}_{U,f}$. Therefore, with U being finite, if we start at any arbitrary node and follow the sequence of parents, we have to end up at l_0. This means $\mathcal{T}_{U,f}$ is a tree rooted at l_0. □

Although fusion is defined as any function from $\tilde{L}$ to H, we are only interested in fusions that result in valid readings, that is, satisfy the constraints.

Definition 4 (Constraint Satisfaction and Admissibility)
Fusion f (similarly $\mathcal{T}_{U,f}$) **satisfies**

- a qeq constraint $q = (h, l)$, if $h = f(l)$, or the directed path from h to l in $\mathcal{T}_{U,f}$[9] consists of only the body edges of quantifier ETs (called a b-**path**);
- a dominance constraint $d = (u, v)$, if u dominates v in $\mathcal{T}_{U,f}$.

Fusion f of U is **admissible** if $\mathcal{T}_{U,f}$ is acyclic and satisfies all the constraints in U.

So far we have informally used the term *scoping* to refer to a fully scope-disambiguated UR. We now formally define this notion and call it a solution.

9 Note that each node u of a solution $\mathcal{T}$ (except the root) corresponds to exactly one hole h_i and at least one label $l_j$of U with $f(l_j) = h_i$. Hence, when we say node h_i, node l_j, or node (l_j, h_i) of $\mathcal{T}$, in all three cases, we are referring to the same node u of $\mathcal{T}$.

Definition 5 (Solution)
$\mathcal{T}$ is called a **solution** of a UG U iff $\mathcal{T} = \mathcal{T}_{U,f}$ for some admissible, total, and *onto*[10] fusion f of U. In informal contexts, solutions are sometimes referred to as **readings**. U is called **satisfiable** if U has at least one solution.

In this definition, admissibility ensures the satisfaction of all constraints, totality ensures that every label (except the top) is fused into some hole, and onto-ness ensures that no hole is left unfused. Following Lemma 1, Definition 5 guarantees that every solution is a tree structure. The fusion in Figure 4(a) is admissible, total, and onto, and hence, Figure 4(b) is a solution.

The following definition will later be used for the comparison of our framework with other frameworks.

Definition 6 (Merging-Free Solutions)
The solution $\mathcal{T}_{U,f}$ is called a **merging-free**, if f is a one-to-one function. Otherwise, it is called a merging solution.

The following lemma directly follows from the definition.

Lemma 1
Given a UG U and a solution $\mathcal{T}$ of U, $\mathcal{T}$ is merging-free if and only if $|\tilde{L}_U| = |H_U|$.

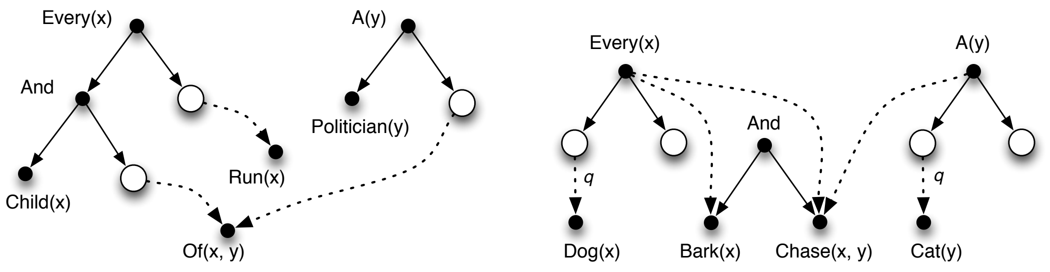
Corollary 1
Either all the solutions of a UG are merging-free or none are.

In practice, merging solutions only happen when the underspecified representation is incomplete, that is, when there are ETs that are floating around and, in order to build a solution, they must be fixed up with other ETs to make conjunctions. In Section 3, we prove that all solutions of a complete UG are merging-free.

2.2 Variations of UG

The definition of UG requires each ET to be of depth at most one and, when the depth is exactly one, all the leaf nodes to be hole nodes. This definition perfectly imitates the notion of elementary predications in MRS, but some frameworks, such as Hole Semantics, use the concept of (*labeled*) *formula*, which does not precisely fit into this definition. Figure 5(a) shows a graph, roughly corresponding to the UR that Hole Semantics assigns to the sentence *Every child of a politician runs*. As seen in this figure, we have to deal with ETs with depth more than one and leaves that are not necessarily a hole. Even in MRS, ETs can be stacked to form trees of depth more than one. Figure 5(b)

10 A function f is *onto* if its image is equal to its codomain.



(a) Hole Semantics UR with stacked ET (b) MRS structure for the sentence in Example (2)

Figure 5
Examples of stacked ETs.

shows the MRS structure of the following sentence in graphical notation, obtained from English Resource Grammar.[11]

2. *Every dog barks and chases a cat.*

Finally, a partially scoped UG U, even if all the ETs of U are standard, will inevitably have these non-standard tree structures. Therefore, for the sake of the robustness of our definitions, we should be able to model these structures. Fortunately, we will be able to do this without too much effort. This is because a stacked ET can be converted into a standard ET without affecting the number of solutions of the UG. This conversion has been demonstrated in Figure 6(a). We now formally express this intuition.

Definition 7 (Stacked ET)
A **stacked ET** is an ordered tree of arbitrary depth whose interior nodes are all label nodes, and whose leaves can be holes and/or labels.

Definition 8 (UG.1: variation 1 of UG)
A **UG.1** is a 9-tuple $\dot{U} = \langle L_{\dot{U}}, L'_{\dot{U}}, H_{\dot{U}}, E_{\dot{U}}, Q_{\dot{U}}, D_{\dot{U}}, T_{\dot{U}}, L^q_{\dot{U}}, P_{\dot{U}} \rangle$ ($L'_{\dot{U}}$ is the only additional element with respect to standard UG), where $\mathcal{F}_{\dot{U}} = \langle L_{\dot{U}}, L'_{\dot{U}}, H_{\dot{U}}, E_{\dot{U}}, P_{\dot{U}} \rangle$ is a forest of *stacked* ETs with $L'_{\dot{U}}$ being the set of non-root label nodes. Everything else in Definition 2 remains the same.[12]

The definitions of *fusion*, *admissibility*, and *solution* for UG.1 will be exactly the same as the definitions of those concepts for standard UG, as stated in Definitions 3, 4, and 5.

11 `http://erg.delph-in.net/`.
12 Note that stacked floating-scopal ETs, exactly the same as standard ones, are required to have a hole as the right-most child of the root.

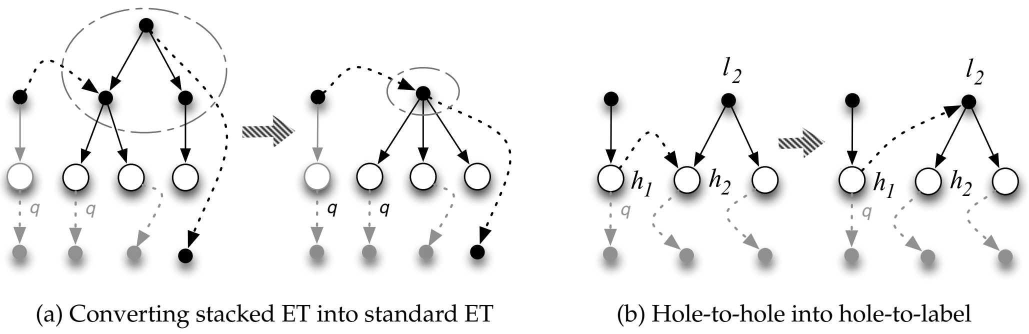

q
q
q
q
q
q
l_2
l_2
h_1
h_2
h_1
h_2
(a) Converting stacked ET into standard ET
(b) Hole-to-hole into hole-to-label

Figure 6
Conversion to original UG.

Theorem 2
Every UG.1 can be converted into a UG, while the solutions remain in a one-to-one correspondence.

Proof. Consider a UG.1 $\dot{U}$ with its set of solutions $\{\dot{\mathcal{T}}_1, \dot{\mathcal{T}}_2, \ldots \dot{\mathcal{T}}_K\}$. We build the UG U by collapsing the set $L'_{\dot{U},\mathcal{E}}$ of non-root label nodes of each stacked ET $\mathcal{E}$ into its root $l_{\mathcal{E}}$, as demonstrated in Figure 6(a). Similarly, we convert each tree $\dot{\mathcal{T}}_j$ into $\mathcal{T}_j$ by collapsing the nodes $L'_{\dot{U},\mathcal{E}}$ of $\mathcal{T}$ into $l_{\mathcal{E}}$ for each stacked ET $\mathcal{E}$. It is easy to see that $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_K\}$ is the set of solutions of U. □

In defining UG, we ruled out dominance constraints that go from holes to holes. This does not restrict the power of UG, because hole-to-hole dominance constraints can be replaced with hole-to-label constraints, while the set of solutions remains the same, as demonstrated in Figure 6(b). In the following, we will state this idea formally.

Definition 9 (UG.2: variation 2 of UG)
A **UG.2** is a 8-tuple $\ddot{U}$, in which the constraints in $D_{\ddot{U}}$ can go from any node to any node. Everything else in Definition 2 remains the same.

Theorem 3
Every UG.2 can be transformed into a UG, while the set of solutions remains the same.

Proof. Consider the UG.2 $\ddot{U}$ and a constraint $\ddot{d} = (h_1, h_2)$ in $\ddot{U}$ (Figure 6(b)), and let l_2 be the parent of h_2. We build U by replacing $\ddot{d}$ in $\ddot{U}$ with $d = (h_1, l_2)$, as demonstrated in Figure 6(b). $\ddot{U}$ and U have the same set of solutions. This is because if $\mathcal{T}$ is a solution of $\ddot{U}$, then h_1 dominates h_2 in $\mathcal{T}$. Because l_2 immediately dominates h_2 in $\mathcal{T}$, h_1 dominates l_2 as well, hence, $\mathcal{T}$ satisfies d. The other direction is trivial. This procedure can be repeated until all hole-to-hole constraints are transformed. □

In the preceding sections, we defined two variations of UG that relaxed some restrictions of the original definition, but we showed that this did not increase their power. In the following, we define some other variations, imposing some restrictions on UG. These restrictions, however, limit the power of UG. The main motivation behind defining these variations is to be able to compare UG with other frameworks (Section 7).

Definition 10 (Normality)
A UG is said to be **normal** iff all its dominance constraints go from holes to labels, that is, $D_U \subset H_U \times L_U$.

Normality therefore rules out constraints emanating from a label node.

Definition 11 (Weak Normality)
A UG is said to be **weakly normal**, iff for every dominance constraint $d = (u, v)$, v is a label node.

Note that weak normality only rules out label-to-hole constraints.

3. Completeness

In this section, we define the notion of completeness. Intuitively, completeness means that we have the minimum connectivity in terms of constraints that is required by the syntax/semantic interface for a complete sentence. Here, we formally define the notion of completeness and prove some properties.

Definition 12 (Canonical Form)
A UG is in **canonical form** (CF-UG) if

- The body hole of floating scopals is not involved in any qeq edge. Every other hole has exactly one outgoing qeq edge.
- Floating scopal nodes, as well as the top label, are not involved in any qeq constraints. Every other label has exactly one incoming qeq edge.

Before we continue, let us define the notion of spanning sub/super-UG. Intuitively, spanning sub/super-UG of G has the same ETs as the G, and only its set of constraints differs. The motivation is to define a type of sub/super relation under which completeness is closed (a super-UG of a complete UG is not necessarily complete, because it may have one or more additional ETs that violate completeness conditions).

Definition 13 (Spanning Sub-UG)
U' is a **spanning sub-UG** of U, if $D_{U'} \subset D_U$ and $Q_{U'} \subset Q_U$. Every other element of the tuple U' is identical to the corresponding item in the tuple U. In particular, notice that the set of ETs is the same in both U and U', hence, the term **spanning**. U' is a **spanning super-UG** of U, if U is a spanning sub-UG of U'. U^q is defined as the spanning sub-UG of U with no dominance edges, that is, $D_{U^q} = \emptyset$ and $Q_{U^q} = Q_U$.

Theorem 4 shows that CF-UGs, ignoring dominance edges, are in the form of Figure 7.

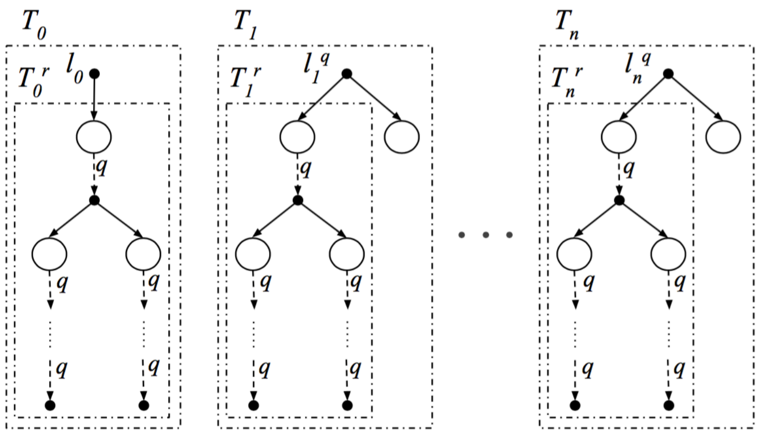


Figure 7
U^q of a generic CF-UG U.

Theorem 4
If U is in **canonical form**, then U^q is a forest of exactly $|L_U^q| + 1$ trees, rooted at $L_U^q \cup \{l_0\}$. $R_U =_{\text{def}} L_U^q \cup \{l_0\}$ are called the roots of U.

Proof. According to the second condition in Definition 12, the n quantifier nodes and l_0 are the only nodes with no incoming edge, therefore they form all and the only $n + 1$ roots of the graph of a CF-UG. Every other label or hole node must be dominated by one of those $n + 1$ roots. According to the first condition of Definition 12, the body holes of quantifiers have no outgoing edge, therefore every other node must be either under the restriction of a quantifier or under h_0. □

Because qeq constraints were introduced by MRS, in Section 7.2, we show that, in practice, all MRS structures generated by MRS's proposed syntax/semantic interface are in canonical form. Canonical form defines the smallest complete UG over a set of ETs.

Definition 14 (Completeness)
A UG is **complete** if it has a spanning sub-UG in canonical form.

The following set of definitions become handy throughout the rest of this paper.

Definition 15 (Floating Scopal Trees/Restriction Trees/Heart Tree)
We refer to $T_1, T_2, \ldots, T_n$, the trees in U^q rooted at L_U^q, as **floating scopal trees** (a.k.a. **quantifier trees**); $T_1^r, T_2^r, \ldots, T_n^r$, the trees rooted at the restriction hole of the floating scopals, as the **restriction trees**; and T_0, the tree rooted at l_0, as the **heart tree**.

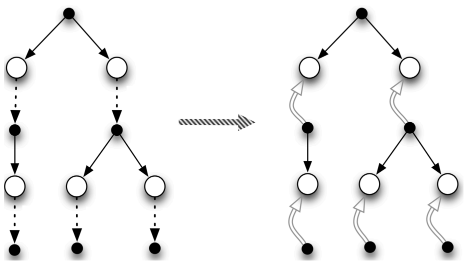
3.1 Equivalence of Qeq and Dominance Relations

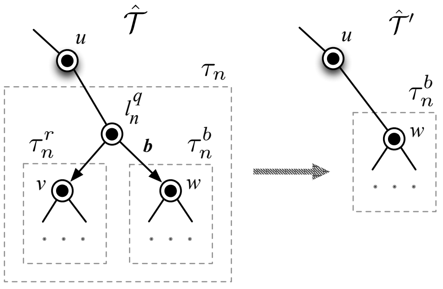
In this section, we show that, for every complete UG, qeq and dominance relations are equivalent. That is, as stated by Theorem 5, if qeq constraints are replaced with dominance relations, the solutions of the UG remain the same. This fact explains why frameworks such as Dominance Graph are able to model scope underspecification even though they only use dominance constraints, and helps to bridge the gap between the two sets of formalisms. We first prove this for the case when there is no quantifier in U.

Lemma 2
Let U be a complete UG with no quantifier. Let $\hat{U}$ be the UG obtained from U by treating all qeq edges as dominance constraints, that is, $D_{\hat{U}} = D_U \cup Q_U$ and $Q_{\hat{U}} = \emptyset$. Every solution of U is a solution of $\hat{U}$ and vice versa.

Proof. The first direction is trivial because qeq is a special case of dominance. In order to prove the other direction, consider the leaf label nodes of $\hat{U}$ (Figure 8(a)). In any solution of $\hat{U}$, these label nodes have to be fused to the hole from which they have received a dominance constraint. Let us fuse these labels, and then, following Theorem 2, collapse the EPs with the fused hole(s) into a single node. We can now apply the same argument to the newly leaf nodes and repeat this until we reach the top (remember that UGs are finite). This shows that every hole h of $\hat{U}$ has to be fused with the label l where $(h, l) \in Q_U$. Because h is fused directly with l, following the definition of qeq, the constraint between h and l is satisfied, even if it is treated as qeq. This means that $\hat{\mathcal{T}}$ is also a solution of U. □

Theorem 5
Let U be a complete UG, and $\hat{U}$ be the UG obtained from U by treating all qeq edges as dominance constraints, that is, $D_{\hat{U}} = D_U \cup Q_U$ and $Q_{\hat{U}} = \emptyset$. Every solution of $\hat{U}$ is a solution of U and vice versa.


(a) A UG with no quantifier



(b) Inductive proof

Figure 8
Proof of Lemma 2 and Theorem 5.

Proof. Because qeq always implies dominance, it is trivial that every solution of U is a solution of $\hat{U}$. Using the lemma and induction on n, the number of quantifiers, we prove the other direction, that is, if $\hat{\mathcal{T}}$ is a solution of $\hat{U}$, then it is also a solution of U.

Let $n = 0$. Because there is no quantifier in $\hat{\mathcal{T}}$, according to Lemma 2, every hole h is fused with l where $(h, l) \in Q_U$, therefore, every qeq constraint is satisfied.

Now let $n > 0$ and U be an arbitrary UG with n quantifiers (see Figure 7), and $\hat{\mathcal{T}}$ be a solution of $\hat{U}$. Consider the quantifier node l^q with the longest distance from the root of $\hat{\mathcal{T}}$ (breaking ties arbitrarily), meaning that l^q does not outscope any other quantifier node in $\hat{\mathcal{T}}$. Without loss of generality, assume that $l^q = l^q_n$ and use $\hat{\tau}_n$, $\hat{\tau}^r_n$, and $\hat{\tau}^b_n$ to refer to the trees rooted at l^q_n and the left and the right child of l^q_n in $\hat{\mathcal{T}}$, respectively, as shown in Figure 8(b). According to Lemma 2:

(i) *All qeq constraints in the quantifier tree* T^r_n *are satisfied in* $\hat{\mathcal{T}}$.

Now let us remove the quantifier tree rooted at l^q_n from U and $\hat{U}$ and call the resulting UGs U' and $\hat{U}'$, respectively. Accordingly, detach the tree $\hat{\tau}_n$ from $\hat{\mathcal{T}}$, replace it with $\hat{\tau}^b_n$ and call the new tree $\hat{\mathcal{T}}'$, as demonstrated in Figure 8(b). $\hat{\mathcal{T}}'$ is a solution of $\hat{U}'$, and hence (based on the induction assumption) is a solution of U'. Therefore, all qeq constraints in U' are satisfied in $\hat{\mathcal{T}}'$, which also proves:

(ii) *All qeq constraints in* U' *are satisfied in* $\hat{\mathcal{T}}$.

This is because if two nodes are connected with a b-path in $\hat{\mathcal{T}}'$, they are also connected with a b-path (possibly including an additional b-edge (l^q_n, w)) in $\hat{\mathcal{T}}$.

From (i) and (ii) every $(h, l) \in Q_U$ is satisfied in $\hat{\mathcal{T}}$, hence $\hat{\mathcal{T}}$ is a solution of U. □

The next section introduces the heart of our framework, the concept of semantic coherence.

4. Coherence

In this section, we introduce the notion of sentence-level semantic coherence based on a simple principle: that every variable introduced in the domain of discourse must contribute to the overall meaning of the sentence. We formally characterize this quality as a property of a UG, and refer to sentences whose interpretation is a coherent UG as **coherent sentences**. We posit as a general principle of language the requirement that sentences should be coherent. This general principle goes back at least as far as Frege (1923), and is widely accepted, although, because it is not a mathematical statement, it cannot be proved. Our definition of coherent UG, on the other hand, is mathematically precise, and can be used to prove that all coherent sentences are tractable, as we will see in Section 5.

4.1 Mathematical Characterization of Semantic Coherence

A variable introduced in the domain of discourse is called relevant if it contributes to the overall meaning of a sentence. This contribution may happen in two different ways, either by directly participating in the main predication[13] or by participating in the definition of another relevant variable. If variable x participates in the definition of y, we say that y is (semantically) dependent on x. To summarize, a variable x is relevant if either the heart or another relevant variable depends on it. For example, in the sentence *Every child of a politician runs*, the variable x, quantified by *Every*, is relevant, because it is an argument of the main predication, and the variable y, quantified by A, is relevant, because x depends on it. Following these intuitions and given that dependencies in a UG are encoded in the binding (i.e., dominance) constraints, we formally define relevance as follows.

Definition 16 (Dependence/Relevance)
Consider a complete U,[14] with the top hole l_0 and $l^q_i, l^q_j \in L^q_U$:

- l^q_j (l_0, for $j = 0$) **depends** on l^q_i, if $(l^q_i, u) \in D_U$ for some u in a restriction tree T^r_j.
- l^q_i is said to be **relevant** if
  - (i) l_0 depends on l^q_i; or
  - (ii) l^q_j depends on l^q_i, and l^q_j is relevant.

Definition 17 (Coherence)
A complete UG U is called **coherent** if every l^q_i in U is relevant.

Trivially, relevance is closed under the increment of constraint edges, resulting in the following lemma.

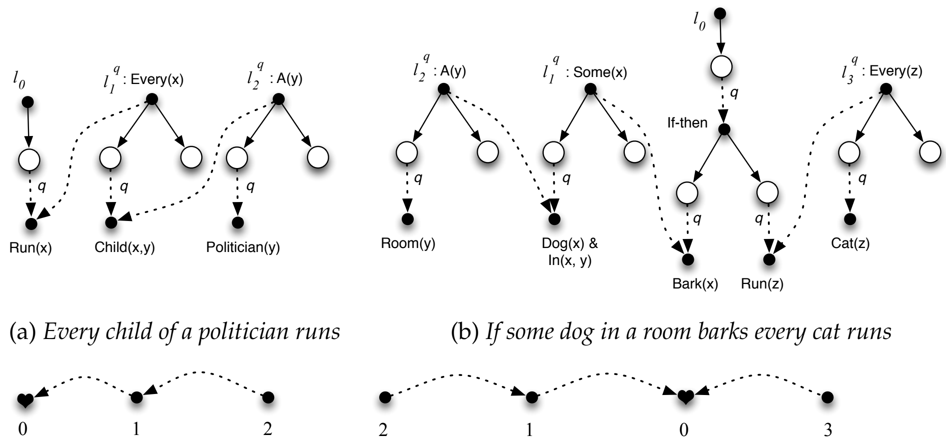
Lemma 3
Coherence is closed under the increment of (dominance) constraint edges.

In Manshadi, Allen, and Swift (2009), we defined the notion of semantic dependency graph and a property of such graphs called **heart-connectedness**. Heart-connectedness is nothing but a notational variant of coherence. In this article, we shall use the notion of semantic dependency graph to compare our framework with Hole Semantics (Section 7.3). Therefore, in the rest of this section, we formally define this notion and prove that the two formulations of coherence are in fact equivalent.

Definition 18 (Semantic Dependency Graph)
Given a CF-UG U, we define **semantic dependency graph** or **SDG** of U as $\mathrm{SDG}_U = (V, E)$, where

- $V = \{0, 1, \ldots, n\}$, where $n = |L^q_U|$.
- $(i, j) \in E$ if and only if $i \neq j$, $i > 0$, and $(l^q_i, u) \in D_U$, where u is a node in T^r_j.

13 Here, by participation, we mean filling an argument position.
14 Hence, dependence and relevance are defined only if U is complete.



(a) *Every child of a politician runs* (b) *If some dog in a room barks every cat runs*

(c) SDG of the UG in (a) (d) SDG of the UG in (b)

Figure 9
Constructing semantic dependency graph.

Intuitively, SDG$_U$ is obtained by taking the CF-UG U and collapsing its heart and quantifier trees—that is, each of the trees $T_0, \ldots, T_n$—into a single node, resulting in a directed graph G of $n + 1$ nodes. Figure 9 demonstrates this transformation for two real-life CF-UGs. The dependencies in U are simply encoded in the edges of G. In other words, if i is the node of G corresponding to T_i, an edge from i to j in G means that l_j^q (l_0, if $j = 0$) depends on l_i^q in U. From Definition 18, SDGs are **simple** graphs, that is, (i) they have no self-loops, and (ii) for every i, j there is at most one (directed) edge from i to j. Node 0, which corresponds to the heart of the CF-UG, hence called the **heart** of SDG, has no outgoing edge, therefore the heart is always a sink node.

Definition 19 (Heart-Connectedness)
A SDG G is called **heart-connected** if every node in G reaches the heart by a directed path. A CF-UG U is called **heart-connected**, if SDG$_U$ is heart-connected.

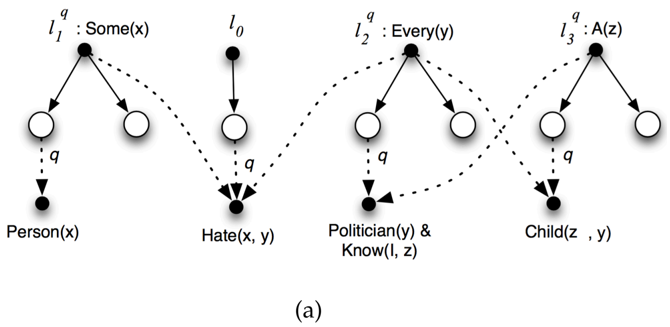
Theorem 6
A CF-UG U is coherent if and only if it is heart-connected.

The theorem directly follows from the following lemma.

Lemma 4
The node l_i^q is relevant in a CF-UG U, if and only if i reaches the heart in SDG$_U$.

Proof. Following Definition 16, the set R of all the relevant nodes in U can be constructed as follows:

- $R_0 = \{l_0\}$.



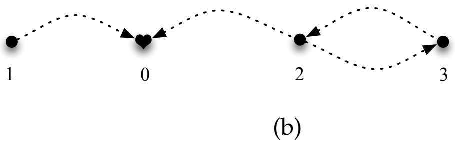


Figure 10
UG and SDG for *Somebody hates every politician whom I know a child of.*

- $R = \bigcup_{m=0}^{n} R_m$, where R_m ($m > 0$) is the set of nodes that $R_{(m-1)}$ depends on.[15]

Using induction on m, it is easy to see that for every node $l_i^q \in R$, node i reaches the heart in SDG$_U$ using a directed path of length m. □

All examples of UG given so far are acyclic. This may suggest that the SDG of every sentence is acyclic, but this is not the case. As a counterexample, consider the UG in Figure 10, motivated by an example from Hobbs and Shieber (1987). This example shows that coherent UGs form a very broad class of UGs, subsuming other previously proposed classes, as we will see in Section 7. Notwithstanding their broad applicability, coherent UGs can be tractably processed, as we will see in the next section.

5. Tractability

An algorithmic problem arising within the context of constraint-based underspecification frameworks is to determine whether a given UG has a solution or not. This is called the **satisfiability** problem, or, in short, **SAT**. This becomes important when new constraints are incrementally added at the deeper levels of language processing. Another closely related problem commonly studied within the same context is to enumerate all possible solutions, the **enumeration** problem, or, in short, **ENUM**. All the constraint-based underspecification frameworks that we have built UG upon (HS, MRS, and DG) are intractable in their general form, meaning that their satisfiability problem is NP-complete. Over the last decade, there has been a series of work on finding a subset of these frameworks that can be solved efficiently. The previously found tractable subset

15 We require that $R_m \subset L_U^q - \bigcup_{j=0}^{(m-1)} R_j$.

is inadequate in that it does not cover all natural language sentences (see Definition 29 in Section 7.1 for details), leaving open the question whether there is a tractable subset with sufficient expressivity. In this section, we answer this question by introducing the largest tractable subset found so far and prove that every coherent sentence, under our linguistically justified notion of semantic coherence, belongs to this subset.

5.1 Dominance Graph

In the previous sections, we showed that for coherent (in fact, complete) UGs, qeq and dominance constraints are equivalent. A UG with only dominance constraints is called a **dominance graph**, or, in short, **DG**, which is the core concept of the Dominance Graphs framework. Most of the work on finding a tractable subset of URs has previously been done within the realm of this framework. Because coherent UGs are a subset of dominance graphs, our work is built on top of this work, hence, in this section we only work within this framework.

Remember from Definition 2 that a UG is an 8-tuple $\langle L, H, E, Q, D, T, L^q, P\rangle$. With no qeq constraints, a dominance graph has no Q component. As a result, there is no need to distinguish floating scopal (i.e., quantifier) ETs; and hence, there is also no L^q component, which in turn means that there is no designated top ET in dominance graphs. Therefore, all labels can potentially form the root of a solution. We now give a formal definition of dominance graph.

Definition 20 (Dominance Graph)
A **Dominance Graph**, or DG, is a 5-tuple $G = \langle L_U, H_U, E_U, D_U, P_U\rangle$ where all the components are as defined in Definition 2. Because there is no designated top node, analogous to $\tilde{L}_U$, we define $\tilde{L}^l_G = L_U - \{l\}$ for every label node l. All the variations of UG defined in Section 2.2 are defined correspondingly for DG.

It should be noted that in order to build a fusion, we have to first pick an arbitrary label l, and then construct f as a function from $\tilde{L}^l_G$ to H. The node l will be a root of $\mathcal{T}_{U,f}$ (the only root, if f is total). Following this definition, every UG can be converted into a DG by dropping the top ET and treating all qeq constraints as dominance constraints.

Definition 21 (DG Counterpart)
Given a UG U, G_U, the **DG counterpart** of U, is obtained by removing the top ET from U and converting all qeq edges into dominance. More precisely, $G_U = \langle H_G, L_G, E_G, D_G, P_G\rangle$ where $H_G = H_U - \{h_0\}, L_G = L_U - \{l_0\}, E_G = E_U - \{(l_0, h_0)\}, D_G = D_U \cup Q_U - \{(u,v) \mid \{u,v\} \cap \{l_0, h_0\} \neq \emptyset\}, P_G = P_U$.

The following lemma directly results from Theorem 5.

Lemma 5
There is a one-to-one relationship between the solutions of a complete UG U and those of its DG counterpart G_U.

In order to prove the tractability of coherent UG, we define a mathematically (more) convenient notion, a subset of DG called **hypernet**. Hypernet includes all coherent UGs,

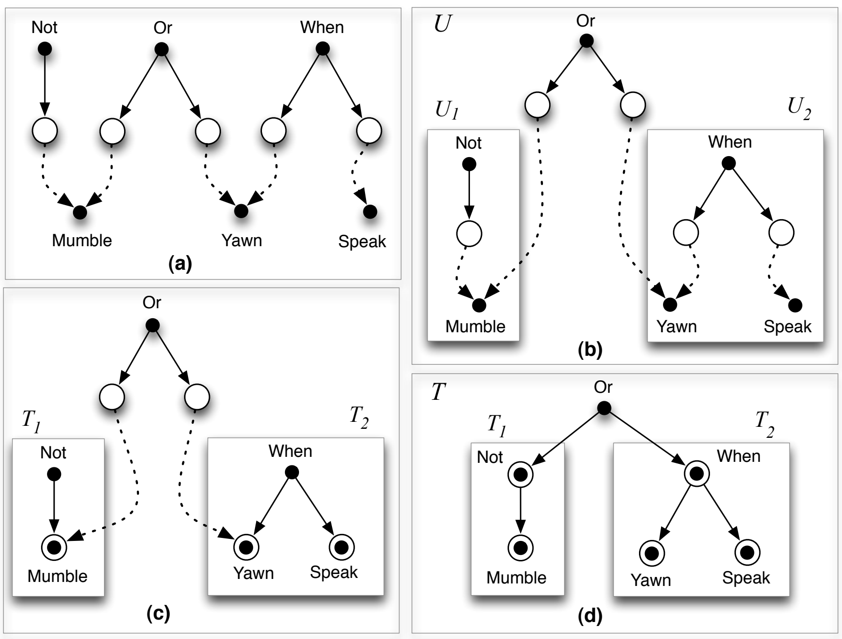


Figure 11
Recursive construction of solutions.

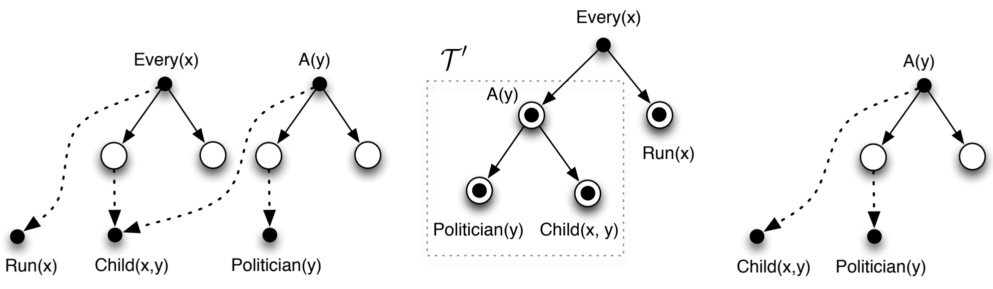
and we treat it as the mathematical characterization of coherence within the context of DGs. The definition of hypernet is motivated by the definition of (weak) nets (Niehren and Thater 2003), a previously found tractable subset of DG, and it translates our semantically motivated concept of coherence into (a slightly more powerful version of) the already popular structures of DG. Built on top of the fairly complex notion of nets together with the incorporation of heart-connectedness (as the mathematical characterization of coherence), there should be no surprise that the definition of hypernet is quite complex. For this reason, instead of presenting the complete definition at once, we step by step justify our way through the full definition of hypernet.

Remember that our ultimate goal is to efficiently solve (a subset of) DGs. Here is a recursive approach. Pick an ET for the root of the solution and remove it from the DG. Recursively solve each of the remaining smaller DGs and plug the root of the resulting trees into the holes of that ET. Figure 11 demonstrates this procedure and Table 1 lists the steps. In order for this approach to work, each of the resulting subgraphs should be a valid DG. Let us make this intuition formal.

Definition 22 (Sub-DG)
A **sub-DG** G' of G is a DG whose set of ETs and constraints is a subset of ETs and constraints in G, respectively. A **spanning sub-DG** of G is a sub-DG that includes all the ETs in G. An **induced** sub-DG G' of G (by a subset $\mathcal{F}$ of ETs in G) is a sub-DG of G in which (the set of ETs is $\mathcal{F}$ and) a constraint of G is present in G', *if and only if* both its

Table 1
Recursive procedure followed in Figure 11.

```
1: procedure SOLVE(DG $G$)
2:     If $G$ contains a single (label) node, return that node as a single-node tree $\mathcal{T}$.
3:     Pick an ET $\mathcal{E}$ satisfying all the conditions to be the root of a solution, otherwise,
       fail.
4:     Let $G^r, G^b$ be the two DGs resulting from the removal of $\mathcal{E}$ (Figure 11(b)).
5:     Let $\mathcal{T}^r$ =SOLVE($G^r$), $\mathcal{T}^b$ =SOLVE($G^b$)
6:     Build $\mathcal{T}$ by plugging $\mathcal{T}^r$ into $h^r$ and $\mathcal{T}^b$ into $h^b$.
7:     return $\mathcal{T}$.
```



(a) DG G (b) Solution $\mathcal{T}$ of G (c) Sub-DG induced by $\mathcal{T}'$

Figure 12
DG counterpart of the UG for *Every child of a politician runs* and a sub-DG of it.

ends are present in G'. Given a solution $\mathcal{T}$ of G and a subtree [16] $\mathcal{T}'$ of $\mathcal{T}$, a sub-DG G' **induced** by $\mathcal{T}'$ is the sub-DG induced by the set of ETs of $\mathcal{T}'$.

Figure 12 gives an example of these concepts. The following property, which directly follows from the definition, will help in proving the completeness of the recursive method (Section 5.3).

Lemma 6
Consider a DG G, a solution $\mathcal{T}$ of G, and a subtree $\mathcal{T}'$ of $\mathcal{T}$. If G' is the sub-DG of G induced by $\mathcal{T}'$, then $\mathcal{T}'$ is a solution of G'.

The notion of sub-DG helps in solving DGs recursively, as shown in Figure 12. Depending on whether, at step 3, among all nodes satisfying the conditions, we pick a node *arbitrarily* or *iteratively* determines whether the algorithm will be a SAT algorithm (generating an arbitrary solution on success) or an iterative ENUM algorithm (every branch of which builds one solution). Note that, whereas the total size of the output for ENUM may be exponential, we are interested in bounding its running per solution produced.

16 Throughout this article, by "subtree of a tree T," we mean a node and all of its decedents in T.

In the rest of this section, we form this intuition into a detailed algorithm and define hypernet as a subset of DG for which the algorithm is both sound and complete. Because label-to-hole constraints add another level of complexity, we first ignore those edges and only consider weakly normal DGs. We then adopt the algorithm by Koller and Thater (2007) to take care of label-to-hole constraints.

5.2 Hypernets

A look at the steps in Table 1 reveals that the heart of this procedure is to find the roots of the solutions. The rest is simply recursing on the smaller DGs.

Definition 23 (Freeness)
A label l is said to be **free** if it is the root of some solution $\mathcal{T}$ of G. An ET $\mathcal{E}$ is called free if its root is free.

Subsequently, we use some graph theory terminology that we need to define. Remember that two nodes of a digraph are **weakly connected** if there is an undirected path between the two. Because weak connectedness is an equivalence relation, it partitions the graph into equivalence classes each of which is called a **weakly connected component** or **WCC**.

Theorem 7 states some necessary conditions for freeness, but first, we state a lemma that is the key to the proof of this theorem.

Lemma 7
Given a weakly normal DG G and a solution $\mathcal{T}$ of G, if nodes u and v in G are weakly connected using an undirected path p, there exists a node w on p such that w dominates both u and v in $\mathcal{T}$.

This lemma is analogous to Lemma 3 in Bodirsky et al. (2004). A detailed proof of the lemma can be found there. We are now prepared to prove the theorem.

Theorem 7
Let G be a weakly normal *DG* and $\mathcal{E}$ be an ET with m holes rooted at label l. l can be a free label, *only* if

T7a. l has no incoming (constraint) edge; and

T7b. Every distinct hole of $\mathcal{E}$ is in a distinct WCC in $G - l$; and

T7c. $G - \mathcal{E}$ consists of at least m WCCs.

Proof. Let $\mathcal{T}$ be a solution rooted at l, h a hole of $\mathcal{E}$, and G_h the WCC of $G - l$ containing h.

T7a. This condition is trivial.

T7b. Following Lemma 7, every node in G_h must be in the scope of h (because for any arbitrary node u in G_h, there is a node w dominating both h and u

in $\mathcal{T}$, but the only possible w is $w = h$). Because $\mathcal{T}$ is a tree, this proves that every two holes of $\mathcal{E}$ belong to two distinct WCCs.

T7c. This follows from T7b and the fact that no hole may be left unfused. □

We now look for a subset of DG, for which the conditions in T7a through T7c are not only necessary, but also sufficient. In defining this subset, the following notion from Althaus et al. (2003) plays an important role.

Definition 24 (Hyper-Normal Connectedness)
Given a DG G, a **hyper-normal path** is an undirected path with no two consecutive constraint edges emanating from the same node. Node u is **hyper-normally connected** to node v if there is at least one hyper-normal path between the two. G is called **hyper-normally connected** if every pair of nodes in G are hyper-normally connected.

For example, in Figure 13, p_2 is a hyper-normal path, but p_1 is not. In spite of that, the whole graph is hyper-normally connected, because even though p_1 is not a hyper-normal path, there is another hyper-normal path connecting the same two nodes. The following notion from Thater (2007) will also come in handy.

Definition 25 (Openness)
A label or hole node u is called an **open node** if it has no *outgoing* constraint edge.

For example, l in Figure 14(a) is an open label node and h_2 in Figure 14(b) is an open hole.

Definition 26 (Hypernet)
A DG G is called a **hypernet** if it has a weakly normal spanning sub-DG G' in which for every elementary tree $\mathcal{E}$ rooted at l:

D26a. $\mathcal{E}$ has at most one open node.

D26b. If l_1 and l_2 are two dominance children of a hole h of $\mathcal{E}$, then l_1 and l_2 are hyper-normally connected in $G' - h$.

D26c. If $\mathcal{E}$ has no open hole in G':

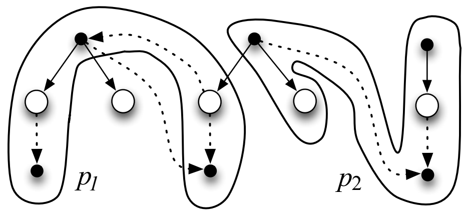


Figure 13
Illustration of hyper-normal path.

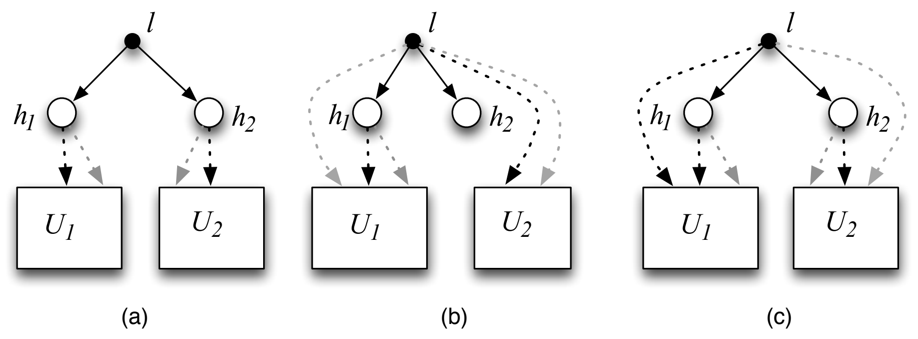


Figure 14
Three types of elementary trees: (a) open-root (b) open-hole (c) closed.

- Each dominance child of l is hyper-normally connected to a hole of $\mathcal{E}$ in $G' - l$.

Otherwise (i.e., when $\mathcal{E}$ has an open hole):

- All dominance children of l, not connected to a hole of $\mathcal{E}$ in $G' - l$, are hyper-normally connected together.

Here, by dominance children of u, we mean nodes v where (u, v) is a dominance edge. Notice that by enforcing conditions (D26a) through (D26c) on a spanning sub-DG of G rather than G itself, we allow the hypernet to be closed under the increment of constraint edges. This is in line with the definition of coherence, which imposes only a minimum connectivity, hence, closed under the increment of dominance constraint edges.[17] Following condition (D26a), a weakly normal hypernet may have three types of scopal ET, characterized in the following definition.

Definition 27
Given a DG G and an ET $\mathcal{E}$, $\mathcal{E}$ is called

D27a. **Open-root**: iff only the root of $\mathcal{E}$ is open in G.

D27b. **Open-hole**: iff only a hole of $\mathcal{E}$ is open in G.

D27c. **Closed**: iff $\mathcal{E}$ has no open node in G.

Figures 14(a–c) show the three types of $\mathcal{E}$, respectively. Definition 26 guarantees the following property.

Theorem 8
Let G be a weakly normal hypernet and $\mathcal{E}$ be an ET of G with m holes and rooted at l. If l satisfies the conditions in Theorem 7, $G - \mathcal{E}$ consists of exactly m WCCs, each of which is a (weakly normal) hypernet.

17 Koller and Thater (2007) call this property "downward connectedness."

Proof. Following conditions (D26b) and (D26c), $G - \mathcal{E}$ consists of at most m WCCs (since the children of each node of $\mathcal{E}$ are hyper-normally connected, they stay connected in $G - \mathcal{E}$). Based on condition (T7c), $G - \mathcal{E}$ has at least m WCCs. Therefore, $G - \mathcal{E}$ has exactly m WCCs. To prove that each of these WCCs is a hypernet, we show that, for any two nodes u and v not belonging to $\mathcal{E}$, if u and v are hyper-normally connected in G, they are also hyper-normally connected in $G - \mathcal{E}$. We do so by proving that there is no hyper-normal path between u and v in G that visits some node of $\mathcal{E}$.

Suppose that $\mathcal{E}$ is an open-hole ET rooted at l (Figure 14(b); a similar line of reasoning can be used for open-root and closed ETs). Assume to the contrary that there is a hyper-normal path p between u and v that visits some node of $\mathcal{E}$. Because an ET consists of a single label node l with some number of holes as children, one of the following three cases holds:

i. p visits exactly one node of $\mathcal{E}$.

ii. p visits (at least) two holes of $\mathcal{E}$.

iii. p visits l and exactly one hole of $\mathcal{E}$.

All three cases result in a contradiction: Case (i) means that p is not hyper-normal; case (ii) shows that $\mathcal{E}$ violates condition (T7b); and case (iii) proves that G is not a hypernet, because $\mathcal{E}$ violates condition (D26c). □

The following corollary will be helpful in the subsequent subsections when stating the relation between heart-connectedness and hyper-connectedness.

Corollary 2
Let G be a weakly normal hypernet and $\mathcal{T}$ be a solution of G. For any subtree $\mathcal{T}'$ of $\mathcal{T}$, the sub-DG of G induced by $\mathcal{T}'$ is a (weakly normal) hypernet.

We are now ready to prove the main result of this subsection.

Theorem 9
If G is a satisfiable weakly normal hypernet, the necessary conditions of freeness (Theorem 7) are also sufficient.

Proof. Let $\mathcal{E}$ rooted at l be an ET satisfying the freeness conditions in Theorem 7. Among all the solutions of G, we pick a solution $\mathcal{T}$ in which the depth d of l is minimal. Using proof by contradiction, we show that $d = 0$, hence, l is the root of $\mathcal{T}$. Assume to the contrary that $d > 0$, meaning that there is some node l' outscoping l (Figure 15(a)). Without loss of generality assume that $\mathcal{E}$ has only exactly two holes (Left and Right). We show that at least one of the trees in Figures 15(b,c) is a solution of G, which means G has a solution in which the depth of l is smaller than d—a contradiction!

Figure 15(d) shows G with the two WCCs of $G - l$ called G_L and G_R. Note that all the nodes in $\mathcal{T}_2$ and $\mathcal{T}_3$ have to be in G_L and G_R, respectively (in the figure, G_2 and G_3 show subgraphs of G induced by $\mathcal{T}_2$ and $\mathcal{T}_3$). In order to show that $\mathcal{T}_b$ or $\mathcal{T}_c$ is a solution, we assume that $\mathcal{T}_b$ is not a solution and prove that $\mathcal{T}_c$ is. Because $\mathcal{T}_b$ is not a solution, there are some unsatisfied constraints in $\mathcal{T}_b$. However, there are not many constraints that are satisfied in $\mathcal{T}$ but not in $\mathcal{T}_b$ (e.g., any constraints between the two nodes in $\mathcal{T}_1$,

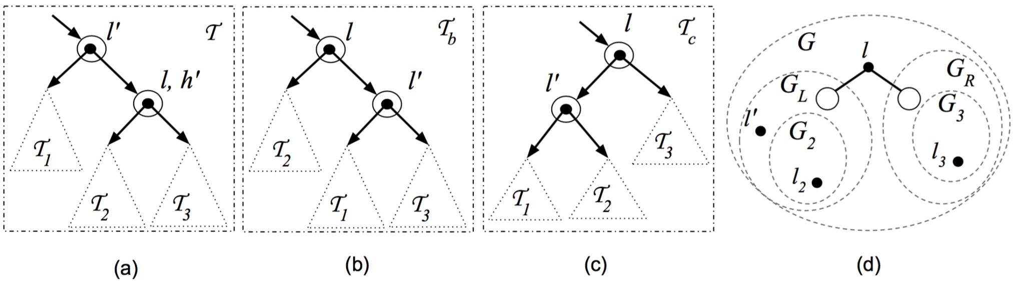


Figure 15
Proof of Theorem 9.

in $\mathcal{T}_2$, or in $\mathcal{T}_3$ are also satisfied in $\mathcal{T}_b$). In fact, an unsatisfied constraint can only be in the form of (l', l_2) or (h', l_2) with h' being the right hole of l' (the hole to which l is fused in $\mathcal{T}$), and l_2 being a label in $\mathcal{T}_2$ (it is true that a constraint from l' to l is not satisfied in $\mathcal{T}_b$ either, but such a constraint does not exist as l is a free node). Because l_2 is in G_L and there is a constraint between l' or h' and l_2, l' belongs to G_L as well.

Similar to $\mathcal{T}_b$, in order for $\mathcal{T}_c$ not to be a solution, there must be a constraint (l', v_3) or (h', v_3) violated, where v_3 is a node in $\mathcal{T}_3$, and hence, in G_R. But l' and h' are in G_L, so such constraints cannot exist. This means there cannot be any unsatisfied constraint in $\mathcal{T}_c$. This proves that T_c is a solution in which l has a lower depth. □

5.3 SAT and ENUM Algorithms

Following Theorems 8 and 9, Table 2 gives the algorithms for **SAT** and **ENUM**.

Theorem 10
ENUM and SAT are correct for all weakly normal hypernets.

Table 2
SAT and ENUM algorithms for weakly normal hypernets.

```
1: procedure SOLVE(Weakly normal hypernet $G$)
2:     If $G$ contains a single (label) node, return the single-node tree $\mathcal{T} = (\{l\}, \{\})$.
3:     Pick an ET $\mathcal{E}$ satisfying the freeness conditions in Theorem 7, otherwise, fail.
   ▷ For SAT, pick arbitrarily.
   ▷ For ENUM, pick iteratively.
4:     Let $l$ be the root and $m$ the number of holes of $\mathcal{E}$.
5:     Let $G_1, G_2, \ldots, G_m$ be WCCs of $G - \mathcal{E}$.
6:     Let $\mathcal{T}_i$ =SOLVE($G_i$) for $i = 1, \ldots, m$.
7:     Let $h_i$ be the hole of $\mathcal{E}$ connected to $G_i$ in $G - l$, for $i = 1, \ldots, m$.
           If $\mathcal{E}$ has an open hole $h_k$, let $G_k$ be the WCC not connected to any hole of $\mathcal{E}$.
8:     Build $\mathcal{T}$ by plugging the root of $\mathcal{T}_i$ into $h_i$, for $i = 1, \ldots, m$.
9:     return $\mathcal{T}$.
```

Proof. Using Theorem 8 and induction on the depth of recursion, it is easy to see that if ENUM or SAT return a tree $\mathcal{T}$, $\mathcal{T}$ is a solution of G. This proves that ENUM and SAT are sound.

An inductive proof is used to prove the completeness as well. By completeness, we mean that any solution of G is *arbitrarily/iteratively* generated by SAT/ENUM. Let $\mathcal{T}$ be an arbitrary solution of G rooted at the ET $\mathcal{E}$, and $\mathcal{T}_1, \ldots, \mathcal{T}_n$ be the subtrees of $\mathcal{T}$ under the holes $h_1, \ldots, h_n$ of $\mathcal{E}$. Following Lemma 6 (Section 5.1), in order for $\mathcal{T}$ to be a solution of G, the subtrees $\mathcal{T}_1, \ldots, \mathcal{T}_n$ must be the solutions to $G_1, \ldots, G_n$, the WCCs of $G - \mathcal{E}$ connected to $h_1, \ldots, h_n$ in $G - l$, respectively (if $\mathcal{E}$ has an open hole h_k, G_k will be the WCC not connected to any hole of $\mathcal{E}$ in $G - l$). Based on the induction assumption, $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are arbitrarily/iteratively generated by SOLVE(). Therefore, $\mathcal{T}$ is also arbitrarily/iteratively generated. □

The running time of the algorithms is proportional to the number of recursive calls to SOLVE(), that is, $O(|L|)$. At each call, it takes $O(|G|)$ to find the set of free ETs (Bodirsky et al. 2004) and also to compute $G - \mathcal{E}$ for some free ET $\mathcal{E}$, where $|G| =_{def} |L| + |H| + |E| + |D|$. Therefore, SAT (as well as each branch of ENUM) runs in $O(|L||G|)$ time, meaning that the worst-case time-complexity of SAT (and each branch of ENUM) is quadratic in the size of G.

This result is for weakly normal hypernets—that is, when there are no label-to-hole constraints. Kallmeyer and Romero (2008) demonstrate cases where having such constraints is meaningful. The function of label-to-hole constraints is quite counter-intuitive. Consider the two ETs $\mathcal{E}$ and $\mathcal{E}'$ rooted at l and l' and containing holes h and h', respectively. A label-to-hole constraint (l', h) results in two types of solutions: those in which l' dominates l, and those in which l' is plugged into h. The algorithm presented in Table 2 may be easily modified to solve any hypernet by taking care of label-to-hole constraints after a free ET $\mathcal{E}$ is picked. At step 3.1, every label-to-hole constraint (l, h') may be replaced with (l, l'). Every (l', h) is taken care of by plugging l' into h. This latter step results in a stacked ET, but we know how to convert this into a standard ET without affecting the set of solutions (Theorem 2). Note that, here, we have to recursively take care of the incoming constraints to the holes of $\mathcal{E}'$. Koller and Thater (2007) prove that adding such a step to the algorithm increases the complexity from quadratic to cubic, when label-to-hole constraints exist. Following this discussion, we have:

Lemma 8
Every hypernet can be solved in polynomial time.

5.3.1 Weighted Constraints. Although we focus on hard constraints on UG solutions in this article, our findings also have implications for statistical disambiguation systems based on soft, or weighted, constraints. Hard constraints are equivalent to soft constraints with infinite weights, so NP-completeness results for hard constraints also apply to systems with weighted constraints. By defining a framework that is tractable with hard constraints, we open the theoretical possibility of efficient algorithms for finding the highest scoring solution according to a set of weighted constraints. We now briefly discuss two possible approaches to this problem.

Table 3
Viterbi algorithm for weakly normal hypernets. We use a function *sig* to determine the dynamic programming signature of a solution, an array δ to maintain the best scores found for partial solutions, and an array *best* to maintain the best partial solutions themselves. Arrays δ and *best* are indexed by a subgraph G and a signature. We use $best[G]$ to denote the set of all the best solutions found for G, regardless of their signature.

```
1: procedure VITERBI(Weakly normal hypernet $G$)
2:     $\delta[G] \leftarrow -\infty$                                ▷ Initialize for all signatures
3:     $best[G] \leftarrow \emptyset$
4:     if $G$ contains a single (label) node then
5:         $\mathcal{T} = (\{l\}, \{\})$
6:         $\delta[G, sig(\mathcal{T})] \leftarrow$score$(\mathcal{T})$
7:         $best[G, sig(\mathcal{T})] \leftarrow \mathcal{T}$
8:         return
9:     for each ET $\mathcal{E}$ satisfying the freeness conditions in Theorem 7 do
10:        Let $l$ be the root and $m$ the number of holes of $\mathcal{E}$.
11:        Let $G_1, G_2, \ldots, G_m$ be WCCs of $G - \mathcal{E}$.
12:        For $i = 1, \ldots, m$
13:        if not defined $best(G_i)$ then
14:            VITERBI$(G_i)$
15:        for $(\mathcal{T}_1, \ldots, \mathcal{T}_m) \in best[G_1] \times \cdots \times best[G_m]$ do
16:            Let $h_i$ be the hole of $\mathcal{E}$ connected to $G_i$ in $G - l$, for $i = 1, \ldots, m$.
17:            If $\mathcal{E}$ has an open hole $h_k$, let $G_k$ be the WCC not connected to any hole of
    $\mathcal{E}$.
18:            Build $\mathcal{T}$ by plugging the root of $\mathcal{T}_i$ into $h_i$, for $i = 1, \ldots, m$.
19:            if score$(\mathcal{T}) > \delta[G, sig(\mathcal{T})]$ then
20:                $\delta[G, sig(\mathcal{T})] \leftarrow$score$(\mathcal{T})$
21:                $best[G, sig(\mathcal{T})] \leftarrow \mathcal{T}$
```

The first approach is simply to use the ENUM algorithm as the framework for search in statistical scope disambiguation systems. We can use ENUM to list solutions meeting the hard constraints and score each solution individually. This rescoring approach may take exponential time in the worst case, but may be an effective algorithm when the hard constraints leave a tractable number of solutions in practice. This approach has the advantage that it can take advantage of arbitrary weighted constraints, also known as features, in scoring candidate solutions.

The second approach is to use dynamic programming to score candidate UGs. In this approach, we use the recursive ENUM algorithm as a framework for scoring partial solutions to the input UG, as shown in the algorithm of Table 3. Dynamic programming requires that the features used to score solutions must be local, that is, that they must be functions of some local subgraph of the solution. We refer to this local subgraph as the **signature** of a solution, and use it to index the dynamic programming chart.

As an example of applying the ENUM together with dynamic programming, suppose that the weighted features of the statistical system are restricted to consider only an ET and the hole into which it is plugged. In this case, we can construct a dynamic programming chart indexed by the subset of ETs covered by a partial solution, and the identity of the ET at the root of the partial solution. The size of this chart is $O(n2^n)$ in the number of ETs—still exponential in the worst case. However, as with the more general features, the ENUM algorithm, memoizing recursive calls with the chart, can take advantage of the hard constraints of the UG to lower the time complexity in practice. More generally, we can define a dynamic programming chart indexed by whatever features of the partial solution are distinct to the statistical model, and memoize recursive calls to ENUM according to this dynamic programming structure.

More efficient algorithms for finding the highest scoring solution may be possible, and are an important area for future work. Our polynomial-time SAT algorithm does not translate directly into a Viterbi algorithm that is polynomial in the size of the input UG. However, we emphasize that for any NP-hard problem with hard constraints, the weighted generalization is also NP-hard. Therefore, identifying a subset of UGs that is tractable with hard constraints makes it possible to attempt to find efficient algorithms for weighted constraints.

5.4 Coherence and Tractability

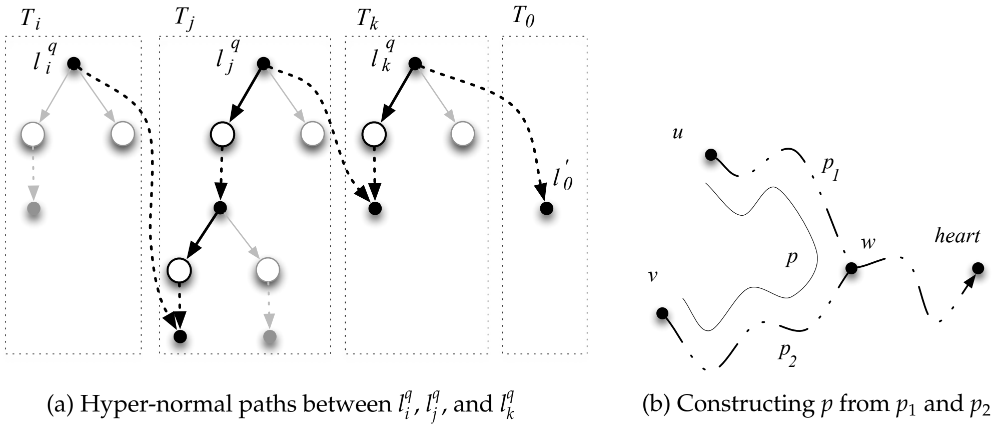
In this section, we show that coherent UGs are a subset of hypernets. Remember that the key notion for defining hypernets is hyper-normal connectedness and for coherence is heart-connectedness. Therefore, as the first step, we demonstrate how heart-connectedness relates to hyper-normal connectedness.

Theorem 11
Let U be a CF-UG. If i is connected to j through a directed path in SDG_U, then l_i^q and l_j^q are hyper-normally connected in G_U.

Proof. First assume that i is directly connected to j through the edge (i, j). This means that there is a dominance constraint (l_i^q, u) in U for some node u of T_j^r. Being the root of T_j, l_j^q reaches u by a directed, and hence hyper-normal, path p. The path $p_{i,j}$ between l_i^q and l_j^q, which is the concatenation of p and (l_i^q, u), is therefore hyper-normal.

Now assume that i is connected to j through a directed path $\overrightarrow{d}$. Following the previous result, l_i^q is connected to l_j^q in G_U through a path p, which is the concatenation of hyper-normal paths corresponding to the edges of $\overrightarrow{d}$. Because, at each point of concatenation, one of the two edges is a solid edge (Figure 16(a)), the concatenation remains hyper-normal. □

This theorem proves that heart-connectedness subsumes hyper-normal connectedness for the following reason. Given any two arbitrary nodes u and v belonging to trees T_i and T_j, both i and j can reach the heart through some directed path in SDG_U. Therefore, l_i^q and l_j^q are hyper-normally connected to the heart through hyper-normal paths p_1 and p_2, as shown in Figure 16(b). Concatenating p_1 and p_2 while taking out the common part, we can construct a hyper-normal path p connecting u and v.



(a) Hyper-normal paths between l_i^q, l_j^q, and l_k^q (b) Constructing p from p_1 and p_2

Figure 16
The relation between heart-connectedness and hyper-normal connectedness.

Corollary 3
If U is a heart-connected CF-UG, G_U is hyper-normally connected.
We now have the tools to prove our main theorem.

Theorem 12
For every coherent UG U, G_U is a hypernet, and hence, tractable.

Proof. U is coherent, hence according to Theorem 6, U has a heart-connected canonical form sub-UG U'. In the following proof, we show that U' is a hypernet. Because hypernets are closed under the increment of constraint edges, it follows that U is a hypernet. Let $\mathcal{E}$ be an arbitrary ET in $G_{U'}$ rooted at l.

T12a. (Proof of condition D26a) We need to show that $\mathcal{E}$ has at most one open hole.

$\mathcal{E}$ is of one of the following three types:

- Floating scopal. Because U' is complete, the only possibly open hole of $\mathcal{E}$ is its body hole. Since U' is coherent, the root of $\mathcal{E}_q$ must also be closed.
- Fixed scopal. Because U' is complete, $\mathcal{E}$ has no open hole, hence, the only potentially open node is the root of $\mathcal{E}$.
- Non-scopal. $\mathcal{E}$ has a single label node, hence, its only potentially open node.

T12b. (Proof of condition D26b) We need to prove the following statement. If l_1 and l_2 are two dominance children of a hole h of $\mathcal{E}$, l_1 and l_2 are hyper-normally connected in $G'_U - h$. This statement is vacuously true, because no hole of G'_U can have two or more dominance children, that is, the antecedent never holds (this is because, in a CF-UG, the body hole of quantifiers have no outgoing constraint and every other hole has exactly one).

T12c. (Proof of condition D26c) First consider the case where $\mathcal{E}$ has no open hole. Because U' is a CF-UG, $\mathcal{E}$ must be either a fixed scopal or a non-scopal ET. In either case, l has no dominance children in G'_U, hence, the statement is vacuously true. Now consider the case where $\mathcal{E}$ has an open hole. Because U' is a CF-UG, $\mathcal{E}$ must be a floating scopal ET. Therefore, l has no incoming dominance edge. Let u, v be the dominance children of l not connected to a hole of $\mathcal{E}$. Following Theorem 3, u and v are connected together through a hyper-normal path p. p cannot visit l, otherwise u or v are hyper-normally connected to a hole of $\mathcal{E}$ (l has no incoming dominance edge, hence, p must go through a hole of $\mathcal{E}$), which is in contradiction with how we chose u and v in the first place. This proves that p is a path in $G{-}l$, hence u and v are hyper-normally connected in $G{-}l$. □

This theorem is the main result of our article. Given that coherence seems to be a natural condition on the semantic interpretation of any sentence, this result implies that it is tractable to resolve the quantifier scope of any sentence. Formally proving that a semantic interpretation system always yields coherent representations will depend on the specifics of the syntax/semantics interface. In Section 7, we will show that the syntax semantics interfaces specified by MRS and HS do in fact always yield a coherent representation.

6. Quantifier Scoping Is an Ordering Problem

Traditionally, scoping has been treated as an ordering problem. For example, all statistical scope disambiguation systems define scoping as learning to predict an order (Higgins and Sadock 2003; Srinivasan and Yates 2009; Manshadi, Gildea, and Allen 2013). For this reason, $n!$ has always been considered as *the* upper bound on the number of readings of a sentence with n quantifiers. This is in contrast with the frameworks discussed in this article, in which scoping means predicting a tree structure. A UG with n quantifiers has $2n$ holes, hence $(2n)!$ fusings can be built (assuming that the solutions are merging-free). Some of these fusings can be filtered out, if we take qeq relations into account. For example, given the following definition of first-order CF-UG, a first-order CF-UG with two quantifiers has four potential solutions, as shown in Figure 17.

Definition 28 (First Order UG)
A UG U with no fixed-scopal ETs is called a **first-order** UG.

It is easy to calculate this number for n quantifiers.

Lemma 9
If U is a first-order CF-UG with n quantifiers , then U^q has exactly $2^{n-1}n!$ solutions.

The factor 2^{n-1} results from the fact that once Q_1 is decided to be in the scope of Q_2, there are two possible configurations based on whether Q_1 is in the scope of the

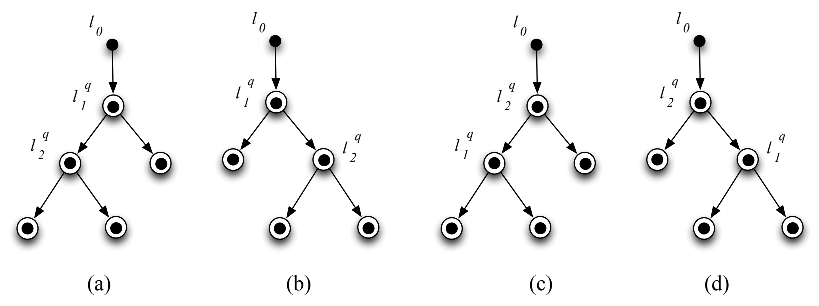


Figure 17
All potential solutions for first-order CF-UG with two quantifiers.

restriction or the body of Q_2. It seems that in the past it has been taken for granted that every permutation of quantifiers uniquely imposes a tree structure. One idea is to filter some of these trees by taking dominance constraints into account, although it should be noted that, unlike qeq constraints, the distribution of dominance constraints is not predefined. The good news is that, using the machinery provided and the results obtained in the last section, we can prove this conjecture. That is, although the exact distribution of dominance constraints is not given, the minimum connectivity required by hypernets guarantees this property. Next, we use the notion of *tree order*, which is the order in which nodes of a tree are visited and depends on the traversal method (e.g., in pre-order traversal, first the branches are visited, then the node itself, while the post order is the opposite). For this theorem, it does not matter which order is picked, but once picked, we should stick to it.

Theorem 13
Let G be a hypernet, and π be an arbitrary permutation of ETs. There exists at most one solution $\mathcal{T}$ of G with π being a tree order of ETs in T.

Proof. We prove this using induction on e the number of ETs in G. For $e = 1$, this is trivial. Assume that it holds for $e = k$ ($k > 0$) and let G be a DG with $k + 1$ ETs. If G has no solution, whose tree order of ETs is π, we are done. Otherwise, $\mathcal{E}$ (the head of π), is a free ET. Based on Theorem 7, $G - \mathcal{E}$ has exactly m WCCs $G_1, \ldots, G_m$, where m is the number of holes in $\mathcal{E}$. Based on the induction assumption, each sub-DG G_i has exactly one solution $\mathcal{E}_i$ whose tree order of ETs matches π. In addition, because G is a hypernet, following the third condition of Definition 26 (D26c), there is a unique hole of $\mathcal{E}$ into which the root of each $\mathcal{E}_i$ can be plugged. Therefore, G has a unique solution whose tree order of ETs is π. □

This theorem proves that if the underspecified representation is coherent, then quantifier scoping is indeed an ordering problem.

7. Comparison with Other Formalisms

In this section, we compare our work with the three frameworks it was built upon. We show that our definition of coherence extends the set of tractable representations previously identified by the Dominance Graph framework. We further show that the semantic representations returned by the syntax/semantics interfaces specified by Minimal Recursion Semantics and Hole Semantics are always coherent, and hence tractable. Our strategy in this section will be to show subsumption relations among classes of underspecification graphs, as shown in Figure 1. We say that one class is **subsumed** by another if it is a subset of the second class, or if there exists a (generally very simple) syntactic transformation from elements of the first class to elements of the second class such that the set of solutions remains the same. In either case, any algorithm for finding solutions for the second class can be applied to graphs from the first class, which is the property that we will use to establish tractability.

7.1 Dominance Graphs

Dominance Graphs[18] and Underspecification Graphs[19] are in fact equivalent when the underspecified representation is complete. The only difference between the two is that UG incorporates qeq relations in addition to dominance constraints, but, as we have shown before, the two constraints become equivalent for complete UGs. The main reason for us to define UG has been to provide a foundation to prove this equivalence.[20]

There is also a subtle difference in the terminology between the two frameworks. What we have defined as *solution* in our framework corresponds to the notion of *configuration* in Dominance Graphs framework. Therefore, the satisfiability (or enumeration) algorithm presented in this article, in Dominance Graphs terminology, decides whether a DG has a configuration (or enumerates all possible configurations of a DG).

A major contribution of this paper was to extend the coverage of the previously known tractable subset of DG, called weak net (Niehren and Thater 2003). We achieved this goal by defining the notion of hypernet, motivated by and built upon weak net. In the following, we give the definition of weak net from Thater (2007), translated into our own terminology.

Definition 29 (Weak Net)
A weakly normal UG U is a **weak net** if and only if for every elementary tree $\mathcal{E}$:

D29a. $\mathcal{E}$ has exactly one open node.

D29b. If l_1, l_2 are two dominance children of a node u of $\mathcal{E}$, then l_1 and l_2 are hyper-normally connected in $U - u$.

18 Dominance Graphs were derived from Dominance Constraints. Dominance Constraints were originally developed as a broad framework for representing arbitrary tree structures. Dominance Graphs is a revision of Dominance Constraints, which is better suited for modeling scope underspecification. For details on how Dominance Graphs compare to Dominance Constraints, see Thater (2007).

19 When capitalized, the terms "Dominance Constraints," "Dominance Graphs," and "Underspecification Graphs" refer to the frameworks; otherwise, and when abbreviated, they refer to a notion within the framework.

20 Another advantage could be the ability to model an incomplete underspecified representation, for example, when dealing with fragments, broken parse trees, and so forth. The latter, however, is not the subject of this article.

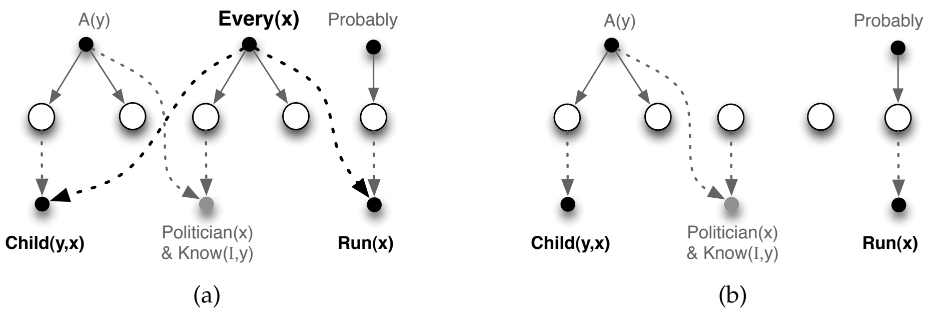


Figure 18
A coherent UG which belongs to hypernet but is not a weak net, because the dominance children of the quantifier *E*very, labeled $Run(x)$ and $Child(x, y)$, are not hyper-normally connected, if the quantifier node is removed from the graph as in (b).

Comparing the definition of hypernet (Definition 26) with weak net, it is clear that weak net is a subset of hypernet. By giving examples of natural language sentences whose underspecified representation is a hypernet but not a weak net, we show that weak net is a proper subset of hypernet. Consider G_U in Figure 18(a), where U is the CF-UG for the following sentence.

3. *Every politician, whom I know a child of, probably runs.*

Let $\mathcal{E}$ be the ET for the quantifier *Every* and l be the root of $\mathcal{E}$. The two dominance children of l are not (hyper-normally) connected in $G_U - l$, as shown in Figure 18(b). Therefore, G_U is not a weak net. What makes this example special is that SDG_U has a loop. Weak net misses a whole class of sentences with similar structures. Because we have proved that hypernet covers all coherent UGs, it should not be surprising that G_U is a hypernet, and hence, covered by our framework. This is because hypernet does not require all the dominance children of the quantifier node u to be hyper-normally connected in $G - u$, but only those that are not hyper-normally connected to a hole of u (Condition D26c). Therefore, for this UG, since the node labeled $Child(x, y)$ is hyper-normally connected to a hole of u in $G - u$, it is not required to be hyper-normally connected to the node labeled $Run(x)$ in $G - u$.

7.2 Minimal Recursion Semantics

As the official semantic language of the LinGo English Resource Grammar (Copestake and Flickinger 2000), Minimal Recursion Semantics has been used relatively widely in practice. As mentioned before, UG has been defined in a way that it subsumes both DG and MRS. Although MRS is not originally defined in graph terms, a limited version of UG can serve as the notational variant of MRS. The limitation is on the usage of dominance constraints. MRS only implicitly uses dominance constraints, and that is to satisfy binding constraints, hence, only realizes dominance constraints that go from a quantifier node to a non-quantifier label node. By restricting UG to only allow those dominance constraints, we can obtain a notational variant of MRS in graphical form.

The heart of MRS is the notion of Elementary Predication or EP, a labeled predication over a set of first order variables $x, y, \ldots$ and second order variables $h_1, h_2, \ldots$

$$l : P(x, y, \ldots, h_1, h_2, \ldots) \tag{6}$$

We have defined ET as the notational variant of EP. Similarly, the definitions of the three types of ET—non scopal, fixed scopal, and floating scopal—have been inspired by their EP counterparts. MRS does not notation-wise distinguish between holes and labels. They are both referred to as **handle** and are represented by the letter h, often subscripted with a number. In defining UG, we followed, for mathematical convenience, the tradition of Hole Semantics and DG and distinguished between the two terms: A handle that is the label of an EP is called a label, and the top handle (see below) and any handle that is an argument of an EP is called a hole. The motivating example given at the beginning of Section 2 was nothing but an MRS structure as a graph (Figure 2(d)). Equation (7) represents the MRS of the same sentence in the original MRS language as defined by Copestake, Lascarides, and Flickinger (2001).

$$\begin{gathered}\langle h0, \{h1 = Every(x, h2, h3),\ h4 = Child(x), h5 : A(y, h6, h7),\ h8 : Politician(y), \\ h9 : Of(x, y),\ h10 : Run(x)\},\ \{h0 =_q h10,\ h2 =_q h4,\ h6 =_q h8\}\rangle\end{gathered} \tag{7}$$

The handle represented as $h0$ is called (global) **top handle** and plays the role of the top ET in UG. It is equivalent to having a dummy EP $l0 : h0$ as the top but then removing the label node and leaving a floating hole. An MRS is defined as the triple $\langle GT, R, C\rangle$, where GT is the top handle, R is the set of EPs, and C is the set of qeq constraints. As mentioned before, the implicit binding constraints encoded in the first order variables x, y of the MRS are made explicit in UG through dominance constraints. Given an MRS, a **scope-resolved structure** (equivalent to the notion of solution in UG) is built by equating handles.

$$h0 = h1,\ h2 = h5,\ h3 = h10,\ h6 = h8,\ h7 = h4,\ h9 = h4 \tag{8}$$

Note that $h9 = h4$ in Equation (8) equates the label of two EPs, resulting in the EP conjunction $Child(x) \wedge Of(x, y)$. UG allows EP conjunctions in two different ways: (1) by defining fusion as a many-to-one mapping from labels to holes (e.g., fusing both $h4$ and $h9$ to $h7$); (2) Using stacked ETs. The second approach models the case where EP conjunctions are built during the semantic composition and before any scope resolution (something that can happen in MRS). Copestake et al. (2005) define a semantic composition process, presenting an algorithm on building MRS structures from constituency trees. This algorithm has motivated our notion of canonical form. By following this algorithm, Manshadi, Allen, and Swift (2008a) proved the following theorem.

Theorem 14
Every output of MRS's semantic composition process (as spelled out by Copestake et al. [2005]) is either in canonical form or trivially unsatisfiable.

The detailed proof of this theorem can be found in Manshadi, Allen, and Swift (2008a). Given that the result of the MRS semantic composition process is in canonical form, we can use the tools developed in Section 5 to show that it is also tractable.

Theorem 15
Every output M of the semantic composition algorithm on a coherent sentence is either trivially unsatisfiable or a coherent UG, and therefore tractable.

Proof. By Theorem 14, every output M of the semantic composition algorithm on a coherent sentence is either trivially unsatisfiable or in canonical form. By Definition 14, a canonical form UG is complete. The interpretation of a coherent sentence must be heart-connected, and by Theorem 6, a heart-connected, complete UG is a coherent UG. By Theorem 12, a coherent UG is a hypernet, and, by Lemma 8, a hypernet is tractable.□

7.3 Hole Semantics

Corresponding to the concept of UG in our framework, Hole Semantics defines the notion of **UR** (**Underspecified Representation**). UR[21] is a subset of normal DGs, therefore it allows for only one type of constraint, hole-to-label dominance constraints. Analogous to the notion of coherence, which was motivated by the canonical form of the output of MRS syntax/semantic interface, URs generated by the syntax/semantic interface, as demonstrated by Koller, Niehren, and Thater (2003), conform to the following two properties: (1) they are leaf-labeled (they call a DG with no open holes leaf-labeled); (2) they are hyper-normally connected. Therefore, Hole Semantics UR equals the set of all hyper-normally connected leaf-labeled normal DGs.

Now, we show that all satisfiable URs are hypernets.

Lemma 10
Let a hyper-normally connected normal DG G, with a solution T be given. If T' is a subtree of T, then G', the sub-DG of G induced by T' is also hyper-normally connected.

Proof. Assume to the contrary that G' is not hyper-normally connected. Consider the two nodes of G' that are not hyper-normally connected in G', and let P be the hyper-normal path that connects these two nodes in G. We prove that l, the root of T', is dominated by two mutually exclusive nodes, which is a contradiction.

Without loss of generality, we assume that both nodes are label nodes (l_1 and l_2) and that P contains no other node in G'.[22]

Because G is normal, the two nodes immediately after l_1 and l_2 in P have to be hole nodes h_1 and h_2 with outgoing constraint edges to l_1 and l_2, respectively (see Figure 19). Note that $h_1 \neq h_2$, otherwise, P is not hyper-normal. The path between h_1 and h_2 cannot be a directed path, because no solid edge leaves a hole, and no constraint edge leaves h_1 or h_2 either (otherwise, P would not be a hypernormal path). Therefore, there are

21 Throughout this subsection, by UR we mean Hole Semantics's UR.
22 Among all pairs of node not hyper-normally connected in G', consider the pair for which the length of P is the shortest.

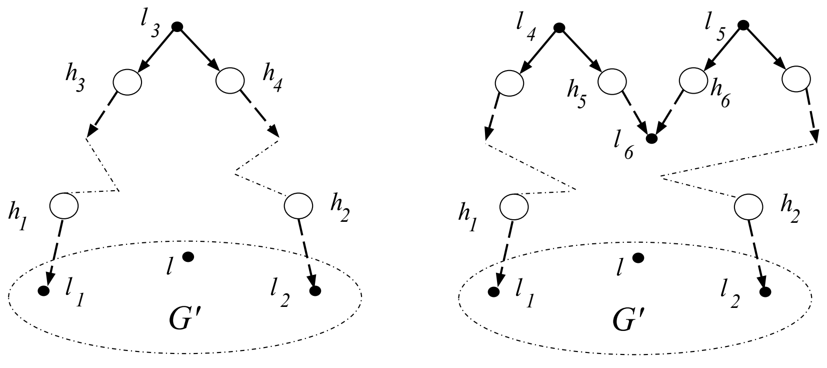


Figure 19
Proof of Lemma 10.

two back-to-back solid edges emanating from a node l_3 on P, as shown in the figure on the left.

We consider two options:

1. The segments of P between h_3 and l_1 and between h_4 and l_2 are both directed, hence, l_1 and l_2 are under the scope of h_3 and h_4, respectively, which means l is under the scope of both h_3 and h_4. Contradiction!
2. One of the two parts, say the segment of P between h_4 and l_2, is not directed. Therefore, P has a segment like the one in the figure on the right. Two possible cases may happen in T: h_5 is in the scope of h_6 or vice versa. In either case l_1 and l_2, and hence l, are under the scope of two mutually exclusive holes. Contradiction! □

Theorem 16
Every satisfiable hyper-normally connected leaf-labeled normal DG is a hypernet.

Proof. Consider a normal DG G, an arbitrary ET $\mathcal{E}$ rooted at l, and a solution $\mathcal{T}$ of G. We have to prove that the three conditions in Definition 26 hold. Condition D26a directly follows from leaf-labeledness, which guarantees that there is no open hole in a UR.[23] To prove condition D26b, consider l_1 and l_2, the two dominance children of a hole h of $\mathcal{E}$, and assume to the contrary that l_1 and l_2 are not hyper-normally connected in $G - h$. Because G is satisfiable, let $\mathcal{T}$ be a solution of G, $\mathcal{T}'$ be the subtree of $\mathcal{T}$ rooted at l, and G'

23 In fact, since we are only considering satisfiable URs, we do not need to presume leaf-labeledness, because it follows from hyper-normal connectedness. Assume to the contrary that this is not the case, and h is the open hole of $\mathcal{E}$. Let l' be the label plugged into h in $\mathcal{T}$. Because G is hyper-normally connected, l' is hyper-normally connected to l, and hence to some hole h' of $\mathcal{E}$ in $G^l - l$, where G^l is the sub-DG of G induced by the subtree of $\mathcal{T}$ rooted at l. Clearly, $h' \neq h$, because h is open, but h' is not, hence, according to Lemma 7, l' must be dominated by h' in $\mathcal{T}$, meaning that l' is dominated by two distinct holes of $\mathcal{E}$; contradiction!

be the sub-DG of G induced by $\mathcal{T}'$. l_1 and l_2 are hyper-normally connected in G, and, by Lemma 10, are hyper-normally connected in G', but are not hyper-normally connected in $G' - h$. This means that the hyper-normal path p connecting the two nodes passes h. Because l is the root of solution $\mathcal{T}'$, it does not have an incoming dominance edge, therefore any path that connects l_1 and l_2 and passes h must also pass another hole h' of $\mathcal{E}$. Let l_1 be the node that connects to h' by a path P without first going through l. Then according to Lemma 7, there is a node u on P such that all the nodes on P are dominated by u in T'. The only nodes dominating h' in T' are h' itself and l, but l is not on P, hence, $u = h'$. This means that one of the nodes l_1 or l_2 must be in the scope of h', but we know that they both are also in the scope of h. This is a contradiction! Condition D26c is vacuously true because $\mathcal{E}$ has no open hole and l has no dominance children. □

This theorem shows that, practically speaking, UR is a subset of hypernet. An important question arising is, given that it does not allow for label-to-label constraints, is Hole Semantics powerful enough to cover all coherent UGs? In other words, given that some constraints, such as the binding constraints, are inherently label to label, how does Hole Semantics get away without it? This question has remained unanswered in the past. Once again, we use our notion of coherence to answer this question. We show that, in coherent UGs, those label-to-label dominance constraints that implement variable binding may be replaced with hole-to-label constraints without affecting the set of solutions. First, we define some terminology.

Definition 30 (Ancestor/Disjoint)
Consider a complete UG U, $SDG_U = (V, E)$, and a node $i \in V$.

- $Anc(i)$ is the set of nodes that reaches i (through a directed path) in SDG_U (including i itself);
- $Dis(i)$ is the set of nodes that reaches the *heart* (through a directed path) without going through i (including the heart itself).

For example, for the SDG in Figure 10, $Anc(2) = \{2, 3\}, Dis(2) = \{0, 1\}$. If U is coherent, SDG_U is *heart-connected*, hence the following lemma directly follows from the definition.

Lemma 11
Given a coherent UG U and $SDG_U = (V, E)$, for every i, j in V:

i. $Anc(i) \cup Dis(i) = V$.

ii. $i \notin Dis(j) \Rightarrow j \in Dis(i)$.

Theorem 17
Let U be a coherent UG, and $\mathcal{T}$ a solution of U, and consider the quantifiers Q_i and Q_j, where l_i^q outscopes l_j^q in $\mathcal{T}$.

- if $j \in Anc(i)$, l_j^q is under the *restriction* of l_i^q in $\mathcal{T}$;
- if $j \in Dis(i)$, l_j^q is under the *body* of l_i^q in $\mathcal{T}$.

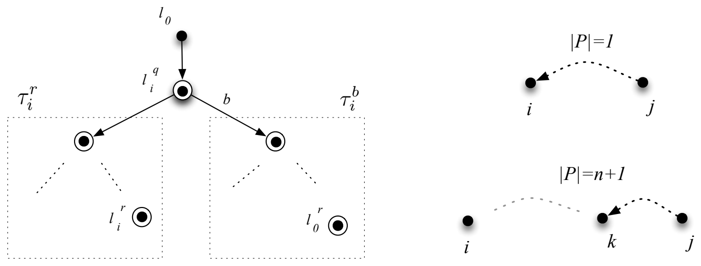


Figure 20
Proof of Theorem 17 (l_k^r represents an arbitrary node in T_k^r).

Proof. Let $j \in Anc(i)$, and τ_i^r and τ_i^b be the subtrees of $\mathcal{T}$ rooted at the restriction and body hole of l_i^q respectively, as shown in Figure 20. We use induction on $|P|$, the length of the path P from j to i in SDG_U, to show that l_j^q is in the restriction of l_i^q. For $|P| = 1$, j immediately dominates i in SDG_U, hence there is a dominance edge from l_j^q to some node $l_i^r \in T_i^r$, where T_i^r is the restriction tree of l_i^q (Definition 13). Therefore, l_i^r must be in the scope of l_j^q in $\mathcal{T}$. Since l_i^r is in τ_i^r, l_j^q must also be in τ_i^r.

Now let $|P| = n + 1$ ($n \geq 1$) and k be the node immediately after j on P. Because j immediately dominates k in SDG_U, there is a dominance edge from l_j^q to some node $l_k^r \in T_k^r$, which means l_k^r must be in the scope of l_j^q. According to the induction assumption, l_k^q, and hence l_i^r, is in τ_i^r. Therefore, l_j^q must also be in τ_i^r.

A similar argument applies for the second part, that is, when $j \in Dis(i)$. □

Corollary 4
Let U be a coherent UG, and Q_i and Q_j two quantifiers in U. If $j \in Anc(i) \cap Dis(i)$, then there exists no solution of U in which l_i^q outscopes l_j^q.

Now, we are ready to prove our main theorem.

Theorem 18
In a coherent UG U, all label-to-label constraints emanating from quantifier label nodes to some node other than a quantifier label node may be replaced with hole-to-label constraints with the set of solutions remaining exactly the same.

Proof. Consider a coherent UG U, an arbitrary quantifier Q_i labelled at l_i^q, and an arbitrary dominance edge (l_i^q, u) emanating from Q_i. We need to show that either u is always (i.e., in *all* solutions of U) under the body of Q_i or always under its restriction. Because U is coherent, it has a canonical form sub-UG (Figure 7 in Section 3); therefore, $u = l_j^r$, where l_j^r is some node in the tree T_j^r. When $j = 0$, l_j^r belongs to the heart tree, hence u is always under the body of Q_i. Throughout the rest, we only consider the case where $j > 0$, that is, u is a node in the restriction tree of Q_j.

First, notice that because of the dominance edge (l_i^q, l_j^r), $SDG_U = (V, E)$ contains an edge from i to j. This means:

$$i \in Anc(j) \tag{9}$$

Now consider $Anc(i)$; depending on whether $j \in Anc(i)$ or not, we consider two possible cases, and prove that the premise holds in both cases.

Case i. $j \in Anc(i)$[24]
We further consider two situations depending on whether $i \in Dis(j)$ or not.

(a) $i \in Dis(j)$. Following Equation (9), $i \in Anc(j) \cap Dis(j)$, which means Q_j does not dominate Q_i in any solution of U (Corollary 4). Because Q_i dominates a node in the restriction tree of Q_j, it cannot be disjoint either, hence, Q_i outscopes Q_j in every solution of U. Since $j \in Anc(i)$, following Theorem 17, Q_j is under the restriction of Q_i, and so is u.

(b) $i \notin Dis(j)$. Following Lemma 11: $j \in Dis(i)$, hence $j \in Anc(i) \cap Dis(i)$, meaning that Q_j outscopes Q_i in every solution, from which and given that $i \in Anc(j)$, Q_i is under the restriction of Q_j. This means every node in the restriction tree of Q_j, including u, is under the body of Q_i.[25]

Case ii. $j \notin Anc(i)$

Unlike case i, here it is possible to simultaneously have both the solutions where Q_i outscopes Q_j and those where Q_j has the wide scope. Interestingly (and again unlike case i), in both types of solution, u is always under the body of l_i^q. We show this for each type separately.

(a) Q_j dominates Q_i in $\mathcal{T}$.
Given that $i \in Anc(j)$, Q_i is in the restriction of Q_j in $\mathcal{T}$, meaning that every node in T_j^r, including u, is under the scope of the body of Q_i.

(b) Q_i dominates Q_j in $\mathcal{T}$.
Because $j \notin Anc(i)$, $j \in Dis(i)$, hence, Q_j is in the body of Q_i, and so is u.

Here we showed that, for a coherent UG, if Q_i dominates some node u, it is impossible to simultaneously have solutions where u is in the body of Q_i and those where u is in the restriction of Q_j. This means that it is possible to build a UG U' by replacing (l_i^q, u) with either (h_i^b, u) or (h_i^r, u) while the set of solutions of U and U' is the same. $\square$

Following this theorem, Hole Semantics is able to enforce variable binding using hole-to-label dominance constraints. Figure 21 summarizes the expressive power of all

24 In this case, SDG_U has a loop, which means, as discussed in the previous section (see Figure 18), U is not a (weak) net. Therefore, as seen in the following, if it was not for the non-net cases, we could always replace these label-to-label constraints with constraints emanating from the *body* hole of the quantifiers.

25 In the proof of the equivalence of qeq and dominance constraints in complete UGs (Section 3, Theorem 5), we showed that given a solution $\mathcal{T}$ of a complete U, if Q_i is plugged in the restriction hole of Q_j, all the nodes $u \in T_j^r$ have to be under the scope of the body hole of Q_i.

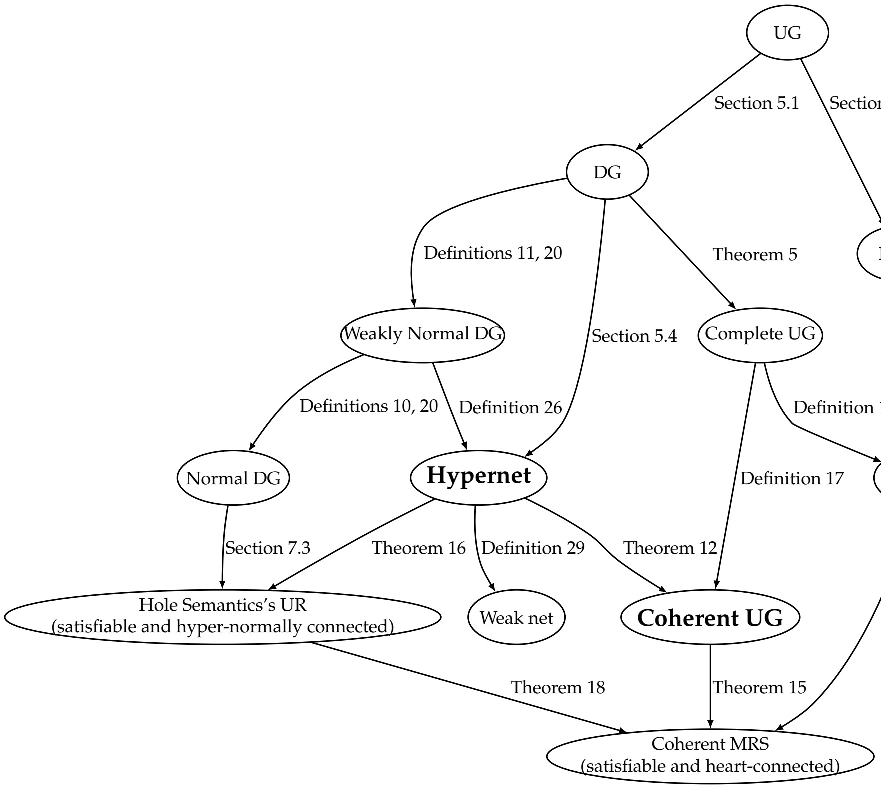


Figure 21
Summary of subsumption relations shown in this article.

proposed subsets. The outcome of this theorem is demonstrated by the edge from Hole Semantics's UR to coherent MRS.

8. Conclusion

We have solved several open questions within the context of the prevalent constraint-based scope underspecification frameworks: Minimal Recursion Semantics, Hole Semantics, and Dominance Constraints.

Although these frameworks are fundamentally different, there has been a conjecture that they become equivalent once restricted to the well-formed structures corresponding to actual URs of natural language sentences. Until now, the characterization of this well-formedness has been an open question, and so has the proof of the conjecture.

Ever since their satisfiability problem was proved to be intractable, there have been efforts on finding a tractable subset of the frameworks that is expressive enough to cover all well-formed structures. But even "(weak) net," the largest tractable subset previously found, leaves a group of natural language sentences uncovered.

Figure 21 summarizes this paper. At the top of this figure is UG, a framework we have defined in this paper to encompass all the rest of constraint-based frameworks shown here. In the heart of this figure, there are two subsets: coherent UG and hypernet.

Coherent UG is the relevant part of UG. It is, in fact, our characterization of what has been referred to in the past as well-formedness. We have linguistically justified that the complete UGs of all coherent sentences belong to this subset. Analogous to this linguistically motivated subset, we have the mathematically motivated notion of hypernet, defined to guarantee the mathematical and computational properties we have been looking for, but to be large enough to cover coherent UG. It is based on the notion of (weak) net from Dominance Graph but expanded to include all coherent UGs. Another central notion in this figure is CF, or canonical form, whose importance lies in the fact that, within this subset, qeq and normal dominance constraints become equivalent, and, hence, the main difference between UG (read MRS) and DG disappears.

There are several other properties of URs that have been taken for granted in the past. For example, although scoping in general is about predicting a tree structure, it has been treated as an ordering problem. We prove that for hypernet, and hence all coherent UGs, scoping is indeed an ordering problem. Finally, a framework like Hole Semantics has taken for granted that binding constraints, which are label-to-label in nature, may be modeled by hole-to-label constraints. Here, we have proved that for coherent UGs, label-to-label binding constraints may in fact be replaced with hole-to-label constraints, bridging another gap between MRS and Hole Semantics.

Acknowledgments
We are grateful to the anonymous reviewers for many suggestions improving this article.

References

Allen, James. 1995. *Natural Language Understanding (2nd ed.)*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA.

Alshawi, Hiyan and Richard Crouch. 1992. Monotonic semantic interpretation. In *Proceedings of the 30th Annual Conference of the Association for Computational Linguistics (ACL-92)*, pages 32–39, Newark, DE.

Althaus, Ernst, Denys Duchier, Alexander Koller, Kurt Mehlhorn, Joachim Niehren, and Sven Thiel. 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48(1): 194–219.

Bodirsky, Manuel, Denys Duchier, Joachim Niehren, and Sebastian Miele. 2004. A new algorithm for normal dominance constraints. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 59–67, Vancouver.

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

Bos, Johan. 2002. *Underspecification and Resolution in Discourse Semantics*. Ph.D. thesis. Saarland University.

Copestake, Ann and Dan Flickinger. 2000. An open source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of Language Resources and Evaluation Conference (LREC-2000)*, pages 591–600, Athens.

Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics—an introduction. *Research on Language and Computation*, 3:281–332.

Copestake, Ann, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Annual Conference of the Association for Computational Linguistics (ACL-01)*, pages 140–147, Toulouse.

Frege, Gottlob. 1923. "gedankengefüge" ["compound thought"]. *Beiträge zur Philosophie des Deutschen Idealismus*, III:36–51.

Fuchss, Ruth, Alexander Koller, Joachim Niehren, and Stefan Thater. 2004. Minimal recursion semantics as dominance constraints: Translation, evaluation, and analysis. In *Proceedings of the 42nd Annual Conference of the Association for*

Computational Linguistics (ACL-04), pages 247–254, Barcelona.

Higgins, Derrick and Jerrold M. Sadock. 2003. A machine learning approach to modeling scope preferences. *Computational Linguistics*, 29(1):73–96.

Hobbs, Jerry R. and Stuart M. Shieber. 1987. An algorithm for generating quantifier scopings. *Computational Linguistics*, 13(1-2):47–63.

Kallmeyer, Laura and Maribel Romero. 2008. Scope and situation binding in LTAG using semantic unification. *Research on Language and Computation*, 6(1):3–52.

Kamp, Hans. 1981. A theory of truth and semantic representation. In P. Portner and B. H. Partee, editors, *Formal Semantics—the Essential Readings*. Blackwell. pages 189–222.

Koller, Alexander, Joachim Niehren, and Stefan Thater. 2003. Bridging the gap between underspecification formalisms: Hole semantics as dominance constraints. In *10th Conference of the European Chapter of the Association for Computational Linguistics (EACL-03)*, pages 195–202, Budapest.

Koller, Alexander and Stefan Thater. 2007. Solving unrestricted dominance graphs. In *Proceedings of the 12th Conference on Formal Grammar*, pages 1–12, Dublin.

Manshadi, Mehdi and James Allen. 2012. Expanding the range of tractable scope-underspecified semantic representations. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics*, pages 142–150, Montréal.

Manshadi, Mehdi, James Allen, and Mary Swift. 2008a. Towards a universal underspecified semantic representation. In *Proceedings of the 13th International Conference on Formal Grammar*, Hamburg.

Manshadi, Mehdi, Daniel Gildea, and James F. Allen. 2013. Plurality, negation, and quantification: Towards comprehensive quantifier scope disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*, pages 64–72, Sofia.

Manshadi, Mehdi H., James F. Allen, and Mary Swift. 2008b. Toward a universal underspecified semantic representation. In *Proceedings of the 13th Conference on Formal Grammar (FG)*, pages 77–94, Hamburg.

Manshadi, Mehdi H., James F. Allen, and Mary Swift. 2009. An efficient enumeration algorithm for canonical form underspecified semantic representations. In *Proceedings of the 14th Conference on Formal Grammar (FG)*, pages 85–101, Bordeaux.

Niehren, Joachim and Stefan Thater. 2003. Bridging the gap between underspecification formalisms: Minimal recursion semantics as dominance constraints. In *Proceedings of the 41th Annual Conference of the Association for Computational Linguistics (ACL-03)*, pages 367–374, Sapporo.

Reyle, Uwe. 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 10(2):123–179.

Schubert, Lenhart K. and Francis Jeffry Pelletier. 1982. From English to logic: Context-free computation of "conventional" logical translation. *Computational Linguistics*, 8(1):26–44.

Srinivasan, Prakash and Alexander Yates. 2009. Quantifier scope disambiguation using extracted pragmatic knowledge: Preliminary results. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 1465–1474, Singapore.

Thater, Stefan. 2007. *Bridging the Gap Between Underspecification Formalisms: Minimal Recursion Semantics as Dominance Constraints*. Ph.D. thesis, Universität des Saarlandes.