Bidirectional Contextual Resolution

Stephen G. Pulman*
University of Cambridge,
& SRI International Cambridge Computer Science Research Centre

This paper[1] *describes a formalism and implementation for the interpretation and generation of sentences containing context dependent constructs like determiners, pronouns, focus, and ellipsis. A variant of 'quasi-logical form' is used as an underspecified meaning representation, related to 'resolved logical forms' via 'conditional equivalences'. These equivalences define the interpretation of contextually dependent constructs with respect to a given context. Higher order unification and abduction are used in relating expressions to contexts. The conditional equivalences can be used unchanged in both the interpretation and the generation direction.*

1 Introduction

This paper has several aims. Firstly, it outlines a formalism within which quasi-logical form based theories of contextual interpretation of sentences can be stated in a way which is completely reversible; that is to say, theories expressed within the formalism can be used to provide interpretations for (utterances of) sentences containing contextually dependent constructs, given a context; and, given an interpretation and a context,to generate a sentence which has that interpretation in that context. Processing in both directions is done using exactly the same grammar, and the same set of contextual interpretation rules.

To give an extremely simplified example, the aim is to have a way of interpreting the sentences in the left hand column below as expressing the logical forms in the middle column, given that they are encountered in that order in an otherwise neutral context: and the reverse - given a sequence of logical forms as in the middle column, to be able to generate (among others) the sequence of sentences in the left hand column, rather than the unnatural although literally correct version given in the right hand column.

Joe sneezed.	*sneeze(joe)*	Joe sneezed
He laughed.	*laugh(joe)*	Joe laughed
Bill laughed too.	*laugh(bill)*	Bill laughed
...	...	...

We will assume that if the context is loosely specified enough to permit alternative realisations of the same content, then different versions of the same text could be generated or analysed:

* University of Cambridge Computer Laboratory, New Museums Site, Cambridge, CB2 3QG. sgp@cl.cam.ac.uk

1 Published as: S. G. Pulman, 2000, Bidirectional Contextual Resolution, in Computational Linguistics, Vol 26/4, 497-538.

1 Joe sneezed and laughed. Bill laughed too
Joe sneezed. Joe and Bill laughed.
Joe sneezed. He laughed. So did Bill.

Secondly, we illustrate the formalism and the general approach by taking a specific approach to contextual interpretation based on a kind of 'quasi-logical form' and giving an account of a fragment (in the sense of Montague (Montague, 1974a)) which treats several core phenomena of English contextual dependence.

We also have some theoretical objectives: the particular approach illustrated here, like most computational approaches to contextual interpretation, uses an intermediate 'quasi-logical form' representation level. Using such a level of representation incurs an obligation to say what it means ('no notation without denotation'). We try to show how the theory presented here leads to a natural semantics for these quasi-logical forms, and indeed leads to a truth theory for contextually dependent interpretation which supports a natural consequence relation, and one appropriate for cases where interpretations are not fully specified. We relate this approach both to the classical tradition of formal linguistic semantics exemplified by Davidson (Davidson, 1972) and Montague (Montague, 1974b) and more recent literature on the use of underspecification in semantics.

The structure of the paper is as follows. In the next section we give an outline of the formalism and illustrate with the small fragment of English that has been implemented within this framework. We present analyses of the contextual interpretation of pronouns, definites, ellipsis, focus, and quantifier scope. There is far more to say about each of these phenomena, of course, and the analyses here are by no means claimed to be definitive. The aim is merely to show that we can, to a first approximation, provide a reasonably fully worked out description of these phenomena in a truly bidirectional way.

We then go on to compare the current approach with that of some other theories with similar aims: the 'standard' version of quasi-logical form implemented in the Core Language Engine, as rationally reconstructed by Alshawi and Crouch (Alshawi and Crouch, 1992) and Crouch and Pulman (Crouch and Pulman, 1994); underspecified Discourse Representation Theory (Reyle, 1993); and the 'glue language' approach of Dalrymple et al. (Dalrymple et al., 1996).

Finally, we discuss some of the semantic and logical issues raised by the approach described here, in particular the extent to which the theory meets the desiderata for accounts of underspecification outlined by van Eijck and Jaspars (van Eijck and Jaspars, 1996), and the extent to which the theory supplies a methodologically satisfactory account of truth and interpretation for sentences involving contextually dependent constructs.

2 Contextual Interpretation

The major components and assumptions of the approach to contextual interpretation here are as follows:

1. We assume that the output of grammatical processing of a sentence is a quasi-logical form, henceforth QLF. Of course, for anything other than a trivial grammar, a given sentence will typically yield many QLFs. We will assume that syntactic and lexical disambiguation have taken place and that the only things still needed for a complete interpretation are the resolution of constructs like pronouns, definites, ellipsis, and so on. We return later to issues concerning robustness of linguistic coverage and to the interleaving of contextual disambiguation with syntactic and semantic processing.

For concreteness, we are are assuming here that QLFs are built using a simple unification grammar formalism of the type described in (Pulman, 1996), and that a chart parser and semantic head driven generator are used for the analysis of sentences to QLFs and vice-versa. But little of this detail is essential to our main aims: a wide range of grammatical formalisms and interpreters would be compatible with the basic assumptions of the contextual interpretation mechanism, assuming only that the same grammatical description is used in both the analysis and generation direction.

What *is* required is that QLFs are, as here, expressed in a typed higher order logic, augmented with constructs representing the interpretation of context-dependent elements (pronouns, ellipsis, focus etc.). These constructs correspond as directly as possible to properties of the linguistic structure that express them and are to as small an extent as possible dependent on the requirements of contextual resolution (unlike, say, the metavariables of standard QLFs (Alshawi and Crouch, 1992), or the labels of UDRS (Reyle, 1996), which are motivated entirely by the mechanisms that operate on them after grammatical processing). Syntactic properties relevant for binding constraints, parallelism, scope constraints, and so on, are not directly represented at QLF (again unlike standard QLFs) but are assumed to be available as components of the linguistic context.[2]

2. The context independent meanings of sentences, which we refer to as resolved logical forms (RLFs), are expressed in the 'ordinary' subset of the QLF language. A fully resolved RLF can be directly evaluated for truth: it contains no QLF constructs. Since it is just an expression of 'ordinary' logic it could serve as a knowledge representation and reasoning language, and thus the output of some information system producing such representations could in principle feed directly into generation (modulo well known 'equivalence of logical form' problems).

3. 'Contexts' are here modelled by sets of sentences in the RLF subset of this language, with some kind of salience ordering on them (recency, in the implementation), about which we say nothing more. These sentences may, but need not, arise from prior linguistic processing. Contexts contain information about the form as well as the content of previous utterances, as mentioned earlier. Context sentences may also reflect features of the non-linguistic context gained by direct observation or inference.

This is a very minimal theory of context. We need to be able to reason about

2 This is probably too strong a position to take. There are good arguments for allowing some syntactic distinctions to be represented more directly.

context, hence we need it represented in a logic. We need to be able to refer to properties of the form of linguistic utterances as well as their content, hence context must contain this information too. We obviously need some non-linguistic information. We also need some structure to reflect the fact that not all components of the context are relevant to everything, hence salience. This is all we need for the time being, although there is clearly much more to be said.

4. QLFs are interpreted by 'conditional equivalences' (Rayner and Alshawi, 1992; Rayner, 1993) of the form:

```
QLF ⇔ RLF
    if
     Condition_1,
     ...,
     Condition_n.
```

These state a contextual equivalence between an expression containing one or more QLF constructs (the left hand side) and an expression containing at least one fewer QLF constructs (the right hand side). 'QLF' and 'RLF' are therefore sometimes used to signify partially as well as fully (un)resolved LFs. An equivalence can be paraphrased as: 'In a context where these conditions hold, this QLF can be interpreted as this RLF', or 'In a context where these conditions hold, this RLF can be expressed as this QLF'. Conditional equivalences, if ⇔ is interpreted as material equivalence, can be unpacked to a conjunction of implications:

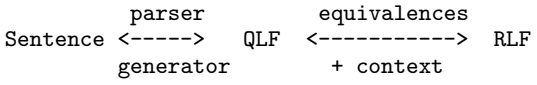
```
(Conditions & QLF → RLF)
&
(Conditions & RLF → QLF)
```

5. Conditions are treated as goals to be satisfied with respect to the current context, in a way familiar from the theorem proving and logic programming tradition. Variables in goals may or may not be instantiated and satisfying a goal can instantiate variables non-deterministically. The scope of a variable is within the whole equivalence. The interpretation of variables is as for Prolog.

Later, we extend the notion of inference involved in checking conditions beyond that provided by Prolog and the like to allow conditions to be 'abduced' and added to the context if they cannot be proved directly, always provided that adding them to the context does not cause a contradiction. We assume some 'cost' mechanism constrains this process.

6. Equivalences describe QLF or RLF patterns, typically containing variables. Determining whether an equivalence applies to a QLF or RLF is done by higher order unification (henceforth HOU) (Huet, 1975; Miller and Nadathur, 1986; Pulman, 1991; Dalrymple, Shieber, and Pereira, 1991; Gawron, 1992) of the logical form with the relevant pattern. Many of the contextual conditions require a higher-order equation to be solved.

7. The interpretation of a QLF is given via the RLFs it can be equivalent to with respect to given contexts. Given a fixed, fully-specified, context, a QLF will generally be equivalent to a single RLF (unless the equivalences allow for several synonymous interpretations). In cases where the context does not resolve an ambiguity the QLF will correspond to different RLFs depending on which



assumptions are added to the context. Likewise, given a partially specified context and an RLF there may be several QLFs that can express the content of the RLF. Notice that the equivalence holds *if* the conditions hold, not *iff*.

The overall architecture of the system can be pictured very simply:

3 An illustrative fragment

3.1 Pronouns

It is easiest to see how all this is supposed to work out by giving some examples. Consider the simple discourse:

2 Smith owned NLPCom. He disappeared.

The QLF we will assign to the first sentence will be:

$\texttt{exists1}(\lambda\texttt{e.pos(past(own(e,smith,nlpcom))))}$

We actually interpret sentences as predicates on eventualities (type 'ev'), and interpret tense and aspect markers as QLF operators, subject to contextual interpretation of a complex kind (see (Pulman, 1997a; Thomas and Pulman, 1999), for an account of reversible tense and aspect interpretation within this framework). But for the purposes of this paper we will simply assume that tense and aspect processing consists of quantifying over the event variable, and further simplify by assuming that this happens in the grammar rather than in resolution. In the logical form, 'pos' is the opposite of 'neg' and and is motivated as an explicit element of QLF by the fact that the positive polarity of a sentence can be focussed, as in 'Smith DID disappear'.

The quantifier $\texttt{exists1}_{(\alpha\texttt{>t)>t}}$ is so-called to distinguish it from the generalised quantifier $\texttt{exists}_{\texttt{(e>t)>(e>t)>t}}$ used later[3].

This QLF/RLF, given our simplifying assumptions, needs no resolution, and will form the context for the interpretation of the QLF for the subsequent sentence:

$\texttt{exists1}(\lambda\texttt{f.pos}_{\texttt{t>t}}(\texttt{past}_{\texttt{t>t}}(\texttt{disappear}_{\texttt{ev>e>t}}\texttt{f}_{\texttt{ev}},\texttt{he}_{\texttt{e}}))$

Note that the interpretation of the pronoun is represented by the QLF construct $\texttt{he}_{\texttt{e}}$ which adequately summarises the properties of singularity and masculinity required of an antecedent. (For computational economy, we might want to generalise this representation in an implementation to something like `pron(X)` where X is 'he', 'she' etc. to enable a single equivalence to cover all the cases, but there is no linguistic motivation for including any more information than we have.)

3 Type subscripts will be omitted where they are easy to infer.

Next we try to resolve the QLF construct 'he'. We have, we will assume, an equivalence of the form:

Pron-he:
```
Pred(he) ⇔ Pred(Ref)
     if
      salientContext(pronoun,Context),
      possibleAntecedent(Context,he,Ref),
      binding_conditions_hold...
```

which is one of several that might be applicable. (We follow Prolog-like notational conventions: query variables begin with upper case; variables beginning with underscore '_áre those whose instantiation we are not interested in; constants begin with lower case, and ',' between expressions is interpreted as conjunction. Lambda-bound variables of type 'ev' (eventuality) are `e,f,g,...` and those of type 'e' (individual) are `x,y,z...`). Applicability is determined in two steps: first, equivalences are indexed by any QLF constructs that they involve (like 'he'), and secondly, higher order unification is tried between the QLF and the left hand side of an equivalence retrieved by the indexing. The indexing step is necessary both for completeness and for efficiency: if we just used HOU then, since it is not decidable, equivalences which did not match the QLF could lead to non-termination, or at best to spurious matches which would be filtered out expensively when checking the conditions. (To see how this could be so, consider that trying to HOU `Pred(he)` with any formula F of the appropriate type will succeed with `Pred = λx.F`, where `x` does not occur in F.) Other than this, application of equivalences is entirely free and non-deterministic. Of course, if in the analysis direction we require full resolution, then we will have to continue until all QLF-constructs have been resolved. But in the generation direction application of some constructs will be optional (such as this pronoun one),and some will be in effect obligatory (like the quantifier scoping eequivalences described later) because failure to apply will not result in a QLF that the grammar can generate from. Equivalences currently apply to an entire QLF, although in reality this is an oversimplification and some more dynamic and incremental control regime should be used. We return to this issue later.

Note also that here and throughout the paper, there may be alternative solutions to equations in equivalences, some corresponding to alternative interpretations, and some that will hopefully be filtered out by the relevant conditions. We assume that it is possible to represent structural constraints like binding and scoping principles as conditions in an equivalence. To keep the presentation manageable, we abstract away from these issues, and also avoid questions of how the correct interpretation is actually chosen, where there is a choice.

The conditions in this pronoun equivalence are stated in terms of several predicates that recur in the treatment of different phenomena. The predicate `salientContext(Construct,Context)` finds a logical form that is a salient one for the current construct in the context. Having the construct as a parameter enables search to be reduced: pronouns and ellipsis typically find their antecedents in either an earlier portion of the current sentence, or the preceding sentence (Hobbs, 1979), whereas definites frequently refer back over several previous sentences. We

parameterise this predicate so that these preferences can be respected.

The predicate `possibleAntecedent(Context,Proform,Candidate)` does most of the work. The simplest clause in its definition is:

```
possibleAntecedent(Context_t,he,Ref_e)
  if
    Context = _OtherPred(Ref),
    isOfType(he,Ref).
```

where = means that the equation is solved by HOU, and the predicate `isOfType` carries out the obvious number and gender checks. More complete definitions of `possibleAntecedent` would include checks for the type of restriction often expressed as binding constraints, and for the type of preferences obtained by centering theory. We ignore these details here since they are not our main focus.

The sequence of unifications now is that the QLF

```
exists1(λf.pos(past(disappear(f,he))))
```

will HOU with the expression `Pred(he)` to give

```
Pred=λx.exists1(λf.pos(past(disappear(f,x))))
```

`SalientContext` will return `exists1(λe.pos(past(own(e,smith,nlpcom))))` as the value of `Context`, and when we attempt to solve the goal `possibleAntecedent` we will have two non-vacuous solutions for the equation in its definition:

```
Context = _OtherPred(Ref)
```

with `Ref=smith`, or `nlpcom`. Of these, only the first will pass the `isOfType` test, and so the RLF side of the equivalence will be instantiated to:

```
[λx.exists1(λf.pos(past(disappear(f,x))))](smith)
```

which after beta-reduction will be the intended interpretation.

Consider now what would happen if we were operating in the other direction; that is to say, we have the same sequence of resolved logical forms and we wish to generate sentences expressing them. The first logical form:

```
exists1(λe.pos(past(own(e,smith,nlpcom))))
```

has no relevant context (for our purposes) and thus leads directly to the sentence 'Smith owned NLPCom'. For the second logical form one outcome is that it is also treated in a context independent way and the sentence 'Smith disappeared' is generated. (We should have some way of ranking this as dispreferred, but we will postpone that issue for now.) The other possible outcome is that we apply the pronoun equivalence in the generation direction. Conditions for applicability are a little more difficult here, because we often have no QLF constructs to index equivalences from. Instead we currently have to rely on coarser indexing heuristics. Assuming that, we then use HOU to check the conditions for applicability: we will unify `Pred(Ref)` with `exists(λf.pos(past(disappear(f,smith))))` and the conditions will locate the prior reference to 'Smith' as constituting a sufficient condition for realising this occurrence of 'Smith' by the pronoun 'he'.

In fact, as the equivalence is stated, we will be able to also generate the sentence 'he disappeared' if the current RLF was `exists1(λf.pos(past(disappear(f,jones)))`, which is clearly incorrect. The reason for this is that the HOU in the `possibleAntecedent` condition might also succeed with a vacuous solution, namely:

`_OtherPred = λx.exists1(λf.pos(past(disappear(f,smith)))`

and the remainder of the conditions will also succeed. We must therefore restrict solutions to this equation to non-vacuous ones: in fact, we will not lose anything by making this a general restriction on admissible solutions[4], as we have already been doing implicitly.

Of course this analysis of pronoun reference will cover only the simplest possible cases of inter-sentential anaphora. Before going on to more complex cases, we will also show how to deal with intra-sentential anaphora, including reflexives, and binding of a pronoun by a quantifier. The relevant equivalence is:

Pron-he-intra
$\texttt{Rest}_{\texttt{(e=>t)=>t}}(\lambda\texttt{y.Pred(y,he)}) \Leftrightarrow \texttt{Rest}_{\texttt{(e=>t)=>t}}(\lambda\texttt{y.Pred(y,y)})$
if
binding_conditions_hold....

This equivalence is doing essentially the same job as Pereira's 'pronoun abstraction' schema in (Pereira, 1990). It will identify a pronoun with any term of type 'e' elsewhere in the QLF, relying on the binding conditions to prevent impossible associations.

We illustrate this equivalence with the relevant instantiations for the following cases (in fact the reflexive case is done with a separate equivalence differing only in that it mentions 'he-self' instead of 'he', with associated differences in binding conditions):

3 Smith admires himself

```
    QLF=exists1(λe.pos(pres(like(e,smith,he-self))))
Rest= λQ.Q(smith)
Pred=λx.λy.exists1(λe.pos(pres(like,e,x,y)))
RLF=exists1(λe.pos(pres(like(e,smith,smith))))
```

4 Smith likes his computer

```
    QLF=exists1(λe.pos(pres(like(e,smith,of(he,computer)))))
Rest= λQ.Q(smith)
Pred=λx.λy.exists1(λe.pos(pres(like,e,x,of(y,computer))))
RLF=exists1(λe.pos(pres(like(e,smith,of(smith,computer)))))
```

4 For this case, and for many other types of restriction currently handled by conditions, more elegant solutions are available using the 'sorted' and 'coloured' versions of higher order unification developed by Michael Kohlhase (Gardent and Kohlhase, 1996b; Gardent, Kohlhase, and van Leusen, 1996; Gardent and Kohlhase, 1997; Gardent, Kohlhase, and Konrad, 1999)

5 Every manager likes his computer

```
   QLF (partially resolved) = forall(manager,λx.exists1(λe.pos(pres(like,e,x,of(he,computer)))))
Rest=λQ.forall(manager,Q)
Pred=λa.λb.exists1(λe.pos(pres(like,e,a,of(b,computer))))
RLF = forall(manager,λx.exists1(λe.pos(pres(like,e,x,of(x,computer)))))
```

Note that here we have assumed that the quantifier has already been scoped. We return later to issues of the interaction of scoping with ellipsis and anaphora. In the meantime we simply point out that bound variable uses of pronouns need no extra mechanisms than those required for simple intra-sentential pronouns.[5]

It is easy to extend to far more complex cases of intersentential anaphora by extending the definition of `possibleAntecedent` to allow for reference to different types of antecedent. For example, if we adopt a theory like Webber's (Webber, 1983), we can construct discourse referents from the representation of quantified NP meanings. Recall that in Webber's approach, a logical form representing the meaning of a sentence processed in a discourse will trigger the application of rewrite rules which will add new entities to the context. For example, given a logical form which in our notation would be:

```
exists(cat,λX.saw(I,X)))
```

a discourse entity like:

```
iota(λX.cat(X) & saw(I,X) & evoke(s1,X))
```

will be produced, where 'iota' is a term forming operator roughly interpreted like a definite description. (The 'evoke' predicate serves as a unique identifier for the referent, tagging if with a label for its source sentence.)

Webber's rules lend themselves very naturally to a higher order formulation, although when systems based on her theory have been implemented on a realistic scale they have been implemented either as code or as Lisp pattern-matching rules (Ayuso, 1989). Using HOU we can formulate an axiom to infer the existence of an entity of the appropriate type:

```
Pred_{t>t}(exists_{(e>t)>(e>t)>t}(Restriction,Body))
  →
  exists1_{(e>t)>t}(λx.x=iota(λy.Restriction(y) & Body(y)))
```

The operator $\texttt{iota}_{(e>t)>e}$ means here 'the (unique) thing satisfying (the intersection of) restriction and body'. An additional clause in the definition of `possibleAntecedent` essentially encodes this inference:

```
possibleAntecedent(Context,he,DE)
  if
    Context = Pred_{t>t}(exists(Restriction,Body)),
```

5 A referee queries whether generating back out from the resolved form of 5 might not leave a 'dangling variable' when the quantifier scoping is undone. This is not so, for the simple reason that no valid application of HOU could result in a previously lambda-bound variable becoming free. We need no 'free variable constraints' given this mechanism.

```
isOfType(he,iota(λx.Restriction(x) & Body(x))),
defined(DE,iota(λx.Restriction(x) & Body(x))).
```

The predicate `defined` will succeed if there is already a discourse referent defined in terms of this iota description. If there is not it will create one (essentially a new Skolem constant) and identify it with the iota term, asserting the definition. It is thus not a strictly logical predicate, but is necessary for thoroughly familiar reasons.

This inference rule will handle well-known Netherlandish examples like:

6 A man walked in a park. He whistled

or:

7 Smith owned a computer. It disappeared.

with the antecedent sentence in the latter being resolved as:

```
exists(computer,exists1(λe.λx.pos(past(own(e,smith,x))))
```

(We will turn below to the resolution of quantified noun phrases, but for now will just assume the appropriate resolved forms.) The pronoun 'it' here will be interpreted as (say) `i1`, equivalent to:

```
iota(λx.computer(x) & exists1(λe.pos(past(own(e,smith,x)))
```

giving an RLF `exists1(λf.pos(past(disappear(f,i1)))`. Thus 'it', in this context, is interpreted, roughly, as 'the computer that figures in the eventuality of being owned by Smith'. (In the simple cases covered here, uniqueness of the eventuality and thus of the denotation of the iota term are not actually guaranteed: in a fuller treatment of tense and aspect the eventuality described by the sentence will be uniquely identified and thus this problem will not arise.)

The quantifier $\texttt{exists}_{\texttt{(e>t)>(e>t)>t}}$ is here the translation of the indefinite article, although as is well known this is not the only alternative. We could encode DRT-like analyses directly via an equivalence creating a new discourse referent for an indefinite. On such an analysis the earlier pronoun equivalence would apply to this discourse referent just as for a proper name, provided the appropriate number and gender information was available.

By extending the definition of `possibleAntecedent`, we can combine with Webber's approach aspects of the DRT theory of plurals (Kamp and Reyle, 1993) to account for examples like:

8 Every manager liked Smith. They admired him.

```
possibleAntecedent(Context,they,DE)
  if
    Context = Pred(forall(Restriction,Body)),
```

```
    isOfType(they,sigma(λx.Restriction(x) & Body(x))),
    define(DE,sigma(λx.Restriction(x) & Body(x))).
```

The antecedent sentence will be resolved to:

```
forall(manager,λx.exists1(λe.pos(past(like(e,x,smith)))))
```

Here 'they' will be interpreted as `i2`, equivalent to:

```
sigma(λx.manager(x) & exists1(λe.pos(past(like(e,x,smith)))))
```

where $\texttt{sigma}_{\texttt{(e>t)>e}}$ is an operator meaning 'the maximal set of things satisfying both restriction and body'. To get the details completely right we would need to add some extra machinery to our existing logic to model the distinction between singular and plural, but there is no problem of principle in doing so, nor of extending to cases where universals[6] scope over existentials:

```
possibleAntecedent(Context,they,DE)
  if
    Context = Pred(forall(R1,λx.__(exists(R2,Body))),
    isOfType(they,sigma(λx.R2(x) & exists(R1,Body)),
    define(DE,sigma(λx.R2(x) & exists(R1,Body)).
```

9 Every manager owns a computer. NLPcom supplied them.

The antecedent sentence is resolved, on the relevant scoping, to:

```
forall(manager,λx.(exists(computer,λy.exists1(λe.pos(pres(own(e,x,y)))))))
```

The pronoun is resolved, on the relevant interpretation:

```
them = i3 = sigma(λx.computer(x) & exists(manager,λy.exists1(λe.own(e,y,x))))
```

Using HOU we can easily formulate many more inferences which create new discourse entities. For example, plurals can be created by assembling terms from individuals mentioned in the context. Define '+' as a functor creating plural individuals from its arguments:

```
possibleAntecedent(Context,they,DE)
  if
    Context = Pred_{e=>e=>t}(X,Y),
    isOfType(heSheOrIt,X),
    isOfType(heSheOrIt,Y),
    define(DE,X+Y).
```

Now we can interpret sequences like:

10 Smith sells the machines to NLPCom. They have a contract

6 Just as with the original analysis, we will need to write different equivalences for the case where two or more universals are involved, which is a little clumsy.

On one interpretation, 'they' will be interpreted as the complex individual `smith+nlpcom`. Depending on the approach taken towards phenomena like collective/distributive predication, this construct may be taken as the QLF or the RLF corresponding to NP conjunction. In the former case it may need further contextual resolution: again this depends on whether these distinctions are matters of resolution or inference from a resolved logical form.

Note that all of the equivalences are reversible: in the generation direction those which assume quantified NP antecedents presuppose that the RLF contains a discourse entity (or, with minimal change, a sigma or iota term).

It is also possible to give a plausible analysis within this framework of more difficult phenomena like 'donkey' sentences and dependent plurals, but the details would take us too far afield.

3.2 Definite Descriptions

We can implement a simple Russellian theory of definite descriptions by means of the following equivalence:

The
$\texttt{Pred}_{\texttt{e=>t}}(\texttt{the}_{\texttt{(e=>t)=>e}}(\texttt{Restr})) \Leftrightarrow \texttt{Pred}(\texttt{Ref}_{\texttt{e}})$
if
`salientContext(definite,Context)`
`possibleAntecedent(Context,the,Ref),`
`unique(Ref,Restr)`

where `unique(Ref,Restr)` is defined so as to be true if `Ref` is the only thing in the local context that satisfies `Restr`.

Clearly, `possibleAntecedent` will be largely the same for definites as for pronouns, although there will have to be provision for inferred antecedents ('a car ... the steering wheel'). In particular, we need to be able to create discourse entities for quantified antecedents, plurals, etc. which are analogous to those we have been discussing for pronouns.

11 Smith bought a computer. The computer disappeared.
Smith hired Jones. The managers wrote a report.
Every manager uses a computer. The managers.../ The computers...

If we allow the expressions created by `possibleAntecedent` identifying discourse referents with iota terms to figure as elements of the context (sentences of the form `defined(DiscRef,IotaTerm)`), then an initially surprising but rather natural consequence of this formulation of the definite equivalence emerges. The resolved logical form for a sentence containing a definite will have either a normal constant (a name) or a discourse referent as the equivalent of the definite description. When we try to generate back out from the resolved logical form name constants will be expressed either as names or pronouns. Discourse referents will not give rise to sentences with names, since the discourse referents are not entries in our lexicon. But since the expressions which equate them with iota or other terms will be possible contexts, the equivalence above will generate both the original definite description, and a fuller one which restates the whole content of the iota term. We can illustrate this informally with the sequence:

12 Smith bought a computer. The computer disappeared.

In processing the definite description we will use our version of Webber's rule embodied in 'possibleAntecedent' to create a discourse referent, say i1, from the existential quantifier arising from the indefinite description 'a computer' in the RLF acting as the context. If the condition 'unique(i1,computer)' succeeds, which it will, then a side effect of resolving the definite will be to add to the context the definition:

```
defined(i1,iota(λx.computer(x) & exists1(λe.pos(past(buy(e,smith,x)))))
```

The RLF for the sentence will be:

```
exists1(λe.pos(past(disappear(e,i1)))
```

In trying to generate a paraphrase of this resolved logical form several equivalences can apply. We can realise 'i1' as a pronoun by the pronoun equivalence earlier. We can also realise 'i1' as a definite 'the computer' by using the definite description equivalence with the same variable instantiations as were just used in the interpretation direction. One QLF produced by the equivalence with the definition above as the value of the Context variable will have as the QLF for the subject of `disappeared` the term `the(λx.computer(x) & exists1(λe. pos(past(buy(e,smith,x))))`, because 'i1' and this lambda expression will be one of the solutions to the condition `unique(Ref,Pred)` in the equivalence. It so happens in the current grammar that this QLF can be realised as an NP with a tensed relative clause and and so another paraphrase of the resolved logical form will be:

13 The computer that Smith bought disappeared.

In fact, similar effects can be obtained for pronouns if some small tweaks to the relevant conditions are made: this turns out to be more than a neat trick, and is very useful in providing informative feedback on what resolutions have been chosen, when developing the system, or in the context of an application.

3.3 Ellipsis

Not surprisingly, we can adapt a version of the HOU approach to ellipsis resolution (Dalrymple, Shieber, and Pereira, 1991; Pulman, 1991; Gawron, 1992; Gawron, 1995) very easily within this framework. On the DSP approach to VP ellipsis, an elliptical sentence like:

14 John likes fish and Mary does too

will be analysed as follows. Firstly (ignoring tense, 'too', etc.) we represent the meaning of the elliptical conjunct with a free variable applied to the subject:

```
Ellipsis(mary)
```

Secondly we locate an element in the antecedent `like(john,fish)` which is parallel

to `mary`, namely `john`. Next we construct a HOU equation:

```
Ellipsis(john) = like(john,fish)
```

the relevant solution to which instantiates the Ellipsis variable to λ`x.like(x,fish)`, which when substituted in the elliptical phrase yields the correct result.

Our analysis is similar in spirit. However, at QLF we represent the semantics of the elliptical sentence not with a free variable but by using the construct $\texttt{vpEllipsis}_{\texttt{ev>e>t}}$. The interpretation of this construct is given by the relevant equivalence:

vp-ellipsis
$\texttt{X}_{\texttt{t>t}}$(exists1($\lambda$e.(PolarityTenseEtc$_{\texttt{t>t}}$(vpEllipsis(e,Subject)))))
$\Leftrightarrow$ $\texttt{X}_{\texttt{t>t}}$(exists1($\lambda$e.(PolarityTenseEtc$_{\texttt{t>t}}$(Predicate(e,Subject)))))
if
salientContext(vpEllipsis,Context),
Context = Y(exists1(λf.(CPolarityTenseEtc(Predicate(f,CSubject))))),
parallel(Subject,CSubject)

The predicate `parallel` implements the use of parallelism in this analysis. Note that the equation in the second line of the equivalence uses HOU to simultaneously suggest candidates for parallel elements *and* the value of the `Predicate` which corresponds to the Ellipsis variable in the description of DSP above.

Some sortal conditions within both the equivalence and the predicate `parallel` are necessary to make sure that the variables `PolarityTenseEtc`, etc. are appropriately instantiated, since there are many possibilities consistent with their type requirements. But no extra-logical mechanisms are needed, apart from the definition of 'parallel', which is intended to correspond to the notion discussed in (Dalrymple, Shieber, and Pereira, 1991; Prüst, 1992; Hobbs and Kehler, 1997; Gardent and Kohlhase, 1997) etc. Parallelism may involve syntactic, semantic, pragmatic and discourse components, depending on the construction involved. VP deletion, for example, requires the parallel element to be the subject of the preceding conjunct as well as being in the same domain of quantification as the remnant; whereas phrasal ellipsis like '... and John' merely requires the antecedent to be in the same domain of quantification.

Now given a sequence like:

15 Smith liked Sandy. Jones didn't.

where the antecedent logical form and the QLF to be resolved are respectively:

```
exists1(λe.pos(past(like(e,smith,sandy)))
neg(exists1(λf.past(vpell(f,jones)))
```

the variables in the `vp-ellipsis` equivalence will be instantiated thus:

vp-ellipsis
$\texttt{X}_{\texttt{t>t}}$(exists1($\lambda$e.(PolarityTenseEtc$_{\texttt{t>t}}$(vpEllipsis(e,Subject)))))
$\Leftrightarrow$ $\texttt{X}_{\texttt{t>t}}$(exists1($\lambda$e.(PolarityTenseEtc$_{\texttt{t>t}}$(Predicate(e,Subject)))))
if
salientContext(vpEllipsis,Context),
% Context = exists1(λe.pos(past(like(e,smith,sandy)))
% X = identity function of type t>t
Context = Y(exists1(λf.(CPolarityCTenseEtc(Predicate(f,CSubject))))),

```
        % PolarityTenseEtc = λx.pos(past(x))
        % Y=neg, CPolarityCTenseEtc = λx.past(x)
        % Predicate = λg.λs.like(g,s,sandy)
        % Subject, CSubject = jones, smith respectively
      parallel(Subject,CSubjectC)
```

When the `Predicate` is applied to the event and subject args on the right hand side of the equivalence, the correct interpretation is obtained.

Again, this is completely reversible. If we were instead generating from the sequence of logical forms:

```
exists1(λe.pos(past(like(e,smith,sandy)))

neg(exists1(λf.past(like(f,jones,sandy)))
```

the equivalence, and the associated conditions can apply in the same way to licence a QLF with the `vpEllipsis` construct in it, causing an elliptical sentence to be generated from that QLF.

3.4 Focus

Let us turn now to examples involving focus-sensitive adverbs. (A more elaborate treatment of a wider range of focus phenomena within the current framework can be found in (Pulman, 1997b). An extension of some of these analyses can be found in (Gardent and Kohlhase, 1996a)). We will illustrate with the adverb 'too', as in:

16 a Smith likes Sandy. Jones likes Sandy too.
   b Smith likes Sandy. Jones does too.

To a first approximation, use of 'too' is appropriate if the sentence asserts that something similar but not identical to a previous event or state occurred. The two sentences must share some information, or it must be parallel, and must differ on at least one point: the different components are focused, in the sense that the main stress when the sentence is spoken will fall on those constituents:'Jones' in the example). We encode this analysis by treating 'too' as a QLF construct which takes as arguments the meaning of the focused constituent and the meaning of the sentence itself (without 'too'). The information structure requirements on 'too' are (partly) captured in the conditions on the following equivalence which interprets this QLF construct: the instantiation of the 'Shared' variable will contain what is similar but this the 'Focus' states what is different. However, as for ellipsis, the focused constituent must be parallel to the corresponding item in the antecedent expression.

Too-focus

```
Rest1_{t=>t}(exists1 λe.Rest_{t=>t}(too(Focus,Pred(e,Focus))))
⇔ Rest1_{t=>t}(exists1 λe.Rest_{t=>t}(Pred(e,Focus))))
    if
     salientContext(too-focus,Context),
     Context = _X(exists1(λf.Shared(f,Ante_Type),
     parallel(Shared,Pred),
     parallel(Ante_Type,Focus_Type),
     not(Ante=Focus)
```

Since focus can fall on almost any constituent in a sentence, the type of the `Focus` variable, and consequently that of `too` are not completely fixed. (Note that this polymorphism means that in the case where the type of a term cannot be inferred before HOU, some preprocessing to instantiate the type variable with likely candidates may be required, since the HOU algorithm requires the inputs to be fully typed.)

In processing the second sentence of 16a, variables will be instantiated as follows:

```
QLF:exists1(λe.too(jones,pos(pres(like(e,jones,sandy)))))

Too-focus
  Rest1t=>t(exists1 λe.Restt=>t(too(Focus,Pred(e,Focus))))
       ⇔ Rest1t=>t(exists1 λe.Restt=>t(Pred(e,Focus))))
       if
        salientContext(too-focus,Context),
           % Context = exists1(λf.pos(pres(like(f,smith,sandy))))
           % Rest1,Rest = identity
        Context = _X(exists1(λf.Shared(f,Antetype),
           % Shared = λg.λx.pos(pres(like(e,x,sandy)))
           % Pred = λh.λy.pos(pres(like(h,y,sandy)))
           % Ante = smith, Focus = jones
        parallel(Shared,Pred),
        parallel(Antetype,Focustype),
        not(Ante=Focus)


RLF:exists1(λh.pos(pres(like(h,jones,sandy))))
```

On our analysis, 'too' adds nothing to the truth conditions of an utterance, but merely serves to compare and contrast with the context.

Notice that Too-focus will also apply in 16b along with VPEllipsis. The order of application of equivalences is in general not significant, except that one order may be computationally more efficient than another. In this case both orders of application result in the same interpretation. The exception to this is that the equivalences for interpreting unscoped quantified NPs, described below, may apply several times in a sentence containing more than one quantifier and different orders of application will correspond to different scopings, if these are permitted by the contextual conditions.

Now consider an example in which we will assume that the focussed element cannot be determined from the linguistic form, and is thus represented at QLF by a free variable. It is of course important for computational purposes not to be committed to an analysis of focus which requires it to be overtly marked in a sentence, for essentially the same reason that it is important not to require quantifier scopes to be explicitly represented: combinatorial explosion. Narrow focus can be marked virtually anywhere in a sentence and to treat a sentence with no apparent focus marking as ambiguous between all possible focus markings would be computationally disastrous (as well as not very plausible psychologically).

Assume that we are analysing the same sequence as before (16a), but that the focus is not marked intonationally. The QLF will be:

```
exists1(λe.pos(pres(too(Focus,like(e,jones,sandy)))))
```

Applying the Too-focus equivalence will instantiate the variables in it as before, except that since `Focus` is not instantiated there would, in the absence of any constraints from context, be multiple solutions for it (and hence for `Pred`), in which `Focus` is instantiated to any constituent of the sentence. However, if the contextual conditions in the equivalence are to be satisfied then only the solution on which `Focus = jones` will be found, for on all the others it will be impossible to find a value for `Shared` and `Ante` which meet the various requirements of parallelism and non-identity. This corresponds with the observation that if focus is explicitly marked in the 'too' sentence, it must fall on 'Jones', for no other choice is coherent (in that context).

17 a Smith likes Sandy. ??Jones likes SANDY, too
b Smith likes Sandy. ??Jones LIKES Sandy, too

To conclude the focus examples, we illustrate the way that higher order unification and abduction can work together to add something to the context, in those cases where there is a context, but where it does not at first sight support the appropriate use of a focussing device. Consider the following sequence in a context where the hearer happens not to know that an iMac is a computer.

18 a Jones bought an iMac
b Smith has a new computer too

We will simplify the QLF to:

`exists1(λe.pos(pres(too(smith,have(e,smith,a-new-computer))))`

In attempting to satisfy the conditions for the Too-focus equivalence we will not be able, let us assume, to find a suitable 'Context' to serve as an expression providing an antecedent, nor be able to solve the equation `Context = _X(exists1(λf.Shared(f,Ante`$_{\text{Type}}$`))`. However, we do know the values of `Pred=λg.λx.pos(pres(have(g,x,a-new-computer)))` and `Focus=smith`.

Recall that we are assuming that the various conditions in equivalences are goals to be proved in the manner of Prolog or similar inference methods. Thus a predicate like 'parallel' can be called with or without arguments instantiated. Defining such a predicate in the general case is not trivial, but it is clear that one clause in its definition should be `parallel(A,A)`. Other clauses should search in the context for entities of similar type and sortal status to any instantiated arguments. Thus if we call `parallel(A,smith)` or `parallel(B,Pred)` we will get back as solutions (among others, perhaps) that `A = jones` and that `B=Pred=λg.λx.pos(pres(have(g,x,a-new-computer)))`. These instantiations will also instantiate the Context variable via the equations in the equivalence, to:

`exists1(λe.pos(pres(have(e,jones,a-new-computer))))`

By an abductive step we can add this to the context as an implicature or 'accommodation' that is needed to make sense of the focus structure of the sequence 18.

The relation between utterance, focus and context is such that either of the latter two relata can be incompletely specified without preventing interpretation of the former. Any analysis of focus should be able to capture this phenomenon. The vital ingredient of the analysis here is the non-directionality of inference provided by higher order unification, supplemented by abduction.

3.5 Quantifier Scope

We can implement a deductive theory of quantifier scope using the conditional equivalence mechanism. The version proposed here combines a basic insight from (Lewin, 1990) with higher order unification to give an analysis that has a strong resemblance to that proposed in (Pereira, 1990; Pereira, 1991), with some differences that are commented on below. Like Pereira's approach, it avoids the need for a 'free variable constraint', nor does it need the explicit recursion on the quantifier restriction imposed by Lewin.

We analyse quantified NPs at the QLF level as illustrated in the QLF for:

19 Every manager uses a computer

```
exists1(λe.pos(pres(use(e,every(e>t)>e(manager),a(e>t)>e(computer))))
```

We assume that every determiner has its own equivalence which resolves it as a quantifier: sometimes this can be quite a complicated matter, as with 'any' (Alshawi, 1990), which will resolve in different ways depending on its linguistic context, but here we avoid this complexity. [7]

The equivalence for 'every' is:

Every

```
Rest(t>t)(Pred(e>t)(every(Nom(e>t)))t ⇔ Rest(t>t)(forall(Nom(e>t),Pred(e>t))t
    if
     salientContext(quant,Context),
     scopeIsLicensed...
```

The final condition is a place-holder to allow for the encoding of whatever structural constraints and preferences on quantifier scopes are thought to be necessary.

Applying the equivalence to the QLF above gives us this solution:

```
Rest(t>t)(Pred(e>t)(every(Nom(e>t)))t ⇔ Rest(t>t)(forall(Nom(e>t),Pred(e>t))t
    if
        % Rest = identity
        % Pred = λx.exists1(λe.pos(pres(own,e,x,a(computer))))
        % Nom = manager
     salientContext(quant,Context),
     scopeIsLicensed...
```

7 Separate equivalences might also make it easier to encode determiner specific preferences, such as that of 'each' for wide scope. A referee points out that the lack of any explicit ordering of application of equivalences makes one natural way of doing this unavailable. But I am not convinced that this would have been the right way in any case. These preferences are just that, not hard and fast rules, and so we need to be able to permit all permutations where the context, or the structure, prefers the less frequent interpretation, as in examples like the following, (from the LOB corpus) where the most salient reading is that in which 'a bird' outscopes 'each':

Out of a total of 100 marks which are to be allocated, 15 are awarded for these attributes, and it has to be remembered that a bird has to earn each one of them when on the judging bench.

The RLF is:

```
forall(manager,λx.exists1(λe.pos(pres(own(e,x,a(computer))))))
```

This still contains the QLF construct 'a' and so the analogous equivalence for 'a' (which we will continue to treat as a quantifier here for illustration) can apply:

A
```
Rest_{t>t}(Pred_{e>t}(a(Nom_{e>t})))_t ⇔ Rest_{t>t}(exists(Nom_{e>t},Pred_{e>t}))_t
     if
         % Rest = identity
         % Pred = λy.forall(manager,λx.exists1(λe.pos(pres(own(e,x,y)))))
         % Nom = computer
      salientContext(quant,Context),
      scopeIsLicensed...
```

The final RLF is then:

```
exists(computer,λy.forall(manager,λx.exists1(λe.pos(pres(own(e,x,y))))))
```

provided that the scoping constraints and the context permit this interpretation.

The ordering of equivalences is not fixed: they simply apply non-deterministically as permitted by the relevant contextual conditions. We could have applied the two quantifier equivalences in a different order, leading to the alternative partial and then full scoping:

```
   exists(computer, λy.exists1(λe.pos(pres(own(e,every(manager),y)))))
forall(manager,λx.exists(computer,λy.exists1(λe.pos(pres(use(e,x,y))))))
```

This is a somewhat incomplete treatment of the relationship between events and quantifier scope, of course.

Because of the particular logical syntax we are using, we need to add another version of the quantifier equivalences to allow for application inside the restriction of the body of an already scoped quantifier:

Every2
```
  Rest_{(e>t)>t}(λx.Pred_{e>e>t}(x,every(Nom))) ⇔ Rest_{(e>t)>t}(λx.forall(Nom,λy.Pred_{e>e>t}(x,y)))
     if
      salientContext(quant,Context),
      scopeIsLicensed...
```

We could avoid this inelegance by using polymorphism in the equivalences, or some 'syntactic sugar' in the logical forms to produce a more uniform representation. As will be seen below, we need in any case something like the 'pair quantifier' notation of (Dalrymple, Shieber, and Pereira, 1991), which would also solve this problem. With this addition we are able to produce both scopings for examples like:

20 Every manager in some company disappeared

This is a rather over-simplified treatment of quantifier scope, which we will refine a little shortly, but even as it stands the treatment has several advantages:

(i) in classic examples like:

21 Every representative in a company saw most samples

only the available five relative scopings of the quantifiers are produced (Hobbs and Shieber, 1987), p47 but without the need for a free variable constraint - the HOU algorithm will not produce any solutions in which a previously bound variable becomes free,

(ii) the equivalences are reversible, and thus the above sentences can be generated from scoped logical forms;

(iii) partial scopings are permitted (see (Reyle, 1996))

(iv) scoping can be freely interleaved with other types of reference resolution;

(v) unscoped or partially scoped forms are available for inference or for generation at every stage.

3.6 Comparison with 'Deductive Interpretation'

It is interesting to compare this analysis with that described in (Dalrymple, Shieber, and Pereira, 1991; Pereira, 1990; Pereira, 1991). Recall that in their treatment, quantified noun phrases are treated in two stages: firstly, what they call a 'free variable' of type 'e' is introduced in the NP position, with an associated 'quantifier assumption' which is added as a kind of premise. At a later stage the quantifier assumption is 'discharged', capturing all occurrences of the free variable. Thus their analysis of something like 'every manager disappeared' would proceed as follows:

every manager	= every(x,manager(x)) ⊢ x
disappeared	= disappear
every manager disappeared	= every(x,manager(x)) ⊢ disappear(x)
- discharge the assumption:	= every(x,manager(x),disappear(x))

In the final logical form I am using an informal representation of their 'pair' notation for generalised quantifiers, which uses the same variable in both the restriction and the body, unlike the one we have been using. If we make a comparison between the way phenomena like antecedent contained deletion are treated in our two frameworks we can see that we also need such a notational change.

DSP's analysis of the relevant antecedent contained deletion cases goes like this:

22 a John greeted every person when Bill did
b John greeted every person that Bill did

In 22a, the context for the ellipsis is the first conjunct, analysed as:

```
every x person(x) ⊢ greet(john,x) when P(bill)
```

The equation is `P(john)=greet(john,x)`, with P = λ`z.greet(z,x)`. The interpretation for the whole sentence is now:

```
every x person(x) ⊢ greet(john,x) when greet(bill,x)
```

When the quantifier is discharged both occurrences of the variable x are bound:

```
every(x,person(x), greet(john,x) when greet(bill,x))
```

If the quant is discharged first, then the context for the ellipsis is:

```
every(x,person(x),greet(john,x))
```

and the equation is:

```
every(x,person(x),greet(john,x)) = P(john)
```

with P = λz.every(x,person(x),greet(z,x))
Now the interpretation for the whole sentence is:

```
every(x,person(x),greet(john,x)) when every(x,person(x),greet(bill,x))
```

For 22b, the QLF is

```
every x person(x) & P(bill) ⊢ greet(john,x)
```

The equation is: `P(john)=greet(john,x)`,with P = λz.greet(z,x), immediately giving the right result. If the quantifier is discharged first, then the QLF is

```
⊢ every(x,person(x) & P(bill),greet(john,x))
```

But now the equation will be

```
P(john) = ⊢ every(x,person(x) & P(bill),greet(john,x))
```

which is invalid because P is on both sides, leading to an 'occurs check' violation.

Note that there is a somewhat uneasy mixture of logic and meta-logic involved in the DSP analysis, caused by merging the deductive, assumption-based reasoning, and straight higher order unification. When the quantifier is undischarged, the associated so-called 'free' variable is not treated as such when solving the HOU equations. It has to be treated as a (unique) constant in order to get the right result. But when the quantifier has been discharged, all occurrences of this constant are treated as bound variables. We have to assume that the HOU algorithm has to be told what status particular occurrences of the variable actually have, because their analysis involves applications of HOU under both guises.[8]

In our system we run into some of the same problems as DSP, but from a slightly different perspective. The analogous QLF, represented in a simplified form akin to that in DSP for ease of comparison, for 22a is:

```
when(greet(john,every(person)), VPELL(bill))
```

If we use the first, unscoped, conjunct as context for the ellipsis, then we get the right result:

```
VPELL(john) = greet(john,every(person)), VPELL= λy.greet(y,every(person))
when(greet(john,every(person))),greet(bill,every(person)))
```

After scoping the two conjuncts we will get the reading on which the greetings are independent. If we had scoped the QLF first we would get:

```
forall(person,λx.when(greet(john,x),VPELL(bill)))
```

The equation we need is `VPELL(john) = greet(john,x)`, which also requires us to treat the `x` as a constant. However, it is plausible to assume that for sentence

8 Pereira acknowledges this elsewhere: (Pereira, 1991):footnote 3 'The direct replacement of ellipsis equation solutions into derivations and subsequent normalisation of the result involve some abuse of the formalism...'

internal ellipsis we will need a different control regime for application of equivalences. So far we have been assuming that each equivalence applies to an entire QLF, but for cases where some earlier portion of the sentence is acting as the context then this will not be possible. Assuming instead that a recursive traversal of the QLF is involved, then this particular unification equation will be formed and solved entirely within the scope of the lambda binding `x`. In terms of Huet's algorithm then `x` will count as a 'rigid' variable, i.e. it will be semantically like a constant.

For 22b, our QLF is

`greet(john,every(λx.person(x) & VPELL(bill)))`

With the 'every' unscoped, there is no choice of context which contains a parallel element to 'bill' that does not also contain the VPELL functor, leading to an infinite regress, analogous to the 'occurs check' failure in DSP. Thus we correctly cannot produce the unavailable reading. However, if we try to resolve the ellipsis after scoping, we have:

`forall(λx.person(x) & VPELL(bill),λy.greet(john,y))`

This will not succeed either, because the choice of context will have to be `greet(john,y)`, but in this expression, 'y' really is free and so we do not have a valid equation. We would have to adopt Pereira's 'pair' notation for our quantifiers in order to make sure that the equation was valid:

`forall(λx.⟨ person(x) & VPELL(bill),greet(john,x)⟩)`

Now everything is taking place within the scope of the lambda, as above. Note that this chain of reasoning is, mutatis mutandis, exactly the same motivation for DSP's use of the pair notation.

Given our different control regime for the application of equivalences, and the pair quantifier notation, we appear to avoid the need for the sleight of hand involved in the dual nature of DSP's assumption variables. However, what is arguably the same problem in a different guise occurs when we examine the interaction of our scoping equivalence and that for sentence internal pronoun reference given earlier. Recall that this equivalence will identify a pronoun with any term of type 'e' elsewhere in the QLF provided the usual binding and agreement conditions are met. Unfortunately, in our analysis, quantified NPs are also of type 'e' and thus we will produce invalid interpretations in cases where the equivalence applies before the quantifier has been scoped. Thus as well as the correct bound variable interpretation for a sentence like 'every manager likes his secretary' we will also produce the structure corresponding to 'every manager likes every manager's secretary'. What we need is some way of capturing the fact that NPs like 'every manager', although of type 'e', have to be treated differently than NPs like 'Smith'. A simple way of doing this would be to introduce sub-types of e, so that both types of NP would still be of type e but they could be distinguished where necessary. The Pron-intra equivalences would then be restricted to apply only to names or variables. This has the advantage that we still stay within the same logical framework ((Kohlhase and Pfenning, 1993) show how to extend HOU to

accommodate subtypes), although of course we are really using subtypes here to record a syntactic distinction that has been erased in the course of constructing the QLF.

4 An Implementation

A small implementation which (with the exception of the examples just discussed, and those needing abduction) covers all of the phenomena described so far has been developed. It uses a simple unification grammar (based on the formalism described in (Pulman, 1996)) to produce QLFs. The same grammar is used in generation to produce sentences from QLFs.

Equivalences are interpreted using a Prolog implementation of Huet's algorithm for higher order unification, with some additional heuristics to bound search in the case where terms of high order are encountered. The implementation is aimed at clarity rather than efficiency but is still not disastrously inefficient (rather to my surprise, I might add). The whole process of parsing, resolving, and generating a paraphrase of the resolved LF for the following little text takes about 30 seconds on a 300 MHz laptop PC. (`X^^Body` is the notation for λ`x.Body`. Upper case in the input (or output) corresponds to narrow focus intonation).

```
Smith hired Sandy.
They wrote a report.
Jones read it.
He liked the report.
He is a manager.
Sandy likes him.
Smith doesn't.
HE likes Roberts.
SANDY does too.
```

Working through the examples we show the input sentence; the (first) QLF found for it; the (first) RLF found for the QLF given the context (usually just the preceding RLF); and as full as possible a paraphrase of the RLF which we get by reversing the equivalences and applying them in a null context to obtain a QLF which we then generate from. Note that in this implementation the existentially quantified event variable has been Skolemised.

```
> Input:   Smith hired Sandy.
QLF:pos(past(hire(e0,smith,sandy)))
RLF:pos(past(hire(e0,smith,sandy)))
Resolved as:   smith hired sandy.
```

No resolution is needed in this example, since there is no preceding context.

```
> Input:   They wrote a report.
QLF:pos(past(write(e1,they,a(report))))
RLF:exists(report,A^^pos(past(write(e1,npand(smith,sandy),A))))
Resolved as:   smith and sandy wrote a report.
```

`npand` is the '+' operator described in the text. It corresponds to one QLF construct for NP conjunction, hence the informative paraphrase. If we ask for more paraphrases with the previous sentence serving as a context we get:

```
he and sandy wrote a report.
he and sandy wrote some report.
he and she wrote a report.
he and she wrote some report.
smith and sandy wrote some report.
smith and she wrote a report.
smith and she wrote some report.
```

('Some' is treated as synonymous with 'a', which is not quite correct).

```
> Input:   Jones read it.
QLF:pos(past(read(e2,jones,it)))
RLF:pos(past(read(e2,jones,i3)))
Resolved as:   jones read the report that smith and sandy wrote.
```

In this version we have incorporated the tweaks to allow for informative paraphrases of pronouns described earlier. 'i3' is a Webber-style discourse referent corresponding to 'a report'. It is identified with an iota term which supports the use of the definite in paraphrasing the resolved LF. Further paraphrases reveal the well-known problem that generated sentences may be ambiguous in a potentially confusing way:

```
jones read the report that he and sandy wrote.
jones read the report that he and she wrote.
jones read the report that smith and she wrote.

> Input:   He liked the report.
QLF:pos(past(like(e4,he,the(report))))
RLF:pos(past(like(e4,jones,i3)))
Resolved as:   jones liked the report that smith and sandy wrote.
```

The same paraphrasing behaviour happens more or less automatically for definites. Alternative paraphrases for the RLF should include 'Jones liked it', and 'Jones liked the report'.

```
> Input:   He is a manager.
QLF:pos(pres(be(e5,he,a(manager))))
RLF:exists(manager,A^^pos(pres(be(e5,jones,A))))
Resolved as:   jones is a manager.

> Input:   Sandy likes him.
QLF:pos(pres(like(e6,sandy,he)))
RLF:pos(pres(like(e6,sandy,jones)))
Resolved as:   sandy likes jones.

> Input:   Smith doesn't.
QLF:neg(pres(vpell(e7,smith)))
RLF:neg(pres(like(e7,smith,jones)))
Resolved as:   smith doesn't like jones.
```

This is an example of simple VP ellipsis. Alternative contextualised paraphrases are:

```
smith doesn't.
smith doesn't like him.
```

Now we have set up a context in which contrastive focus is appropriate:

```
> Input:   HE likes Roberts.
QLF:focus(he,pos(pres(like(e8,he,roberts))))
RLF:focus(smith,pos(pres(like(e8,smith,roberts))))
Resolved as:   SMITH likes roberts.
```

This example shows a QLF construct (from (Pulman, 1997b)) not discussed earlier, and for which no equivalence has been written yet. Thus the input is only partly resolved and focus is retained in the paraphrase.

```
> Input:   SANDY does too.
QLF:pos(pres(too(sandy,vpell(e9,sandy))))
RLF:pos(pres(like(e9,sandy,roberts)))
Resolved as:   sandy likes roberts.
```

A combination of VP ellipsis and too-focus as described earlier. Contextualised alternatives are:

```
SANDY likes him too.
SANDY likes roberts too.
sandy does.
sandy likes him.
```

This reveals a bug somewhere, as we do not get out the sentence we put in, which should always be one of the options.

Here is the Hobbs-Shieber scope example, from a slightly differently configured version of the system in which the event variable is explicitly quantified rather than Skolemised.

```
| ?- ana([every,manager,in,some,company,owns,a,car],RLF,null:t),display_in_readable_form(RLF,LF).

% every, some, a

LF = forall(A^^exists(company,
                      B^^and(manager(A),in(A,B))),
            C^^exists(car,
                      D^^exists1(E^^pos(pres(own(E,C,D))))))
;
% some, every, a
LF = exists(company,
            A^^forall(B^^and(manager(B),in(B,A)),
                      C^^exists(car,
                                D^^exists1(E^^pos(pres(own(E,C,D)))))))
;
% a, every, some
LF = exists(car,
            A^^forall(B^^exists(company,C^^and(manager(B),in(B,C))),
                      D^^exists1(E^^pos(pres(own(E,D,A))))))
;
% some, a, every
LF = exists(company,
            A^^exists(car,
                      B^^forall(C^^and(manager(C),in(C,A)),
                                D^^exists1(E^^pos(pres(own(E,D,B)))))))
;
% a, some, every
LF = exists(car,
            A^^exists(company,
                      B^^forall(C^^and(manager(C),in(C,B)),
                                D^^exists1(E^^pos(pres(own(E,D,A)))))),
```

```
;

no
```

The missing combination which is correctly excluded is `every - a - some`.

Clearly, these are small beginnings. But this small scale implementation demonstrates that the approach is computationally viable in principle. Issues to do with how to scale up to wider coverage are addressed later.

5 Comparison with alternative approaches

5.1 Core Language Engine Quasi-Logical Form

The starting point for the approach followed here was a dissatisfaction with certain aspects of the theory of 'quasi-logical form' as described in (Alshawi, 1990; Alshawi, 1992), and implemented in SRI's Core Language Engine (CLE). In the CLE-QLF approach, as rationally reconstructed by (Alshawi and Crouch, 1992; Crouch and Pulman, 1994), the context-independent meaning of a sentence is given by one or more QLFs which are built directly from syntactic and semantic rules. Just as here, these QLFs represent the basic predicate argument structure of the sentence, and contain constructs which represent those aspects of the meaning of the sentence which are dependent on context.

The effects of contextual resolution are uniformly represented via the instantiation of 'metavariables'. This instantiation is brought about by the operation of 'resolution rules' which are essentially user-defined Prolog predicates finding appropriate instantiations for metavariables from the current context. Contextual resolution is therefore a process of adding information to an underspecified meaning representation until it is sufficiently specified for the task at hand. (In translation, for example, it need not be fully specified. For tasks like database query, it usually will have to be). This process is completely monotonic and therefore fulfils a necessary (though not a sufficient) condition for reversibility.

Some simplified examples will give the flavour of this theory. A pronoun is represented at QLF by a term containing essentially a syntactic category, an index, a restriction predicate, and a metavariable; schematically:

```
term(pro,<idx>,<restriction>,<metavrble>)
```

Thus a sentence like 'he sneezed' will, ignoring tense and aspect, be represented as follows at the QLF level; and, when the metavariable has been instantiated to the contextually preferred candidate referent, at the 'resolved quasi-logical form' (RQLF) level:

```
QLF  =  sneeze(term(pro,+1,masc,Referent))
RQLF =  sneeze(term(pro,+1,masc,john))
```

Scoping of quantifiers is also a matter of instantiating metavariables. QLF formulae containing quantifiers are prefixed by a scoping metavariable which scoping resolution rules instantiate to a list of the indices associated with quantifiers, in an order which indicates the preferred scoping:

```
QLF  =
Scope:like(term(q,every,+1,philosopher),
           term(q,some,+2,book))
```

```
RQLF =
[+1,+2]:like(term(q,every,+1,philosopher),
             term(q,some,+2,book))
     % every .... some ...

[+2,+1]:like(term(q,every,+1,philosopher),
             term(q,some,+2,book))
     % some ... every ...
```

The denotational semantics of RQLF structures involving instantiated scope and referent metavariables is given in terms of simple interpretation rules which have the effect of interpreting the quantifiers as having the scopes indicated by the lists of indices, and the pronouns as having the interpretation of the instantiation of the metavariable (in these cases at least).

Alshawi and Crouch (Alshawi and Crouch, 1992) present an illustrative first order fragment along these lines and are able to supply a coherent formal semantics for the CLE-QLFs themselves, using a technique essentially equivalent to supervaluations: a QLF is true iff all its possible RQLFs are; false iff they are all false, and undefined otherwise.

There are many good things about this approach. It has proved itself amenable to a large scale implementation of impressive coverage, generality, and relative efficiency (Alshawi, 1992). It has the theoretically desirable property of monotonicity and in practice a large degree of reversibility. In the implementation, generation can take place from the QLF level, or from resolved QLF. (Which is the appropriate level depends partly on the application: generation from QLF is all that is needed for many types of translation, for example. Generation from resolved QLF is chiefly used for checking with a user that resolution has accurately resolved contextually dependent constructs.) Furthermore, QLF has, in principle at least, a coherent formal semantics via the supervaluation technique - these are not uninterpreted representations (although the supervaluation semantics does not lead to an appropriate consequence relation, as we shall see below).

Nevertheless, there are several aspects of the theory that are not completely satisfactory. Firstly, the QLFs themselves, although they are built by technically compositional semantic rules from syntactic structures, contain many constructs which are solely motivated by the requirements of the resolution process. QLFs contain, to take the most obvious example, indices and metavariables: constructs for which there is no apparent motivation in the syntax and morphology of English.

Secondly, the semantic relation between underspecified QLFs and their further specified RQLF representations is given entirely in terms of subsumption: a QLF subsumes all its possible RQLFs. They differ syntactically only via the instantiation of metavariables, giving a particularly simple way of determining subsumption. But this notion of subsumption does not model the intuitive relationship between contextually dependent sentences and (relatively) contextually independent paraphrases that one might expect: the QLF for 'he sneezed', for example, does not subsume the RQLF of 'John sneezed' even in a context where 'he' can only be interpreted as 'John'. In fact, when the CLE generates the sen-

tence 'John sneezed' as a check that 'he sneezed' has been interpreted correctly, it does not do so from the resolved QLF corresponding to the latter. This RQLF has the form:

```
sneeze(term(pro,+23,masc,john))
```

But the QLF corresponding to 'John sneezed' is:

```
sneeze(term(name,+32,λY.name(Y,john),Referent))
```

Some inference has to take place to relate the RQLF for the interpreted sentence to a QLF which unambiguously expresses its contextualised meaning. This makes the task of expressing the output of some application system in a context-dependent way quite difficult: rather than relating this output to RQLF, it has to be related to a QLF that is sufficiently instantiated for a contextually unambiguous sentence to be generated from it. The resolution mechanism is not intended to be reversible, although by redefining resolution rules this is achievable to some extent within the limitations just discussed (Hurst, 1994).

A third problem arises with the approach to the semantics of QLFs that this notion of the relationship between QLF and RQLF encourages one to adopt: it is that taken by (Alshawi and Crouch, 1992). This describes the semantics of QLFs via a supervaluation over the semantics of the RQLFs that they subsume. Although the problem does not arise for the simple fragment they illustrate there, if it were extended to cover a wider range of constructions, it would be found that many QLFs subsumed RQLFs that are not actually permitted by the resolution rules: for example, those which can only arise via a violation of scoping or binding constraints. The role of resolution rules (for perfectly good presentational reasons) is completely ignored by their treatment. However, it is really the case that in giving the semantics of a QLF one is interested only in the set of RQLFs that are obtainable from it under closure of the resolution rules. Ideally, therefore, we would like a formal reconstruction of resolution rules as well. This is so, not just for reasons of formal hygiene in trying to make logical sense out of underspecified representations, but also because resolution rules and the knowledge they express are an important object of study in their own right. Anyone who has built a wide coverage system knows that the range of context-dependent phenomena encountered in real life is a lot wider than the preoccupations of many linguists might suggest. In the CLE, for example, contextual resolution forms a larger part of the system than do syntactic and semantic processing. Unfortunately in the CLE there is no formal theory of resolution rules, and thus no prospect of capturing their role in assigning a semantics to QLFs.

A further problem, that the supervaluation semantics does not yield the right consequence relation, is discussed further below.

The QLF-based theory illustrated in the approach advocated here does not suffer from these problems:

(i) QLFs contain only information for which there is a direct syntactic or morphological reflex. In particular, there are no indices or metavariables.

(ii) the relation between QLF and RLF is directly reversible

(iii) the semantics of QLFs is completely given by the conditional equivalences that relate them to RLFs, thus avoiding the problem of the subsumption based treatment, and the associated supervaluation semantics. (More detail on precisely how this semantic account works is given below.)

(iv) conditional equivalences are a formal language for resolution rules, thus bringing them within the scope of the theory.

5.2 Glue language

Within the LFG framework, Dalrymple and her colleagues have been working on a linear logic 'glue language' approach to semantic assembly and underspecification (Dalrymple et al., 1996). LFG distinguishes two different levels of syntactic representation: constituent structure and functional structure (f-structure) at which the basic syntactic relations are distinguished (subject, object, etc.). Semantic interpretation is also at two levels: a 'σ-projection' maps an f-structure to a σ-structure. Although σ-structures are described as semantic structures, they are not themselves meanings. Rather they are connected to meanings or logical forms via '$\rightsquigarrow$', an 'otherwise uninterpreted binary predicate symbol'. Given a σ-structure and the 'meaning constructors' associated with the lexical items in the f-structure from which it was projected, the initial semantic level is a (linear logic) conjunction of the meanings associated with the lexical items. This is approximately the equivalent of our own QLF level of representation, although there are enough different assumptions that this equivalence is not very meaningful.

From the initial level inferences can be drawn via linear logic derivations. These inferences correspond roughly to our RLFs, in that they are logical forms which can be evaluated directly for truth (I assume: this is not stated explicitly).

To illustrate, we show the derivation of the two different scopings of our earlier example:

23 Every manager uses a computer

This sentence will receive the following f-structure.

$$f:\begin{bmatrix}\text{PRED} & use & \\ \text{SUBJ} & g: & \begin{bmatrix}\text{SPEC} & every \\ \text{PRED} & manager\end{bmatrix} \\ \text{OBJ} & h: & \begin{bmatrix}\text{SPEC} & a \\ \text{PRED} & computer\end{bmatrix}\end{bmatrix}$$

The σ-projections for g introduces initially empty VAR and RESTR attributes. The lexical entry for 'every' is

$\forall$G,R,S.
($\forall$x.($\uparrow_\sigma$ VAR) $\rightsquigarrow$ x $\multimap$ ($\uparrow_\sigma$ RESTR) $\rightsquigarrow$ R(x))
$\otimes$
($\forall$x.$\uparrow_\sigma \rightsquigarrow$ x $\multimap$ G $\rightsquigarrow_t$ S(x))
$\multimap$ G $\rightsquigarrow_t$ every(R,S)

which can be paraphrased:

> If
> if the variable of the NP σ-projection this is part of is arbitrary x
> then the noun restriction of the NP σ-projection is interpreted as R(x)
> and
> if the σ-projection of the whole NP is arbitrary x
> then the scope is interpreted as S(x)
> then the scope can be (re)-interpreted as the quantification: every(R,S)

The semantic lexical entry for 'manager' will be:

$$\forall X.(\uparrow_\sigma \text{VAR}) \rightsquigarrow X \multimap (\uparrow_\sigma \text{RESTR}) \rightsquigarrow \text{manager}(X)$$

When these lexical entries are unified with the f-structure the $\uparrow_\sigma$ will be instantiated to g_σ. In the result, the entry for 'manager' unifies with the nested implication in the antecedent of the entry for 'every' allowing the deduction (by modus ponens, with substitutions $\{\langle X,x\rangle,\langle R,\text{manager}\rangle\}$ to the conclusion:

$$\begin{aligned}\forall G,S.\ &(\forall x.\ g_\sigma \rightsquigarrow x \multimap G \rightsquigarrow_t S(x))\\ &\multimap G \rightsquigarrow_t \text{every}(\text{manager},S)\end{aligned}$$

The meaning of 'a computer' is constructed analogously:

$$\begin{aligned}\forall H,Q.\ &(\forall y.\ h_\sigma \rightsquigarrow y \multimap H \rightsquigarrow_t Q(x))\\ &\multimap H \rightsquigarrow_t a(\text{computer},Q)\end{aligned}$$

The meaning for a transitive verb like 'use' is of the form:

$$\forall X,Y.(\uparrow_\sigma \text{SUBJ}) \rightsquigarrow X \otimes (\uparrow_\sigma \text{OBJ}) \rightsquigarrow Y \multimap \uparrow_\sigma \rightsquigarrow \text{use}(X,Y)$$

i.e. 'if the subject means X and the object means Y then the sentence means `use(X,Y)`'. The meaning of a sentence is obtained from the conjunction of this expression with the meanings of the subject and object. With non-quantified arguments, the meanings of the subject and object simply satisfy the antecedent of the implication allowing the consequent to be deduced, with X and Y instantiated to the subject and object meanings. In the case of quantified arguments this inference will not go through directly. Instead, the verb meaning is rewritten to one of two logically equivalent forms, where g_σ and h_σ are the subject and object meanings:

$$\text{use}_1\text{: } \forall X.\ g_\sigma \rightsquigarrow X \multimap (\forall Y.\ h_\sigma \rightsquigarrow Y \multimap f_\sigma \rightsquigarrow \text{use}(X,Y))$$

$$\text{use}_2\text{: } \forall Y.\ h_\sigma \rightsquigarrow Y \multimap (\forall X.\ g_\sigma \rightsquigarrow X \multimap f_\sigma \rightsquigarrow \text{use}(X,Y))$$

(The theoretical status of these rewriting operations is not clear: it is presumably something that happens in the lexicon, but whether it counts as a spurious ambiguity of verb meaning as in some similar categorial grammar treatments of quantifier scope is not specified). From the meaning of 'use$_1$' and the meaning of 'a computer' we can deduce the following formula, corresponding to the choice of narrow scope for 'a computer' (linear implication $-\circ$ is like $\rightarrow$ in that {q,p $\rightarrow$ (q $\rightarrow$ r) $\vdash$ p $\rightarrow$ r}).

$$\forall X.g_\sigma \rightsquigarrow X -\circ f_\sigma \rightsquigarrow_t a(manager,\lambda v.use(X,v))$$

The variable substitutions are $\{\langle H,f_\sigma\rangle,\langle Y,y\rangle,\langle Q,\lambda v.use(X,v)\rangle\}$. We can now combine this with 'every manager' to give:

$$f_\sigma \rightsquigarrow_t every(manager,\lambda u.a(manager,\lambda v.use(u,v)))$$

with substitution $\{\langle G,g_\sigma\rangle,\langle X,x\rangle,\langle S,\lambda u.a(manager,\lambda v.use(u,v)\rangle\}$.

To get the alternative scoping we combine 'every manager' with 'use$_2$' to get:

$$\forall Y.h_\sigma \rightsquigarrow Y -\circ f_\sigma \rightsquigarrow_t every(manager,\lambda u.use(u,Y))$$

which then combines with 'a computer' to give:

$$f_\sigma \rightsquigarrow_t a(computer,\lambda v.every(manager,\lambda u.use(u,v)))$$

There are several points of contact between this glue language analysis and our own:

(i) Both share the somewhat inelegant feature that a quantified noun phrase has to have a denotation of type 'e' at some level, because it is an argument of a verb. In our case, this is achieved by resolving a determiner like 'every' of type $\langle\langle e,t\rangle,e\rangle$ as a quantifier like 'all', where the scope is supplied by higher order unification, which in effect abstracts over this argument position. This means that we do not have a very plausible story to tell about the independent denotation of QLF level 'every' - it just denotes some function from noun meanings to individuals. In the glue language version, the same is true: the antecedent of the relevant implication says 'if we can assign an arbitrary meaning x of type e to the f-structure of the whole NP, ...'.

(ii) both use higher order unification: in order to assemble the correct values for the variables R, S and Q above, a higher order unification is necessary.

However, I would maintain that the QLF treatment has several distinct advantages:

(i) it uses *only* HOU: we do not need to allow verb rewriting, in particular.

(ii) it is reversible: nothing further is required to be able to generate sentences from scoped logical forms. The glue language treatment is not obviously reversible, at least in its present form.

(iii) it is sensitive to context: the conditional equivalences require the context to be an appropriate one for the scoping derived. The form of the implication in

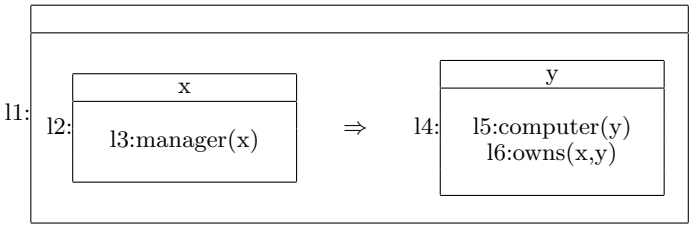
the interpretation direction is 'QLF & Context ⊢ RLF'. By contrast, if I have understood correctly, the glue language deductions as presented require only the linguistic **forms** to be present: thus all interpretations of an ambiguous form will be derivable, whatever the context. Some further specification of how context acts to eliminate impossible readings is required. [9]

5.3 Underspecified Discourse Representation Structures

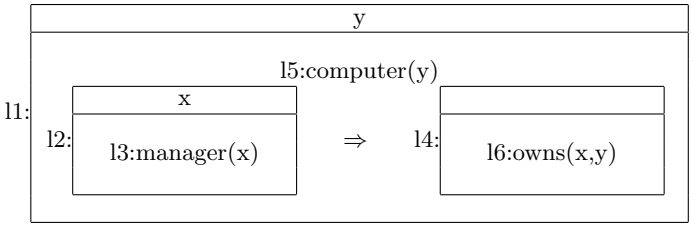
In a series of papers, Reyle (Reyle, 1993; Reyle, 1995; Reyle, 1996) has elaborated a version of DRT which is able to represent quantifier scope and other ambiguities in a single underspecified representation. (In other respects like pronoun or definite description interpretation, standard DRT is already an underspecification-based theory.) UDRT differs from standard DRT in that the familiar 'boxes' are partly replaced by a set of labels for the conditions in the boxes and the relations between them, and partial relations of inclusion between (the components indexed by) these labels. When sentences are fully scoped the representations are like standard DRT with extra labels. Thus a sentence like

24 Every manager owns a computer

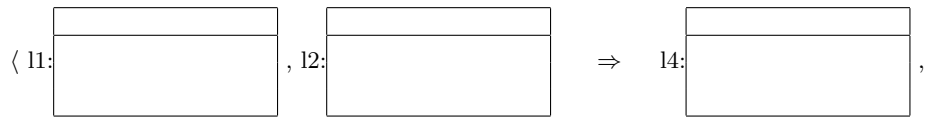
would be represented in its different scopings by:



and:



But a representation which does not specify the scoping in ambiguous cases can be given by listing the component elements, along with the inclusion ordering that determines the scoping. The components are:



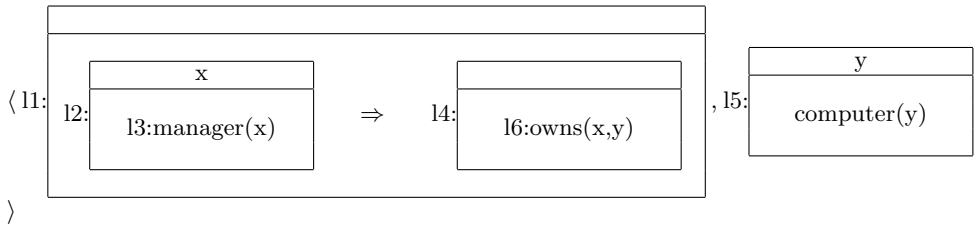
9 The glue language approach makes much of the 'resource sensitivity' of linear logic. But in the specific instances of the analysis of quantifier scope and pronouns discussed in Dalrymple et al., the linearity and resource sensitivity of the logic assumed is, as far as I can see, subverted by the device of 'reinterpreting' constructs like the 'scope' variable (cf example 28) or the 'reintroduction' of pronoun meanings (cf example 39).

l3:[manager(x)], l5:[computer(y)], l6:[owns(x,y)] ⟩

and the two orderings are:

{l2 ≺ l1, l3 ≺ l2, l4 ≺ l1, l5 ≺ l4, l6 ≺ l4}
{l2 ≺ l1, l3 ≺ l2, l4 ≺ l1, l5 ≺ l1, l6 ≺ l4}

We now represent the unspecified scoping by (roughly) the intersection of these inclusion constraints, which gives the following partial order, here determined just on the basis of the syntactic structure of the sentence. This representation leaves it unresolved as to whether the indefinite has wide or narrow scope:



Resolution of scoping consists of adding further inclusion constraints. Other than those which are the result of general principles (e.g. binding constraints) it is not part of the theory to say where these constraints come from. Just as for pronoun resolution, (U)DRT provides a representation that allows for the monotonic addition of information to do the resolution, but has nothing to say about the mechanisms that do this.

Reyle sketches various methodological requirements that should be met by a theory of meaning underspecification (Reyle, 1996), p 241ff. Firstly, it should be possible to represent partial orders of scoping relations. Secondly, it should be able in effect to emulate the DRT treatment of donkey sentences (p243) (a somewhat parochial requirement, given that the case is not yet closed on whether this treatment is correct: (Elworthy, 1995), etc). Thirdly, the theory should not need anything like the free variable constraint.

Clearly, UDRS meets these requirements, as does our own QLF-based approach. There are some similarities between the UDRS and the 'glue language' approaches, as detailed in (Crouch and van Genabith, 1997). There are also some differences: unlike our approach, or the glue language approach, UDRS does not have the problem of how to represent the meaning of quantified NPs as things of different type at different levels. However, it achieves this at the cost of not representing the meanings of quantified NPs as independent units at all: determiner and restriction are separate components that have no close connection to each other until the inclusion constraints are imposed. It remains to be seen whether this unconstrained approach to semantic assembly can be implemented on a large scale, given that it is prima facie not very compositional.

Early versions of UDRS (Reyle, 1993) treated ambiguity as disjunction, which as we shall see, is not correct. The more recent version (Reyle, 1996) remedies

this. UDRS also makes a serious attempt at developing a calculus for reasoning directly with underspecified DRSs, a necessary move: the whole point of working on underspecification is so as to be able to work with underspecified representations directly, rather than relying on their fully specified resolutions. This is a deficit in our own account (and the glue language account) which we shall begin to remedy below.

But while there are many points of contact between the two approaches, there are at least two dimensions along which I would maintain the QLF approach to be preferable:

(i) it is reversible. It may be possible to do reversible resolution within UDRT, but since the theory does not specify how to do resolution, we cannot really say one way or the other.

(ii) the representations postulated are motivated only by overt linguistic elements. UDRS shares with the CLE-QLF approach a proliferation of meta-constructs (labels, indices, ordering constraints etc) that are motivated only by the resolution process, not by the linguistic forms of sentences. It might be argued that this is an aesthetic preference rather than a substantive one, but it is likely to have consequences for both methodology and implementation: semantic assembly is surely going to be a very unconstrained and non-compositional process in this framework.

6 The Semantics of QLF

6.1 The Meaning of ⇔

As was pointed out earlier, conditional equivalences of the form:

$$\begin{array}{l}\texttt{QLF} \Leftrightarrow \texttt{RLF} \\ \quad \texttt{if} \\ \quad\quad \texttt{Condition}_1\texttt{,} \\ \quad\quad \ldots\texttt{,} \\ \quad\quad \texttt{Condition}_n\texttt{.}\end{array}$$

are logically equivalent to the conjunction:

$$\begin{array}{l}(\texttt{Conditions}_{1\ldots n}\ \texttt{\&}\ \texttt{QLF} \rightarrow \texttt{RLF}) \\ \texttt{\&} \\ (\texttt{Conditions}_{1\ldots n}\ \texttt{\&}\ \texttt{RLF} \rightarrow \texttt{QLF})\end{array}$$

if the symbol ⇔ is interpreted as material or logical equivalence. [10] We would like to preserve this interpretation, because by doing so we can claim that our conditional equivalences collectively provide a truth definition for expressions of our QLF language.

In, say, first order logic, truth is defined directly via clauses like:

```
∃x.P(x) is true iff some value of x makes P(x) true.
P & Q is true iff P is true and Q is true.
etc. etc.
```

10 I am grateful to Stanley Peters for helpful discussion of the issues in this section.

But for QLFs, truth is defined derivatively, via the truth conditions for the RLFs with which a QLF is associated via the equivalences:

QLF $\Leftrightarrow$ RLF
if
$C_1 \ldots C_n$

The RLFs themselves are assigned truth conditions directly via the usual interpretation function for a typed higher order logic of the kind we are assuming. An actual QLF may require a sequence of equivalences in order to arrive at a fully truth evaluable RLF, of course. But we can simplify by assuming that this sequence is represented as a single equivalence, because:

$(Q_1 \Leftrightarrow Q_2$ if C_1) & $(Q_2 \Leftrightarrow Q_3$ if $C_2)$ & ... $(Q_n \Leftrightarrow R$ if $C_n)$

is equivalent to:

$Q_1 \Leftrightarrow R$ if C_1 & C_2 & ... C_n

which has the same form as a single equivalence. (We ignore the possibility of abduction in this section and assume that all conditions are fully evaluable.)

So for a particular resolved QLF the truth definition induced by the equivalences will be an instance of a schema like:

if $C_1 \ldots C_n$, then P(he) is true iff P(john)
if $C_1 \ldots C_n$, then P(every(R)) is true iff forall(R,P)
etc.

Truth will be relative to a particular known context.

For an unresolved QLF, the truth definition will have to take into account all the possible contexts in which it could be resolved, and all the ways within each context that it could be resolved (there may be equally plausible choices of pronoun antecedent, for example). In the general case, there could be an infinite number of these. The number of different sequences of equivalence involved in resolutions will hopefully be bounded by the number of QLF constructs appearing in the initial QLF, and the number of valid solutions to attempts to match equivalences to them. (Unfortunately, nothing in the formal mechanism itself guarantees this. It would be perfectly possible to write equivalences that generated cycles. I am assuming - or rather, hoping - that no such analysis would be linguistically plausible). But because many of the contextual conditions are just variables over contextually available propositions, there will be no fixed upper limit on the number of valid contexts that could be considered. So the form of a truth definition for an unresolved QLF will be:

$C_1 \rightarrow$ (Q is true iff R_1) & $C_2 \rightarrow$ (Q is true iff R_2) & ... & $C_n \rightarrow$
(Q is true iff R_n).

This seems like an intuitively sound reconstruction of our basic intuitions about the meaning of constructs like pronouns or ellipses. It doesn't make much sense (unless talking theory) to ask 'what does 'he' mean?' in the abstract. But if we did, then an answer like 'Well, in this utterance context 'he' means 'John', and in this utterance context 'he' means 'Bill', etc...' is a perfectly satisfactory answer. That is essentially the form of answer that the current theory proposes. It does not assign a full meaning to QLF constructs like 'he' or 'every' in isolation, but only in a context.

However, the coherence of such an approach is dependent on how fine-grained our notion of context is made to be. For if our truth definition is defining meanings as above:

```
C1 → (Q ⇔ R1)
&
C2 → (Q ⇔ R2)
...
```

then it will follow that if, for a QLF Q in a given situation both C1 and C2 are satisfied, R1 and R2 must be equivalent. It is clear that for the case of quantifier scope (at least) there will be many examples where the same QLF appears to be capable of being resolved to two non-equivalent or even incompatible logical forms.

```
neg(leave(e,every(boy)))
```

('every boy didn't leave') can be resolved to

```
neg(every(boy,λx.leave(e,x)))
```

or:

```
every(boy,λx.neg(leave(e,x)))
```

which are not equivalent. If we are to maintain that ⇔ is interpreted as logical equivalence, then we must argue that such a situation cannot happen. Either the QLFs for these cases will be different, or there will be something in the context that means that only one interpretation can be derived. (Michael Kohlhase has pointed out to me that this is equivalent to a requirement that a set of equivalences are 'confluent' if viewed as a rewriting system.)

In the quantifier scope example above it might be, for example, that there is stress-marked focus on 'every' (and falling intonation on the VP) leading to the wide-scope 'every' interpretation. An appropriate context for this would be where what is being denied is the proposition that `some(boys,λx.neg(leave(e,x)))`. The other interpretation is most naturally associated with stress on 'didn't', forcing the negation to have wide scope. An appropriate context for this would be one in which what is being denied is the proposition `every(boy,λx.leave(e,x))`.

If the focus is overtly marked, then the two QLFs will be different in that respect and so only one interpretation will be obtained. But if the focus is underspecified, then the two contexts are still incompatible with each other, and

only one interpretation will be derived, as is required by our interpretation of the equivalences.

My assumption is that this is always the case: if the equivalences are capable of resolving the same QLF to two logically distinct RLFs, then there is no (full) context that will simultaneously support both resolutions. What this amounts to is the question of whether an utterance - an utterance, not a sentence - is ever genuinely ambiguous. There are of course cases of deliberate ambiguity for poetic or humorous effect (see (Poesio, 1996)), but I think it is legitimate to regard these as metalinguistic or parasitic on the normal case (I read (Barwise and Perry, 1983)40-41 as also taking this view). In most cases the purpose of the ambiguity is precisely to cause the audience to become aware of the two different contexts that are associated with the different interpretations. It is not the case that the real circumstances of the utterance support both of these contexts.

To summarise, on our theory, if we interpret $\Leftrightarrow$ as logical equivalence, then we are committed to the claim that no utterance (where the context is fully specified) is truly ambiguous. (This does not entail that particular speakers must be able to fully resolve all utterances). That is to say, there must be some feature of the form and content of the utterance, or the context in which it is produced, that exclude all but one of the possible interpretations.

6.2 Truth and Consequence

In (van Deemter, 1996; van Eijck and Jaspars, 1996) and (Jaspars, 1997) a set of criteria for a notion of ambiguous consequence are outlined. In the following, R_1 and R_2 are (all) the resolutions of Q, and $\models_a$ is an ambiguous consequence relation.

We can summarise these requirements as follows:

1.	Q	$\models_a$	R_1 or R_2
2.	R_1 and R_2	$\models_a$	Q
3.	$\neg$Q	$\models_a$	$\neg R_1$ or $\neg R_2$
4.	$\neg R_1$ and $\neg R_2$	$\models_a$	$\neg$Q
5.	Q	$\not\models_a$	R_1 and R_2
6.	R_1 or R_2	$\not\models_a$	Q
7.	$\neg$ Q	$\not\models_a$	$\neg R_1$ and $\neg R_2$
8.	$\neg R_1$ or $\neg R_2$	$\not\models_a$	$\neg$Q

If an ambiguous expression is true then at least one of its readings is true (1). But the stronger version, that all readings are true, is not plausible (5). This would mean that any expression with mutually contradictory readings would lead to inconsistency. (The CLE-QLF supervaluation semantics falls prey to this problem.) On the other hand, if we know both readings are true, then we can safely assert the ambiguous expression (2). If only one reading is true, we cannot assert the ambiguous expression safely (6). To do so would be to identify ambiguity with disjunction (given 1), and as van Deemter (van Deemter, 1996) points out, this is to confuse the level at which the disjunction holds: when an expression is

ambiguous, then either it means P, or it means Q. This is not the same as saying that it means either P or Q.

If we know that an ambiguous expression is false, then at least one of its interpretations must be false (3). Again, we cannot strengthen this to the conclusion that all readings are false (7) because that would lead us again to be regarding ambiguity as equivalent to disjunction: $\neg(P \vee Q) = \neg P \ \& \ \neg Q$. However, if we know that both readings are false, we can assert that the ambiguous expression is false (4), but just as before, we cannot do this on the basis of knowing that just one of the readings is false (8).

Is our account of truth for QLFs consistent with 1-8? Consider 1. In the case that R1 and R2 are the only disambiguations then the truth definition tells us that:

$$(T)\ C1 \rightarrow (Q \Leftrightarrow R1)\ \&\ C2 \rightarrow (Q \Leftrightarrow R2)$$

If we know that Q is true then, because Q can only be equivalent to one of R1 or R2, for T to hold it must be that case that either C1 holds and so Q is equivalent to R1 (and the other conjunct is vacuously true) or ditto for C2 and R2. So one of R1 or R2 will be true. Since T will hold here if only *one* of R1 or R2 is true, this shows that 5 is valid also.

Consider 2. If both possible resolutions are true, for example, in a situation in which there are two tall men, John and Bill, and thus both 'John is tall' and 'Bill is tall' are true, a QLF corresponding to 'he is tall' will count as true even if we do not know which context holds. For T to hold where both R1 and R2 are true, either the equivalence (Q iff R1), and the corresponding context, C1, must both fail, or the equivalence (Q iff R2) with context C2 must both fail. But Q will still be true by virtue of the remaining equivalence. However, if only one of R1 and R2 is true, then there is a model where Q is false, namely where R1 is true and C1 is false, and C2 is true but R2 is false, or vice versa. This shows that 6 is valid.

Consider 3. If Q is false then for T to hold either R1 is false, and C1 holds, or R2 is false and C2 holds. In either case the other conjunct is vacuously true. However, if we try to strengthen this to the case where both R1 and R2 are false, T will only hold if both C1 and C2 hold: this cannot be so on our account, showing that 7 is also true.

Consider finally 4. If R1 and R2 are both false, then if Q is true, for T to hold C1 and C2 must both be false, which is impossible. If either of them is true, Q must be false. If only one of R1 and R2 is false, then it is possible for T to hold and for Q to be true, namely where R1 and C1 are false and R2 and C2 are true, or vice versa. This shows that 8 also holds for us.

So our truth definition appears to support the kind of consequence relation that is appropriate for reasoning with ambiguous sentences. Notice that several other properties that are desirable will also fall out of our truth definition. For example, we want it to be the case that

9. $Q, \neg R1 \models R2$

which says that if there is a true two-ways ambiguous sentence and one of the interpretations is not true, the other one must be: perhaps the most basic kind of disambiguation strategy. It is in fact not completely trivial to arrive at such a conclusion without reducing ambiguity to disjunction: the logic of ambiguity in (van Eijck and Jaspars, 1996), for example, does not have this property (Jaspars, 1997). But a version of 9 follows directly from T, with the additional conclusion that C2 must also hold.

6.3 Reasoning with QLFs

Why would we want to be sure that we have a coherent semantics for QLFs and a sensible consequence relation? There are several reasons for doing so. Firstly, there are overwhelming arguments that some level like QLF is essential as part of a theory of utterance interpretation, both for linguistic and computational reasons. The practical arguments for this position are well known and have been implicit in computational practice since at least (Woods, 1968; Woods, 1978). It is simply not feasible to interleave the processes of quantifier scope or reference and ellipsis resolution etc. with the otherwise compositional process of meaning assembly. The space of possible interpretations becomes unmanageably large. However, postulating such a level of representation incurs an obligation to say what it means: logical and computational hygiene require us to supply a semantic account of it.

Secondly, since the processes of contextual interpretation involve a certain amount of inference to be successfully achieved, and since some of the ingredients in that inference are components of meaning of the sentence itself (such as the fact that a pronoun is masculine, or that a determiner like 'any' is in the scope of a negative) we need to be sure that our partly specified representations have enough of a semantics that we can carry out this reasoning in a logically respectable way.

In fact some types of linguistic processing presuppose that meanings are not fully resolved. Consider the following exchange:

25 A: She's here!
B: Who is?

The VP ellipsis in B's response has to be resolved with respect to an antecedent sentence that cannot be fully resolved: indeed, B's question would be pointless if it was. Observations like these compel the conclusion that partially resolved LFs need to have enough of a semantics to support this kind of inference, while still being subject to further linguistic processing.

Thirdly, there are many practical natural language processing applications that can be carried out without needing (or being able) to produce a fully contextualised interpretation of a sentence. Translation is an obvious example: while there will always be some cases for which full interpretation is required for a correct translation to be possible, in general translation on the basis of purely linguistic properties can often be perfectly adequate. A less obvious example is information extraction: since it is not possible at the current state of the art to

find complete grammatical analyses for every sentence, let alone full contextual interpretations, information extraction proceeds by reasoning from partial or underspecified representations that are in most logical respects the same kind of animal as the unresolved QLFs we have been talking about. Information extraction systems typically carry out such reasoning in a way that is, in Jerry Hobbs' phrase, unhindered by theory.

Developing a calculus for reasoning with QLFs is too large a task to be undertaken here. But the general outlines are reasonably clear, and we can adapt some of the UDRS (Reyle, 1995) work to our own framework. Reyle points out that many of the inferences involving underspecified representations that we would like to capture rely on the assumption that whatever context disambiguates the premise also disambiguates the conclusion, even if we do not know what that context or disambiguation is. His example is:

If the students get £10 then they buy books
The students get £10

They buy books

Our treatment of the interpretation of QLFs makes it a tautology that if one resolved form implies another, then the corresponding QLFs also do, given a fixed context.

The other common patterns of inference that we want to capture are those on which some (unambiguous) conclusion will follow from an unresolved form whichever resolution of the unresolved form is correct. Examples of these are things like:

Every student went to a lecture.
Mary is a student

Mary went to a lecture

Two hundred companies lost more than $2m last year.

Two hundred companies lost money last year.

A teacher who gave a low mark to every student was dismissed.

A teacher was dismissed.

The first argument is valid whichever scoping of the first premise is taken, and it is possible that most people would accept the argument as valid without even noticing the ambiguity. The second argument is valid whether the premise is construed in a collective or a distributive way. The third argument is valid whichever scoping of the relative clause is correct.

We can begin to capture such inferences by using proof rules for QLFs (partly modelled after those for UDRS in (Reyle, 1995)) such as these:

CONJ: (where R is resolved, and Q may contain some unresolved constructs)

R & Q

R

QUANT: (where Q is a downward monotone determiner, and P does not contain a negative)

$$\frac{P_{e=>t}(Q_{(e=>t)=>e}(R_{e=>t}))}{\text{exists(R,P)}}$$

CONJ and QUANT need considerable refinement in order to cover more than the simplest cases, but they will give the correct results for the latter two examples. For the first example something more is needed, perhaps along the lines suggested by Muskens (Muskens, 1998).

7 Conclusions and Further Work

We have presented what is probably the first fully bidirectional formalism for the interpretation and generation of quasi-logical form representations and illustrated its application with a fragment of English grammar that contains (admittedly simple) instances of some of the most important types of context-dependent construct. This fragment has been fully implemented and works as advertised.

We have tried to show that the interpretation of QLFs implicit in our treatment is a logically coherent one, supplying a kind of contextual truth definition for unresolved QLF constructs. We have also argued that this truth definition supports a notion of logical consequence that meets all the obvious desiderata for such a relation, and sketched how a calculus for reasoning directly with wholly or partially unresolved QLFs could be developed, again in a logically coherent way.

We conclude with a brief discussion of a series of issues that arise in thinking how to extend and apply the system described here.

Robustness: The work described here is an instance of what might be called 'classical' NLP: a (hopefully) neat bit of theory, a nice clean logical formulation, and a small-scale implementation. This kind of thing is currently desperately unfashionable on the grounds that such methods cannot scale up to real-world applications. (If you want to get ahead, get a corpus.) This is not the place to argue over whether this view is the correct one, but it is worth pointing out that the current theory is at least in one direction consistent with the kind of large scale statistical processing that is viewed as the appropriate alternative. It was stated earlier that the grammatical formalism used was not an essential component of the theory. Thus any alternative robust parsing system could be plugged in instead.

What *is* required is that QLFs be stated in a typed higher order logic. The QLFs we have been dealing with correspond to complete and correct grammatical analyses of sentences: in the real world, as we know, such things are not usually available. But in fact for the current approach, they are not needed: a partial or fragmentary analysis can be represented in QLF either by introducing a quantifier over a relation that is assumed to hold between the components (thus presupposing that there was a coherent message expressed by the partially analysed sentence) or by introducing Skolem-like predicate constants to achieve the same effect. (See (Pinkal, 1995) for a similar suggestion). The conditional equivalences

will apply to such representations directly, leaving the quantified relation or the Skolem constants in the resolved form. Of course, the resulting system will not be fully reversible, but it would be capable in principle of carrying out contextual disambiguation as part of a robust text processing system.

Disambiguation: We have been assuming that the 'correct' QLF has been chosen before applying our conditional equivalences. However, this is an unrealistic assumption in the fully general case, because it is quite conceivable that lexical disambiguation could require some contextual disambiguation first. Likewise, many PP attachment decisions have to be made on contextual grounds.

There are several stategies that might be pursued. One is to adopt Pinkal's 'radical underspecification' approach (Pinkal, 1995) and use underspecified representations for all types of ambiguity, even syntactic ambiguity. The more conservative approach is to try to integrate existing statistical disambiguation schemes for QLFs, either individually or in a 'packed' structure (Alshawi and Carter, 1994), with the resolution process as described here. Alternatively, I believe it is worth exploring the approach to disambiguation described in (Pulman, 2000), which would mesh nicely with the theory presented here.

Efficiency: Extending coverage of linguistic constructs, and trying to achieve robustness or integrate with disambiguation schemes each pose the further problem of the efficiency of the HOU based resolution process itself. While efficiency is acceptable for the short simple sentences illustrated earlier, the computational properties of HOU mean that processing times increase in a highly non-linear way when larger QLFs are encountered.

There are several avenues worth exploring to solve this problem. While the equivalences are stated in a direction-neutral manner, there is scope in an implementation for compiling them in different ways for the analysis and synthesis directions (recall that the equivalences decompose to a conjunction of higher order Horn clauses). Once you know which direction you are going in, most of the unifications actually reduce to matchings (since one side of the equation is fully instantiated), which may allow for various optimisations to the unification algorithm itself. (Prehofer, 1994) describes some tractable sub-cases of higher order unification.

Another strategy is to change the control regime by which the equivalences apply. The regime assumed here, and that implemented, is entirely non-deterministic. Equivalences apply to whole QLFs, in any order, whenever they can. This means that many equivalences are tried which are later filtered out because their associated conditions cannot be met. This is both expensive, and an unrealistic model of language processing.

The other strategy is to make the resolution process incremental, rather than operating on whole QLFs at a time. To some extent the linguistic facts force this option on us anyway, of course, because for many types of elliptical or anaphoric devices the appropriate context is an earlier part of the same sentence. It ought to be a relatively straighforward matter to devise a control strategy to resolve QLFs essentially a component at a time, perhaps guided by the original syntactic structure. This would bound the scope of HOU and keep it manageable,

as well as having an intuitely satisfactory 'dynamic' aspect to the resolution process. (The strategy used above is incremental and dynamic in that only one construct at a time is resolved, and each resolution changes the context, but this does not necessarily always correspond to a left-to-right traversal of the original sentence). There are some interesting interactions with the incremental interpretation scheme proposed in (Pulman, 1986) to be explored here.

Contextual resolution primitives: Currently the content of the conditions in conditional equivalences is rather unconstrained: any Prolog-definable predicate could be used. My hope is that in developing descriptions of a wider range of context-dependent phenomena, a set of conditions that recur (like 'parallel') can be isolated and defined in a way that covers their use in resolving different types of contextual dependency. Eventually one might hope that all the equivalences necessary would call on just a restricted range of such contextual predicates. These predicates would then in effect constitute the primitives of the linguistic theory of contextual resolution, factoring out all of the inferential processes that are not specifically linguistic. In moving towards such a theory, the requirement of reversibility is a hard one, but I believe it places a useful and productive methodological constraint on us, as well as yielding a significant practical payoff.

8 Acknowledgements

This paper is a descendant of (Pulman, 1994). Versions of it have been given at Bilkent University, Ankara, IMS Stuttgart, Cambridge, Edinburgh, ITRI Brighton, Sheffield, Oxford and at workshops in Bad Teinach and Saarbrücken. I thank the audiences on these occasions: I can remember (at least) Varol Akman, Nick Asher, Robin Cooper, Dick Crouch, Kees van Deemter, Jan van Eijck, Tim Fernando, Josef van Genabith, Jan Jaspars, Hans Kamp, Ron Kaplan, Stanley Peters, Manfred Pinkal, and Massimo Poesio making suggestions that changed the content of the paper in one way or another. The usual absolutions apply. Apologies to those who I have forgotten. Thanks are also due to three anonymous Computational Linguistics referees for insightful comments and criticisms, to Fernando Pereira for answering various questions, and to Michael Kohlhase for his detailed comments and helpful discussion of the pre-final draft.

References

Alshawi, Hiyan. 1990. Resolving quasi logical forms. *Computational Linguistics*, 16 number 3:133–144.

Alshawi, Hiyan. 1992. *The Core Language Engine*. M.I.T.Press.

Alshawi, Hiyan and David M. Carter. 1994. Training and scaling preference functions for disambiguation. *Computational Linguistics*, 20/4:635–648.

Alshawi, Hiyan and Richard Crouch. 1992. Monotonic semantic interpretation. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 33–39.

Ayuso, Damaris M. 1989. Discourse entities in Janus. In *Proceedings of the 27th ACL, Vancouver*, pages 243–250. Association for Computational Linguistics.

Barwise, John and John Perry. 1983. *Situations and Attitudes*. MIT Press.

Crouch, Richard and Josef van Genabith. 1997. On interpreting f-structures as UDRSs. In *Proceedings 35th ACL/ 8th EACL*.

Crouch, Richard S. and Stephen G. Pulman. 1994. Monotonic semantics. In R. Cooper, R. Crouch, J. van Eijck, C. Fox, J. van Genabith, J. Jaspars, H. Kamp, M. Pinkal, M. Poesio, S. G. Pulman, and E. Vestre, editors, *Describing the Approaches: Deliverable D8 of the FraCaS Project*. Cognitive Science, University of Edinburgh, pages 184–218. available from `ftp.cogsci.ed.ac.uk/pub/FRACAS`.

Dalrymple, Mary, J. Lamping, Pereira Fernando C.N., and Vijay Saraswat. 1996. Quantifiers, anaphora, and intensionality. *Journal of Logic, Language, and Information*, 6(3):219–273.

Dalrymple, Mary, Stuart M. Shieber, and Fernando C.N Pereira. 1991. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14(4):399–452.

Davidson, Donald. 1972. Semantics for natural languages. In D. Davidson and G. Harman, editors, *Semantics of Natural Language*. Reidel. Also in his *Inquiries into Truth and Interpretation*, Oxford, 1984.

Elworthy, David A. H. 1995. A theory of anaphoric information. *Linguistics and Philosophy*, 18(3):297–332.

Gardent, Claire and Michael Kohlhase. 1996a. Focus and higher–order unification. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen.

Gardent, Claire and Michael Kohlhase. 1996b. Higher-order coloured unification and natural language semantics. In *Proceedings of the 34th Annual Meeting, Association for Computational Linguistics*, pages 1–9. ACL.

Gardent, Claire and Michael Kohlhase. 1997. Computing parallelism in discourse. In *Proceedings of IJCAI '97*, pages 1016–1021, Tokyo.

Gardent, Claire, Michael Kohlhase, and Karsten Konrad. 1999. Higher-order colored unification: A linguistic application. *Téchniqe et Sciences Informatiques, special issue for JFPLC-UNIF'97*, 18(2).

Gardent, Claire, Michael Kohlhase, and Noor van Leusen. 1996. Corrections and Higher-Order Unification. In *Proceedings of KONVENS'96*, pages 268–279, Bielefeld, Germany. De Gruyter.

Gawron, Jean M. 1992. Focus and ellipsis in comparatives and superlatives: a case study. In C. Barker and D. Dowty, editors, *Proceedings of the Second Conference on Semantics and Linguistic Theory, Working Papers in Linguistics No 40, Ohio State University*, pages 79–98. Ohio University Press.

Gawron, Jean M. 1995. Comparatives, superlatives, and resolution. *Linguistics and Philosophy*, 18:333–380.

Hobbs, Jerry R. 1979. Coherence and coreference. *Cognitive Science*, 3(1):67–90.

Hobbs, Jerry R. and Andrew Kehler. 1997. A theory of parallelism and the case of vp-ellipsis. In *Proceedings of the 35th Annual Meeting, ACL, and 8th Conference of the European Chapter of the ACL*. Association for Computational Linguistics.

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

Huet, Gerard. 1975. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57.

Hurst, Matthew. 1994. Reversible resolution with an application to paraphrasing. In *Proceedings of the 15th International Conference on Computational Linguistics, Kyoto*, pages 551–555.

Jaspars, Jan. 1997. Minimal logics for reasoning with ambiguous expressions. Technical report available from `www.turing.wins.uva.nl/~jaspars`.

Kamp, Hans and Uwe Reyle. 1993. *From discourse to logic: Introduction to Model Theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*. Kluwer.

Kohlhase, Michael and Frank Pfenning. 1993. Unification in a lambda calculus with intersection types. In Dale Miller, editor, *Logic Programming: Proceedings of the 1993 International Symposium, Vancouver*, pages 488–505. MIT Press.

Lewin, Ian. 1990. A quantifier scoping algorithm without a free variable constraint. In *Proceedings of the 13th International Conference on Computational Linguistics, Vol. 3*, pages 190–194, Helsinki, Finland.

Miller, Dale and Gopalan Nadathur. 1986. Some uses of higher order unification in computational linguistics. In *Proceedings of the 24th Annual Meeting, Association for Computational Linguistics*, pages 247–255.

Montague, Richard. 1974a. The proper treatment of quantification in English. In Thomason R., editor, *Formal Philosophy*. Yale University Press, New York.

Montague, Richard. 1974b. Universal grammar. In Thomason R., editor, *Formal Philosophy*. Yale University Press, New York.

Muskens, Reinhard. 1998. Logic, reasoning and underspecification of linguistic structure. Papers from a workshop in Bad Teinach, Germany.

Pereira, Fernando C.N. 1990. Categorial semantics and scoping. *Computational Linguistics*, 16:1–9.

Pereira, Fernando C.N. 1991. Deductive interpretation. In E. Klein and F. Veltman, editors, *Natural Language and Speech*. Springer Verlag, pages 116–133.

Pinkal, Manfred. 1995. Radical underspecification. In *Proceedings of the 10th Amsterdam Colloquium on Formal Semantics*.

Poesio, Massimo. 1996. Semantic ambiguity and perceived ambiguity. In S. Peters and K. van Deemter, editors, *Semantic Ambiguity and Underspecification*. CSLI, pages 159–202.

Prehofer, Christian. 1994. Decidable higher-order unification problems. In *Automated Deduction CADE-12, 12th International Conference on Automated Deduction*. Springer.

Prüst, Hub. 1992. *On Discourse Structure, VP Anaphora, and Gapping*. Ph.D. thesis, University of Amsterdam.

Pulman, Stephen G. 1986. Grammars, parsers and memory limitations. *Language and Cognitive Processes*, 1(3):197–225.

Pulman, Stephen G. 1991. Comparatives and ellipsis. In *Proceedings of the 5th European Meeting of the Association for Computational Linguistics*, pages 1–6. Berlin: ACL.

Pulman, Stephen G. 1994. A computational theory of context dependence. In H. Bunt, R. Muskens, and G. Rentier, editors, *Proceedings: International Workshop on Computational Semantics*, pages 161–170. Institute for Language Technology, Tilburg University, The Netherlands.

Pulman, Stephen G. 1996. Unification encodings of grammatical notations. *Computational Linguistics*, 22/3:295–328.

Pulman, Stephen G. 1997a. Aspectual shift as type coercion. *Transactions of the Philological Society*, 95:2:279–317.

Pulman, Stephen G. 1997b. Higher order unification and the interpretation of focus. *Linguistics and Philosophy*, 20:73–115.

Pulman, Stephen G. 2000. Statistical and logical reasoning in disambiguation. *Philosophical Transactions of the Royal Society, Series A*.

Rayner, Manny. 1993. *Abductive Equivalential Translation and its application to Natural Language Database Inferencing*. Ph.D. thesis, Stockholm University. also available at `http://www.cam.sri.com/tr`.

Rayner, Manny and Hiyan Alshawi. 1992. Deriving database queries from logical forms by abductive definition expansion. In *Proceedings of the 3rd Interntational Conference on Applied Natural Language Processing, Trento, Italy 1992*, pages 1–8. Association for Computational Linguistics.

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

Reyle, Uwe. 1995. On reasoning with ambiguities. In *Proceedings of EACL 95*. Association for Computational Linguistics.

Reyle, Uwe. 1996. Co-indexing labelled drss to represent and reason with ambiguities. In S. Peters and K. van Deemter, editors, *Semantic Ambiguity and Underspecification*, pages 239–268. CSLI.

Thomas, James and Stephen G. Pulman. 1999. Bidirectional interpretation of tense and aspect. In H. Bunt et al., editor, *Proceedings of the Third International Workshop on Computational Semantics*, pages 247–263.

van Deemter, Kees. 1996. Towards a logic of ambiguous expressions. In S. Peters and K. van Deemter, editors, *Semantic Ambiguity and Underspecification*. CSLI, pages 203–238.

van Eijck, Jan and Jan Jaspars. 1996. Ambiguity and reasoning. CWI Technical Report CS-R9616.

Webber, Bonnie L. 1983. So what can we talk about now? In M. Brady and R. Berwick, editors, *Computational Models of Discourse*. MIT Press, pages 331–371.

Woods, William. 1968. Procedural semantics for a question-answering machine. In *Proceedings of the Fall Joint Computer Conference, New York*, pages 457–471. New York.

Woods, William. 1978. Semantics and quantification in natural language question answering. In Yovits, editor, *Advances in Computers*. Academic Press, New York, pages 2–64.