Decoupling Scope Resolution from Semantic Composition

Iddo Lev
Computer Science Department, Stanford University
iddolev@stanford.edu

Abstract

A translation from Quasi-Logical Forms (QLFs) to Hole Semantics (HS) is presented. The translation makes explicit the dominance constraints that are implicit in the QLFs. QLFs can then be used in the semantic composition stage only, and HS in the scope resolution stage only. Thus, each representation is used to accomplish the stage it is most suited for while avoiding its cumbersomeness in the other stage.

1 Introduction

One approach in computational semantics for handling scope ambiguities is using Quasi-Logical Forms (QLFs) [HS87, Als92]. In these forms, quantifiers are represented using qterms that include the restrictor but not the body of the quantifier and that are left "in place" in the logical form. More recent techniques for dealing with scope ambiguities, such as Hole Semantics (HS) [Bos96], make use of dominance constraints between pieces of logical form.[1] It is easy to understand what formulas may be obtained from a representation in HS thanks to the explicit declarative specification of these dominance relations, whereas it is harder to understand what formulas are described by a QLF, because the dominance constraints are there only implicitly, as a side-effect of the way QLFs are resolved procedurally (see §2.3).[2]

This paper presents (in §3) a method for extracting the dominance constraints that are implicit in QLFs and expressing them explicitly using HS. This translation makes it easier to understand what formulas are described by a QLF. More importantly, because QLFs are simpler than HS in the semantic composition stage (see §4.1), the translation makes it possible to combine the two frameworks so that each can be used only for what it is best at, namely HS for scope resolution and QLFs for semantic composition. This contributes to an overall conceptually clearer system.

[1] Similar approaches are CLLS [EKN01] and MRS [CFPS03].

[2] Other approaches such as Categorial Grammars [Car98] and Glue Semantics [Dal99] obtain different scopings by different logical derivations. I do not address these here.

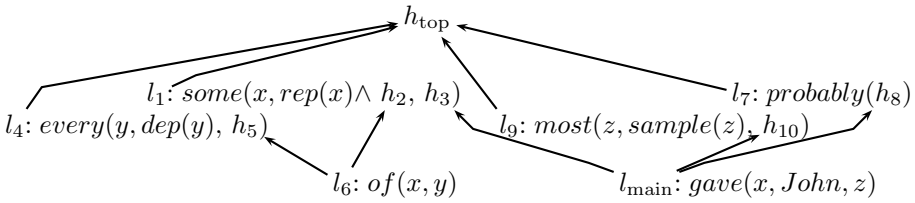

Figure 1: ULF for "Some representative of every department probably gave John most samples."

2 Explicit and Implicit Dominance Constraints

2.1 Hole Semantics

Hole Semantics (HS) [Bos96] specifies the dominance constraints between semantic pieces. Given a logical language L, it is extended to L^* by adding meta-variables called *holes* and *labels* which can serve as (sub-)formulas of L^*. Holes are marked by $h_1, h_2, ..$ and labels by $l_1, l_2, ...$ E.g., if L is first-order logic then L^* includes $\forall x.\ man(x) \rightarrow h_1$. An Unplugged Logical Form (ULF) is a tuple $\langle F, C \rangle$ consisting of a set F of labeled forms from L^* and a set C of dominance constraints between labels and holes.

Figure 1 is a graphical representation of the ULF for "Some representative of every department probably gave John most samples." An arrow from l to h represents the constraint $l \leq h$ meaning that l has to be a descendant of h in the final logical formula viewed as a tree. Thus $gave(x, John, z)$ must appear inside the scope of both quantifiers *some* and *most*, but these quantifiers' relative scoping is not determined (underspecified).

A legal solution ("plugging") to a ULF is a bijection from labels to holes which satisfies all the dominance constraints.[3] The formulas described by a ULF are those that can be obtained from a legal solution by actually plugging each labeled form into its corresponding hole. One legal plugging for the ULF above is $[h_{\text{top}} = l_9;\ h_{10} = l_7;\ h_8 = l_1;\ h_2 = l_4;\ h_5 = l_6;\ h_3 = l_{\text{main}}]$, producing the formula:

$$most(z, sample(z), probably(some(x, rep(x) \wedge every(y, dep(y), of(x, y)), gave(x, John, z)))).$$

[3]Formally, given a bijection P from labels to holes, let $G_P = \langle V, E \rangle$ be a graph, where the vertices V are all the holes and labels mentioned in the ULF, and E includes all edges (h, l) such that $P(l) = h$, and all edges (l, h) such that h is mentioned in the form labeled by l. Then P is legal iff G_P is a tree rooted at h_{top} such that for every dominance constraint $l \leq h$ in the ULF, h is an ancestor of l in G_P.

2.2 Quasi-Logical Forms

In Quasi-Logical Forms (QLFs) [Als92], quantifiers are represented using qterms that include the restrictor but not the body of the quantifier. Each qterm is placed in the QLF in a point that parallels its place in the syntactic analysis of the sentence. Thus, the QLF for the sentence above is:

(1) $probably(gave(qterm[some, x, rep(x) \wedge of(x, qterm[every, y, dep(y)])],$
$john,$
$qterm[most, z, sample(z)]))$

If $\gamma = qterm[Q, v, \psi]$ appears somewhere inside a QLF ϕ, then "pulling out" γ and "unstoring" it at the top of ϕ yields the result $quant(Q, v, \psi, \phi[v/\gamma])$.[4]

A formula is a valid solution for a QLF only if it can be obtained by a series of "pulling out" transformations that obey certain constraints, including: (a) A qterm γ_1 in the restrictor of a qterm γ_2 cannot be pulled out of γ_2 (γ_2 must be unstored first); and: (b) The top of a QLF ϕ can serve as the landing site for qterms inside ϕ only if ϕ is the entire sentence QLF, or if ϕ sits in an opaque argument position of its surrounding form [HS87]. The second constraint aims to deal with floating operators other than quantifiers, such as *probably* in (1), as well as other modals, negation, and opaque verbs like *think*. Whereas ULFs represent them in a similar way to quantifiers, QLFs treat them differently (they always stay in place rather than being pulled out, and they affect where qterms could be unstored).

2.3 Implicit Dominance Constraints in QLFs

The aim here is to discover, in explicit form, the dominance constraints that are implicit in a QLF. As a first step, a qterm is written as $qterm[Q, v, \psi, S]$, where the head noun ψ of the restrictor is separated from the set S of its modifiers. Figure 2 shows the (modified version of the) QLF from (1) as a tree, where directions are added to the lines to represent dominance constraints as follows: An arrow $a \rightarrow b$

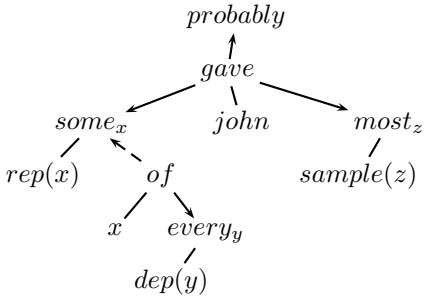

Figure 2: QLF as a tree

[4] A QLF is like a Keller store [Kel88] except that qterms in a QLF are written "in place" whereas in a Keller store, the place is filled with the quantified variable while a separate list holds the qterms. QLFs and Keller stores, in contrast to the original Cooper store [Coo83], retain the hierarchical information on quantifiers, i.e. which appears in the restrictor of which, and respect this information when unstoring quantifiers.

represents a constraint where a form a is dominated by the body of a floating formula b (i.e. the body of a quantifier, or of a floating operator like *probably*). A solid line directly connects a relation to an atomic argument, or a quantifier a to its restrictor's head b. A dashed arrow $a \dashrightarrow b$ represents a constraint where the main form a of an NP modifier is dominated by the restrictor of the NP's quantifier b.

Figure 2 shows one reason why dominance constraints in QLFs are implicit: the hierarchical relations in (1) are not the same as the dominance relations in the figure. Some of them have the same direction (e.g. *probably* over *gave* and *some* over of), but others have the reverse direction (e.g. *gave* is higher than *some* and of is higher than *every* in (1), but the directions of the dominance constraints in Figure 2 are reversed). In addition, a QLF is confusing regarding what dominance constraints it induces: although the qterm *every* is hierarchically a sub-formula of the qterm *some* in (1), there is no dominance constraint between *every* and *some* (but rather between of and *some*).

We can see that it is *conceptually* simpler to produce all the readings from a ULF compared to a QLF for several reasons. The dominance constraints are explicit in ULFs while they are implicit in QLFs (compare Figure 1 and Figure 2). The specification of a legal solution for a ULF is declarative and order-independent: The decisions that a ULF resolution algorithm needs to make are just which label should be plugged into which hole, and what matters is the final plugging rather than the order in which the various plugging decisions are made (indeed, different implementations exist, e.g. [BB99, ADK$^+$03]). In contrast, the specification in §2.2 is more complicated and is defined using transformational terms, so the decisions that a QLF resolution algorithm needs to make are where qterms should be unstored *and in what order* – these decisions are order-sensitive. Consequently, the published QLF resolution algorithms [HS87, Als92] are more complex to understand and follow compared to the ULF plugging algorithms (the output of the plugging algorithm is a plugging that can be understood independently of how the algorithm works, whereas there is no parallel thing for the QLF-disambiguation algorithms). Also, checking manually whether a given formula is described by a ULF is easier – only *local* tests are necessary (check whether each dominance constraint is satisfied in the formula), whereas checking the same for a QLF is more global and requires actually running the QLF disambiguation algorithm.

This explains the motivation for finding the dominance constraints of QLFs explicitly before enumerating their readings, and this is done in §3. The motivation for still using QLFs for semantic composition is given in §4.

3 Extracting Explicit Constraints from QLFs

3.1 Translation of QLFs to ULFs

How can explicit dominance constraints be extracted from QLFs? Consider the QLF in (1) from §2.2. We want to traverse it and accumulate dominance constraints and labeled forms to create an equivalent ULF. As shown in Figure 2 in §2.3, this can be achieved by *local* transformations (when visiting a sub-QLF, one needs only examine its direct children and a simple message passed to it from its parent). To avoid clutter, constraints and labeled forms will be accumulated in global variables $Constraints$ and $Lafos$. The function qlf-to-hs below returns on each non-atomic sub-QLF its main labeled form. After executing qlf-to-hs on the entire QLF of a sentence, the equivalent ULF for that sentence is $\langle Lafos, Constraints \rangle$.

function qlf-to-hs(ψ)
  if ψ is atomic **then return** ψ
  $l : \varphi \leftarrow$ qlf-to-hs-case(ψ)
  add $l \leq Hole$ to $Constraints$
  return $l : \varphi$

For now, $Hole = h_{\text{top}}$ (this global variable will be modified in §3.2 to deal with scope islands). The function qlf-to-hs-case branches on the structure of ψ. If ψ is a qterm, it should be converted to a quantifier form (quant). The quant's restrictor should include a new hole for each NP modifier, and the hole should dominate the modifier's main form (though not necessarily any quantifiers mentioned inside the modifier):

qlf-to-hs-case : $qterm[Q, v, \tau, \{\phi_1, \ldots, \phi_n\}]$
  let $l, h, h_1, \ldots, h_n$ be fresh label and holes
  $\varphi \leftarrow quant(Q,\ v,\ \tau \wedge h_1 \wedge \cdots \wedge h_n,\ h)$
  $l_i : \phi'_i \leftarrow$ qlf-to-hs(ϕ_i) for $1 \leq i \leq n$
  add $l : \varphi$ to $Lafos$, and add $\{l_i \leq h_i \mid 1 \leq i \leq n\}$ to $Constraints$
  return $l : \varphi$

$$qterm[Q, v, \tau, \{\phi_1, \ldots, \phi_n\}] \Rightarrow \begin{array}{c} l : quant(Q,\ v,\ \tau \wedge h_1 \wedge \cdots \wedge h_n,\ h) \\ \uparrow \qquad\qquad \uparrow \\ l_1 : \phi'_1 \quad \ldots \quad l_n : \phi'_n \end{array}$$

In the case of a (proposition-type) composite non-qterm form headed by an operator F, qlf-to-hs is recursively applied on each of the form's arguments. There are three types of arguments: (a) Atomic arguments should be left without change; (b) A qterm argument should be replaced by its quantified variable, and F should be dominated by the hole in the

new quant's body; (c) An opaque argument ([HS87]) is a composite non-qterm form (e.g. *gave*(..) is an opaque argument of *probably* in (1)), and it is replaced by a new hole which should dominate the argument's main part.

qlf-to-hs-case : $F(\phi_1, \ldots, \phi_n)$
  let l be a fresh label
  $l_i : \phi'_i \leftarrow$ qlf-to-hs(ϕ_i) for $1 \leq i \leq n$
  $args \leftarrow$ [as-arg$(\phi'_i) \mid 1 \leq i \leq n$] // sequence
  $C_1 \leftarrow \{l \leq h_i \mid \phi'_i = quant(\ldots, h_i)\}$
  $C_2 \leftarrow \{l_i \leq h_i \mid args[i] = h_i\}$
  add $l : F(args)$ to $Lafos$, and add $C_1 \cup C_2$ to $Constraints$
  return $l : F(args)$

$$\text{as-arg}(\phi) = \begin{cases} \phi & \phi \text{ is atomic} \\ v & \phi = quant(Q, v, \ldots) \\ h & \text{otherwise, where } h \text{ is a fresh hole} \end{cases}$$

Here is an example of the transformation, with c and d as atomic arguments.

$$F(qterm[Q_1, v_1, \tau_1, \{\phi_1\}], c, d, qterm[Q_2, v_2, \tau_2, \{\}]) \Rightarrow$$
$$l_1 : quant(Q_1, v_1, \tau_1 \wedge h_3, h_1) \qquad l_2 : quant(Q_2, v_2, \tau_2, h_2)$$
$$l_3 : \phi'_1 \qquad l : F(v_1, c, d, v_2)$$

An example of an opaque argument is the second argument of the predicate *want*. The QLF and the resulting ULF for "John wants to find a unicorn" are shown below. The plugging $[h_1 = l_3;\ h_2 = l_2]$ produces the *de dicto* reading, whereas the plugging $[h_2 = l_1;\ h_1 = l_2]$ produces the *de re* reading:

$$want(john, find(john, qterm[a, x, unicorn(x)])) \Rightarrow$$
$$l_1 : want(john, h_1) \qquad l_3 : quant(a, x, unicorn(x), h_2)$$
$$l_2 : find(john, x)$$

3.2 Scope Islands

Scope islands are areas in a sentence where nested quantifiers cannot outscope that area. For example, according to some views, (2.a) has two scopings, whereas the relative clause in (2.b) introduces a scope island (due to the use of "that"), so the quantifier *every* cannot escape it to outscope *some*.

(2) a. Some representative of every department arrived.
  b. Some person that represented every department arrived.

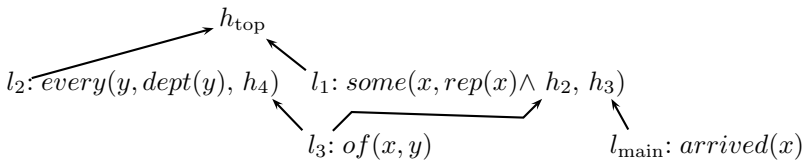

Figure 3: ULF for "Some representative of every department arrived."

Following [Als92, p. 161], we assume scope islands are marked in QLFs by SI(ψ).[5] A scope island can be enforced in the ULF by extending the function qlf-to-hs-case with a case for SI(ψ). The case introduces a new pair $l : h$ to $Lafos$, and then all the recursive calls starting with qlf-to-hs(ψ) should use that h rather than h_{top} as the hole dominating all the labels. To achieve this with minimal change to the code above, we do not pass h as an explicit parameter to the functions, but instead use a Lisp-like style of dynamic binding of global variables: If X is a global variable, then the command "**let** X **be** a **in** *expression*" temporarily binds X to a, evaluates *expression*, and then restores X to the value it had before it was bound to a. We bind the global variable *Hole* (used above in qlf-to-hs) as follows:

qlf-to-hs-case : SI(ψ)
  let l, h be a fresh label and hole
  add $l : h$ to $Lafos$
  let *Hole* **be** h **in** $l' : \psi' \leftarrow$ qlf-to-hs(ψ)
  return $l : h$

To illustrate how this works, consider the ULF in Figure 3. This ULF can be resolved in two ways, one where *some* outscopes *every* (by plugging l_2 into h_2), and the other where *every* outscopes *some* (by plugging l_1 into h_4). This ULF is obtained as usual from the QLF of (2.a), namely: $arrived(qterm[some, x, rep(x), \{of(x, qterm[every, y, dept(y), \{\}])\}])$. However, (2.b) will get a QLF with SI: $arrived(qterm[some, x, person(x),$ $\{\mathbf{SI}(represented(x, qterm[every, y, dept(y), \{\}]))\}])$. When qlf-to-hs-case is called on the SI(...) form, *Hole* will be bound to a new h in the recursive call qlf-to-hs($qterm[every, \ldots]$). Therefore, the constraint $l_2 \leq h$ rather than $l_2 \leq h_{\text{top}}$ is used in the resulting ULF, as shown in Figure 4. This ULF has only one legal solution where *some* outscopes *every*. The same idea would work for scope islands introduced by certain argument positions of some predicates (e.g. sentential complements), and by conditionals.[6]

[5] SI(...) is just used to mark a scope island and does not affect the semantics.

[6] The code here sometimes generates the same solution twice because a pair $l : h$ may serve as a redundant landing site for embedded floating forms. This can be fixed if such

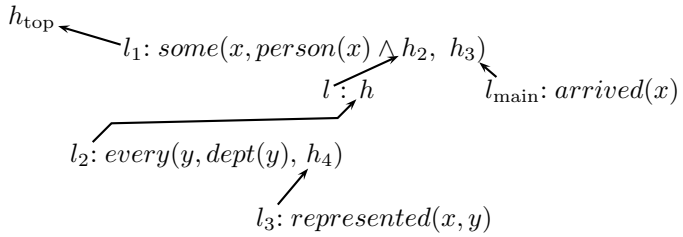

Figure 4: ULF: "Some person that represented every department arrived."

Indefinites (and other operators) may sometime escape scope islands, or obey only stricter scope islands (e.g. they may escape relative clauses but not the antecedents of conditionals). The translation presented here could be revised to take such distinctions into account by using more than one kind of global *Hole* and adding different constraints for different kinds of operators in qlf-to-hs (thus generalizing the treatment of indefinites in [Cha02]).

3.3 Significance of the Translation

The translation of a QLF produces an equivalent ULF that describes the same formulas. Moreover, it is the same ULF that is given in the HS literature for the sentence.[7] The translation exposes in an explicit form the dominance constraints that are implicit in QLFs, thereby making it easier to understand what formulas are described by a QLF.[8]

Moreover, the simpler ULF resolution algorithms can now also be used to resolve QLFs after they are translated to ULFs. One might argue that this is unnecessary because the existing resolution algorithms for QLFs can be used as-is. But note that one may want to modify the resolution process to novel or special needs, such as filtering logically equivalent scopings. It is easier to modify the more transparent ULF plugging algorithm of [BB99] to achieve such filtering (see [Cha03]) than it is to modify the QLF algorithm (such modifications in [Als92] make the basic algorithm even less transparent).

Now we face the question: if ULFs are conceptually simpler than QLFs, should we use only ULFs? The next section addresses this question.

---

a pair is always plugged directly into the hole that immediately dominates l.

[7]These claims were verified by implementing and testing the algorithm on standard cases from the literature, roughly those shown above, and a formal proof is in the works. It can also be easily verified that the running time of the translation algorithm and the size of the resulting ULF are linear in the size of the QLF.

[8]It might turn out that HS is more expressive than QLFs, so a translation between the two might be possible only in one direction. However, whether the formalisms differ on linguistically-relevant cases remains to be investigated.

4 Taking Semantic Composition Into Account

4.1 Constructing QLFs is Easier than ULFs

To answer the last question, one also needs to consider how QLFs and ULFs are constructed from the syntactic analysis – let us call this stage "semantic composition". Constructing QLFs is simpler than ULFs, as shown by considering what the semantic pieces of syntactic sub-trees look like.

First consider the QLF version of noun phrases. Proper names can be represented by a simple constant:

John $\Rightarrow$ $john$

This combines simply with "sleeps" i.e. $\lambda y.sleep(y)$ to form $sleep(john)$. The nice thing about QLFs is that this can also be done for quantifiers because they are treated during the composition stage as qterms. Thus, the semantic entry for "every" is something like:

every $\Rightarrow$ $\lambda P.[every, x, P(x)]$

Combined with the semantic entry for "man" $\lambda y.man(y)$ this gives:

every man $\Rightarrow$ $[every, x, man(x)]$

and this form combines with "sleeps" in the same way as a proper name:

every man sleeps $\Rightarrow$ $sleep([every, x, man(x)])$.

In contrast, approaches based on Montague's grammar, such as HS [BB99], require a higher type for quantified NPs, as well as force the meaning of proper names to be raised to this higher type to allow uniform treatment of NPs in the semantic rules. Thus, the semantic entries become:

John $\Rightarrow$ $\lambda P.P(john)$

every $\Rightarrow$ $\lambda P \lambda Q.every(x, P(x), Q(x))$

Of course, with two or more quantifiers, such entries will produce only one of their scopings if only naive lambda-application is used. This is the very reason for using a scope ambiguity framework like HS. And in HS, the situation is made even more complicated by requiring the semantics of every sub-phrase to be composed of a partial UR which is preceded by two additional variables – the local hole and label – and sometimes followed by merge operations with other partial URs. Following [BB99], the entries become:

John $\quad \lambda P \lambda h \lambda l.\ \ P(john, h, l)$

sleeps $\quad \lambda y \lambda h \lambda l.\ \langle \{l : sleep(y)\}, \{l \leq h\} \rangle$

man $\quad \lambda x \lambda h \lambda l.\ \langle \{l : man(x)\}, \{l \leq h\} \rangle$

every $\quad \lambda P \lambda Q \lambda h \lambda l.\ \langle \{l_1 : every(x, l_2, h_1)\}, \{l_1 \leq h,\ l \leq h_1\} \rangle \oplus P(x, h, l_2) \oplus Q(x, h, l)$

every man $\quad \lambda Q \lambda h \lambda l.\ \langle \{l_1 : every(x, l_2, h_1),\ l_2 : man(x)\}, \{l_1 \leq h,\ l \leq h_1,\ l_2 \leq h\} \rangle \oplus Q(x, h, l)$

every man sleeps $\quad \lambda h \lambda l.\ \langle \{l_1 : every(x, l_2, h_1),\ l_2 : man(x),\ l_3 : sleep(x)\}, \{l_1 \leq h,\ l_3 \leq h_1,\ l_2 \leq h,\ l_3 \leq h\} \rangle$

It is clear that QLFs are less complex and are simpler to follow during composition compared to ULFs. This is because the structure of QLFs mimics closely the syntactic analysis, as they leave material more-or-less "in place", and they do not mention anything about scoping. Partial QLFs are just sub-expressions of full QLFs, with only few preceding lambda variables, as shown above. In contrast, the meta-level ULFs are further removed from the surface forms and it is more complex to design a composition stage that computes them directly from the parse trees. In particular, each semantic piece needs to specify what dominance constraints it has internally as well as w.r.t. other pieces, so one needs to resort to using a local hole and label and merge operations. Moreover, it is hard for a human to comprehend the big and complex partial ULFs and to visualize in one's mind graphically what an expression like $\langle\{l_1 : every(x, l_2, h_2), ...\}, \{l \leq h_2, ...\}\rangle$ means, and so grammar writers who want to work with partial ULFs and debug them often need to draw their constraint graphs on paper.[9]

4.2 Combining the Advantages of the Frameworks

We have seen that for semantic composition, it is easier to understand and work with QLFs rather than ULFs, whereas the reverse holds for scope resolution. Thanks to the translation, one can enjoy the advantages of the representations without having to use either for what it is more cumbersome at and less revealing about: QLFs can be used during composition, they can then be translated to ULFs, and these can be subsequently resolved by the plugging algorithm. Again one might argue that this is unnecessary because the existing ULF-construction methods can be used. But one obviously needs to extend the scope of the existing HS grammar and debug it by examining semantic results for syntactic sub-trees, and this is easier to do with QLFs as was shown above. Drawing graphical ULFs on paper is now required only for the entire sentence rather than for each sub-tree.[10]

QLFs are still relevant today and are preferred to ULFs for another reason: thanks to their simplicity, they are easier to use in combination with robust statistical parsing. For example, in [CMvGW03], the parse trees

[9]The same point holds for feature-structure representations of partial ULFs as well, like [Cha02, RS03], which can be viewed simply as notational variants of partial ULFs.

[10]Resolving ULFs might be *computationally* more complex than resolving QLFs (while there are studies on the complexity of ULF resolution (e.g. [ADK+03]), I am not aware of such studies for QLFs). But asymptotic complexity results might be quite different from actual run-time results on real sentences. In any case, I agree with the view that it is better to have a conceptually clear algorithm first and worry about efficiency issues later.

obtained from the probabilistic parser are automatically converted to LFG f-structures and then to QLFs. So another benefit of our translation is that it can add the further step of converting these QLFs to ULFs, and then resolving the latter using the simpler ULF resolution algorithms. This contributes to bridging the gap between robust processing and precise semantics.

5 Related Work

[vGC96] explores translations between QLFs, UDRSs [Rey93] (the precursor to Hole Semantics), and LFG f-structures. Although the paper provides a translation from QLFs to f-structures to UDRSs, the analysis is limited. In particular, it does not cover cases of quantifiers in the restrictor of other quantifiers, floating forms with opaque arguments, and scope islands.

Other translational work between underspecified representations worth mentioning is the translation from HS and MRS to CLLS [KNT03, FKNT04]. However, semantic composition is complex in these frameworks for the same reasons as explained in §4.1.

6 Conclusion and Future Work

A translation of QLFs to Hole Semantics ULFs was presented. The translation exposes in explicit form the dominance constraints that are implicit in the QLFs, thus revealing more clearly what formulas are described by them. The gain is that we can combine the relative advantages of the two formalisms – QLFs for semantic composition and ULFs for scope resolution.

It might be interesting to investigate whether our translation can be modified to produce LFG f-structures (and thereby representations in Glue Semantics [Dal99]) rather than ULFs. Moreover, the Hole Semantics framework [Bos96, BB99] actually computes ULFs of Discourse Representation Structures (DRSs) rather than first-order logic formulas. DRSs can then support anaphora resolution after scope resolution is completed. QLFs also represent anaphoric expressions and have a separate stage for resolving them. Future work may investigate how to extend the idea of this paper to connect the two formalisms also for anaphora resolution.

Acknowledgments

Thanks to Stanley Peters, Johan Bos, and Rui Chaves for reading a draft of this paper and for useful comments and discussions. All errors are mine.

References

[ADK+03] Ernst Althaus, Denys Duchier, Alexander Koller, Kurt Mehlhorn, Joachim Niehren, and Sven Thiel, "An efficient graph algorithm for dominance constraints." Journal of Algorithms. To appear, 2003.

[Als92] Hiyan Alshawi, ed., *The Core Language Engine*. MIT Press, 1992.

[BB99] Patrick Blackburn and Johan Bos, *Representation and Inference for Natural Language*. Stanford: CSLI Press, (1999). Forthcoming.

[Bos96] Johan Bos, "Predicate logic unplugged," in *Proc. of the 10th Amsterdam Colloquium*, pp. 133–142, 1996.

[Car98] Bob Carpenter, *Type-Logical Semantics*. MIT Press, 1998.

[CFPS03] A. Copestake, D. Flickinger, C. Pollard, and I. Sag, "Minimal recursion semantics: an introduction," 2003. To appear.

[Cha02] Rui P. Chaves, "Principle-based DRTU for HPSG: a case study," in *Proc. of SCANALU'02*, 2002.

[Cha03] Rui P. Chaves, "Non-redundant scope disambiguation in underspecified semantics," in *Proc. of the 8th ESSLLI Student Session* (Balder ten Cate, ed.), pp. 47–58, 2003.

[CMvGW03] Aoife Cahill, Mairead McCarthy, Josef van Genabith, and Andy Way, "Quasi-logical forms for the penn treebank," in *Proc. of IWCS-05* (Harry Bunt, Ielka van der Sluis, and Roser Morante, eds.), pp. 55–71, 2003.

[Coo83] R. Cooper, *Quantification and Syntactic Theory*. Dordrecht: Reidel, 1983.

[Dal99] Mary Dalrymple, *Semantics and Syntax in Lexical Functional Grammar: The Resource Logic Approach*. MIT Press, 1999.

[EKN01] M. Egg, A. Koller, and J. Niehren, "The constraint language for lambda structures," *Journal of Logic, Language, and Information*, 2001. To appear.

[FKNT04] Ruth Fuchss, Alexander Koller, Joachim Niehren, and Stefan Thater, "Minimal Recursion Semantics as dominance constraints: Translation, evaluation, and analysis," in *Proc. of the 42nd Meeting of the ACL*, 2004.

[HS87] Jerry R. Hobbs and Stuart M. Shieber, "An algorithm for generating quantifier scopings," *Computational Linguistics*, vol. 13, no. 1-2, pp. 47–63, 1987.

[Kel88] W. R. Keller, "Nested cooper storage: The proper treatment of quantification in ordinary noun phrases," in *Natural Language Parsing and Linguistic Theories* (U. Reyle and C. Rohrer, eds.), Reidel, Dordrecht, 1988.

[KNT03] Alexander Koller, Joachim Niehren, and Stefan Thater, "Bridging the gap between underspecification formalisms: Hole semantics as dominance constraints," in *Proc. of the 11th EACL*, 2003.

[Rey93] U. Reyle, "Dealing with ambiguities by underspecification: Construction, representation and deduction," *J. of Semantics*, vol. 10, pp. 123–179, 1993.

[RS03] Frank Richter and Manfred Sailer, "Basic concepts of lexical resource semantics," in *Series of the Kurt Gödel Society, Vienna*, 2003. Also appeared as ESSLLI'03 course notes.

[vGC96] Josef van Genabith and Richard Crouch, "F-Structures, QLFs and UDRSs," in *Proc. of the 1st Int'l Conference on LFG* (Miriam Butt and Tracy Holloway King, eds.), pp. 190–205, CSLI Publications, 1996.