Second order inference in natural language semantics

Stephen Pulman
Department of Computer Science, Oxford University

ABSTRACT

In this paper I look at a number of apparently trivial valid inferences (as well as some invalid and missing inferences) associated with the possessive construction and with different types of adjectival modification of nouns. In the case of possessives, all analyses I know of, whether implemented or not, systematically sanction invalid inferences. In the case of adjectives, there are some model-theoretic linguistic analyses that are adequate at a theoretical level, but no satisfactory practical computational implementations that I am aware of which capture the correct inference patterns.

A common thread between the possessive and the adjectival constructions is that to derive the correct inferences we need second order quantification. This is an uncontroversial move within model-theoretic formal semantics but a problem for computational semantics, since we have no fully automated theorem provers for anything other than first order logic (and only for subsets of first order logic do we have provers that are both fully decidable and efficient). I explore what is needed to provide a proof-theoretic account of the relevant inference patterns, and suggest some analyses requiring second order axioms. In order to make this a practical computational possibility I go on to propose two techniques for approximating such inferences in a first order setting. The suggested analyses have been fully implemented, and in an appendix I provide a small FraCaS-like corpus of relevant examples, all of which are handled correctly by the implementation.

Keywords: first order, second order, inference, adjectives, possessives

1 INTRODUCTION

The aim of this paper is to be able to capture some apparently trivial natural language inferences (and lack of inferences) involving adjective modification and possessive determiners, which like many other constructions turn out to have the property that second order quantification is required to capture these inferences. I will assume a simple and standard setting in which to address this problem, assuming that we have a syntax-driven compositional semantics producing logical forms for a (disambiguated) parsed sentence in a familiar way. These logical forms will ideally be sent to an automated theorem prover of some type (resolution, tableau...) which can mechanically check the validity of the inferences. A common version of this setting is to have the translations of declarative sentences or statements added as 'axioms' or 'premises', and then to have questions corresponding to the inferences we are interested in treated as 'theorems' to be proved, as for example in versions of the FraCaS inference suite (Cooper *et al.* (1996), MacCartney and Manning (2008)). The questions can be yes/no type questions where we will expect the answer 'yes' if there is a proof and either 'no' or 'don't know' otherwise (there's more to be said here: failure to find a proof does not always mean a negative answer), or Wh-questions where if there is a proof we will ideally return unifying substitutions corresponding to the values of the 'wh' constituent in the question.

Here is a very simple example:

All bankers are rich.	axiom: $\forall x.\text{banker}(x) \rightarrow \text{rich}(x)$
Jones is a banker.	axiom: $\text{banker}(\text{jones})$
Is Jones rich?	prove: $\text{rich}(\text{jones})$
Who is rich?	prove: $\exists x.\text{rich}(x)$

The first order logical (FOL) forms in the right hand column can be submitted to a first order theorem prover such as Prover9 (McCune (2005–2010)) and the answers retrieved (after some housekeeping) should be 'Yes' and 'Jones' respectively.

My modest aim in this paper is to be able to do something similar with inferences such as those described in the following sections, involving different types of adjectives, possessive determiners, and their combinations.

2 ADJECTIVES

For completeness, we will go through the standard examples of the inferential phenomena we are interested in even though, at least for adjectives, they are comparatively well-known. We begin with the simplest class of adjectives, usually called 'intersective', which sanction inferences like these:

(1) a. Jones is a red-haired farmer.
b. $\models$ Jones is red-haired.
c. $\models$ Jones is a farmer.

If we now add some extra information about farmers we get the following inference pattern:

(2) a. All farmers are gamblers.
b. $\models$ Jones is a red-haired gambler.

With what are commonly called 'gradable' or 'subsective' adjectives, we get a different pattern of inferences:

(3) a. Minnie is a large mouse.
b. $\models$ Minnie is a mouse.
c. $\not\models$ Minnie is large. *(can be valid with some contextual assumptions)*

(4) a. All mice are animals.
b. $\models$ Minnie is an animal.
c. $\not\models$ Minnie is a large animal.

(5) a. All mice are small animals.
b. $\models$ Minnie is a small animal.

Gradable adjectives have some implicit scale of comparison associated with them, and thus something can have contradictory properties if these are associated with different comparison scales. You can be a tall person but a short basketball player, for example. For many such adjectives it sounds odd to have the property ascribed unless the comparison scale is obvious from the context. In a few cases there can be a default comparison class, e.g. 'Mary is good' can be meaningful without a specific hidden parameter since for most people a

default parameter for 'good' will be 'behaviour' or 'character'. Some gradable adjectives like 'clever' or 'generous' have a further dimension, in that someone might be generous not only by comparison with other members of a class, but generous with respect to some properties (e.g. money) and not others (e.g. time). Recovering these relevant contextual parameters is a long way beyond the state of the art computationally, and so here we only use examples where the parameter is supplied linguistically, for example 'John is a tall man', and 'This is a red apple', rather than 'John is tall' or 'This apple is red'. Clearly what is tall for a man is not what is tall for a tree, and red for an apple is very different from what is red for a face, and until we know what this parameter is, few inferences are sanctioned.

A third class of adjectives are sometimes called 'privative', and whereas the first two classes have the property that from 'X is Adj Noun' we can always infer 'X is Noun', privatives do not behave in this way:

(6) a. Tony Blair is the former Prime Minister.
b. $\not\models$ Tony Blair is the Prime Minister.

(7) a. Smith showed an apparent proof of the theorem.
b. $\not\models$ Smith showed a proof of the theorem.

(8) a. He owns a fake diamond.
b. $\not\models$ He owns a diamond.

All of these adjectives have the property that from 'X is Adj Noun' the inference 'X is Noun' does not hold, and some have argued that for some cases, like 'fake', the inference to 'not-Noun' holds:

(9) a. This is a fake diamond.
b. $\models$ This is not a diamond.

Intuitions vary about this: Partee (2007) thinks that 'former P' entails 'not P now', whereas for me a sentence like 'In 2014, Obama was both the former and the current US President' is not contradictory.

Inferences from the complement of a privative adjective seem quite varied: 'This is a fake Picasso painting' does not entail 'This is a fake painting', whereas 'Bush is a former US president' does entail 'Bush is a former president'. However, 'Jane is a former fussy eater',

does not entail 'Jane is a former eater'.[1] Clearly there is more to be learned about such examples: for an interesting extended discussion and analysis of different types of privative adjectives, see Del Pinal (2015).

For some adjectives of this type, there are also some further interesting properties when combined with possessive determiners, such as the ambiguity of 'Mary's former mansion', which can be interpreted as referring either to the mansion that Mary used to own, or the building that Mary still owns which used to be a mansion. See Partee (2007) for discussion.

It is reasonably easy to specify truth conditions for **intersective** adjectives as follows, where **D**(x) = 'denotation of x':

> 'Jones is a red-haired farmer' is true iff
> **D**(jones) ∈ **D**(red-haired) ∩ **D**(farmer).

However, extending this definition to the other two classes requires appeal to notions which are not all intuitively clear and not very easy to pin down with mathematical precision. For **subsective** adjectives, perhaps:

> 'Minnie is a large mouse' is true iff **D**(minnie) ∈ {X | X a mouse larger than the relevant standard for mice}

Making the notion of "relevant standard" precise might involve assuming an ordering over mice by size (presumably adjusting for age) and fixing an interval representing the expected norm. As many people have commented (e.g. Kamp (1975)), this seems a little odd, in that it implicitly uses the comparative form of the adjective to define the semantics of the non-comparative form, whereas pre-theoretically one might have expected things to be the other way round.

In the case of **privative** adjectives, truth conditions seem to vary according to the specific adjective. For example, 'X is a former Y' is true iff **D**(X) ∈ **D**(Y at earlier time), and 'X is an alleged Y' is true (according to Morzycki (2014)) iff X is a Y in every possible world compatible with the allegation (although wouldn't 'X is an alleged Y iff someone has alleged that X is a Y' be simpler?).

[1] Thanks to a referee for this example.

It is clear that there is a large contextual component in the interpretation of all of these adjectives, and it is perfectly reasonable to pursue a style of analysis in which the logical form associated with them is rather minimal, most of the hard work being done by setting of various contextual parameters, with the analysis perhaps also involving probabilities or utilities (Rett (2014); Lassiter and Goodman (2017)). But whatever the undoubted merits of these approaches to defining interpretation conditions for adjectives or other context dependent constructs, the exercise is not very practically relevant for computational purposes, for which we need an explicit logical form that will support the relevant inferences proof theoretically, or which will lend itself to computationally tractable model building and checking techniques. In this respect, computational semantics for natural language is a rather different pursuit than purely linguistic semantics.

When constructing logical forms, if we are to be as compositional as possible, then any differences in the logical form of these three types of adjective under discussion must come either from some syntactic differences between the sentences in which they occur, or from their lexical properties. Since there seems to be no compelling evidence of a syntactic difference between these types of adjective (there are distributional differences to do with attributive and predicative uses of adjectives but this seems to cross-cut the present set of distinctions) I propose to build semantic differences into their lexical logical forms directly.

I will illustrate this with a small but precise fragment: a context-free grammar with associated semantic rules which build the meaning of a mother constituent by combining the meanings of daughter constituents. The meanings are expressed in a simply typed higher order logic of a familiar kind. For example, the first rule says that a Sentence (S) consists of a Noun Phrase (NP) followed by a Verb Phrase (VP) and that the meaning of the sentence is obtained by substituting the meanings of the NP and VP in the typed[2] higher order logic schema following the rule, which applies the NP meaning to the VP meaning.

[2] A type like `(et)t` is equivalent to $(e \rightarrow t) \rightarrow t$ or `<<e,t>,t>` in other notations.

S	→	NP VP	: $NP_{(et)t}(VP_{et})$
NP	→	Det N'	: $Det_{(et)(et)t}(N'_{et})$
NP	→	Name	: $\lambda P_{et}.P(Name_{e})$
N'	→	N	: N_{et}
N'	→	Adj N'	: $Adj_{(et)(et)}(N'_{et})$
Adj	→	wooden, etc.	: $\lambda P_{et}x_{e}.wooden_{et}(x) \wedge P(x)$
Adj	→	small, etc.	: $\lambda P_{et}x_{e}.small_{e(et)t}(x,P)$
Adj	→	apparent, etc.	: $\lambda P_{et}x_{e}.apparent_{e(et)t}(x,P)$
Det	→	some, etc.	: $\lambda P_{et}Q_{et}.\exists x_{e}.P(x) \wedge Q(x)$ etc.
VP	→	Vi	: Vi_{et}
VP	→	Vt NP	: $Vt_{((et)t)et}(NP_{(et)t})$
Vi	→	snores, won, etc.	: $snore_{et}$
Vt	→	hits, is, likes, etc.	: $\lambda O_{(et)t}x_{e}.O(\lambda y.hit_{eet}(x,y))$

This grammar will deliver logical forms for our representative cases as follows, with some simplifications concerning the copula:[3]

Jones is a red-haired rugby player.
red-haired(jones) ∧ rugby-player(jones)
Minnie is a large mouse.
large(minnie, mouse)
Tony Blair is a former Prime Minister.
former(tonyblair,prime-minister)

For the intersective cases, we get the inferences we want immediately, since we have built conjunction into the lexical entry. For the subsective cases, we supply the non-logical constant encoding the adjective with an extra second order argument, which picks up the denotation of the noun as supplying the relevant comparison class. We can read small(x,P) as 'small by the standards relevant for P'. Of course, there is one crucial inference not proof-theoretically sanctioned by this logical form: to get the inference from adj(x,P) that P(x) we add (in an implementation, via an axiom schema) an axiom for each such adjective:

(10) $\forall xP.adj(x,P) \rightarrow P(x)$

[3] Following Montague, 'be' is translated as a transitive verb meaning '=', and the resulting logical forms can be simplified using the equivalence: $\exists x.P(x) \wedge x=a \equiv P(a)$. We notate this below as '$\Rightarrow_{=}$'.

We could instead have had a lexical entry for these adjectives that builds the inference in directly:

(11) $\lambda Px.small_{e(et)t}(x,P) \wedge P(x)$

but this is not in my view particularly compositional.[4] While compositionality is difficult to define (see the survey and discussion in Szabó (2017)) and may be no more than a methodological rule of hygiene, some simple principles would surely include a requirement that a single content word in a sentence should correspond to no more than one component of a logical form (whereas function words in most frameworks have to be allowed to introduce an amount of logical 'glue').

In order to cope with the privative cases we simply refrain from generating these axioms and so we (correctly) cannot infer from $apparent(x,P)$ that $P(x)$. For those privative adjectives, if there are any, that sanction the negation of the property we can add an axiom:

(12) $\forall xP.adj(x,P) \rightarrow \neg P(x)$

This is all very tidy and makes it easy to define truth conditions for these logical forms with rather less contextual clutter than would be needed for simpler forms that did not include these parameters. But these logical forms do not solve our computational inferential problem because they involve second order arguments to predicates. The state of the art in automated inference is that we have reasonably efficient general purpose theorem provers for first order logic (with equality) except that they are bounded by the inescapable semi-decidability of FOL, and the unpredictable computational complexity of general inferences. By restricting the expressivity of FOL to tractable subsets (Baader *et al.* (2003)) we can guarantee good performance, but only for a small number of the cases we would like to handle.

Regrettably, it is both in theory and in practice impossible to reason directly, as we would like to do, with higher order logics: even the notion of higher order unification needed as a component is undecidable (Huet (1975)). There do exist some higher order logic proof assistants like HOL (`https://hol-theorem-prover.org/`), Isabelle (`https://www.cl.cam.ac.uk/research/hvg/Isabelle/`), and Coq

[4] The same objection, and a version of the same solution, are relevant to the treatment of intersective adjectives just given.

(`https://coq.inria.fr/`). However, these are not fully automatic theorem provers, but interactive systems requiring human guidance and input at every step, and are usually used for checking already generated proofs. It is, however, possible to write special purpose proof 'tactics' to guide a proof assistant like Coq to carry out some specific higher order logic inferences derived from natural language expressions semi-automatically, and in a series of papers from Chatzikyriakidis and Luo (2014) onwards, Chatzikyriakidis and Luo have carried out such experiments on a variety of constructions. Similar efforts are described in Mineshima *et al.* (2015) and related papers. However, while this is an interesting experiment from the point of view of validating particular higher order analyses of linguistic phenomena, it is important to recognise that it is a very different exercise from our current aims: the approach is not a general purpose technique of the type we would like, but something which will only work on prespecified patterns and derivations. The results could not, for example, form a component of an automatic natural language processing system performing these inferences as part of an application task like question answering or task-oriented dialogue.

The limitations of automated higher order logic inference constitute a real barrier to computational semantics of natural language, because like the analysis here, many natural language constructs are intrinsically higher order. Some obvious ones are generalised quantifiers and intensifying modifiers, where outline logical forms are shown below. 'Most' will be a function from a noun meaning to a function from verb meanings to truth values. 'Very' will be a function from adjective meanings to adjective meanings.

(13) a. Most dogs bark. = (most dog) bark

b. John is very tall. = (very (tall)) john

One approach to this problem, since the specialised HOL or Coq proof tactics just mentioned are not a general solution, is to try to translate or compile the higher order forms to something that a FOL prover can deal with. There are a number of strategies that have been tried: reification or 'ontological promiscuity' attempts to 'compile out' the higher order aspects by adding different types of abstract individuals to first order models. Some common examples of this strategy in-

clude event analyses of verb modification (Davidson (1967)), although in this case, arguably, there is also some linguistic motivation:

(14) a. John ran in the park. = (in the park)(run)(john)
b. ⇒ ∃e.run(e,john) ∧ in(e,the park)

or the so-called 'standard translation' of modal logic:

(15) □p ⇒ ∀w.R(thisWorld,w) → p(w)

which translates 'necessarily p' into 'in all worlds w in the appropriate relation R to this world, p is true in w'.

Hobbs (1985) has been a notable advocate of this approach, and it has been applied to the semantics of adjectives in Amoia and Gardent (2007). But it's not obvious how such a strategy could help us here, in the general case at least, although it has been used successfully in specific limited domains where we can precompute values for the various adjective parameters. Let's assume we try to eliminate the second order arguments in our subsective Adj meanings by adding entities representing standards of Adj-ness for those adjectives. We will then translate 'John is a tall man' as something like:

(16) ∃s.tall(john,s) ∧ man(john) ∧ tallness-for-men(s)

'John is tall to s, where s is that degree of tallness for men which qualifies as being tall'. (Note that in forms like 'John is tall' we will have to fill in the relevant noun parameter from context or non-linguistic knowledge, but this is the case for all approaches). So far, so good: it is easy to see how to make implementational sense out of this, given a sufficiently well structured domain. However, when we look at what else we need to do to make this analysis work things get more complicated: for example, we need to ensure that an adjective interacts properly with related (usually antonymous) adjectives:

(17) John is a tall man. ⊨ John is not a short man.

∀xyz.tall(x,y) ∧ tallness-for-men(y) →
¬(short(x,z) ∧ shortness-for-men(z))

This is doable, if a little clumsy, and as we extend similar axioms we need to be careful to ensure that 'John is not short' does not wrongly entail 'John is tall'. A more serious problem is that there are

a potentially infinite number of such 'adjness-for-X' entities and their predicates, and therefore a potentially infinite number of such relatedness axioms. This happens because it is possible to combine adjective modification in principle to an arbitrary depth, essentially creating 'standards of comparison' on the fly:

(18) a. This is an old American building.
b. This is an older mid-period Anglo-Saxon religious site.

The interpretation we are interested in here is that on which each adjective modifies everything that follows it, rather than the usually possible 'conjunctive' reading on which each adjective just modifies the head noun. So we need a standard of comparison for age relevant for 'American building', which will be different from that for 'English building', as well as a standard for mid-period Anglo-Saxon religious sites. 'Mid-period' is itself subsective, the standard for that type of Anglo-Saxon religious sites will be different from that for Anglo-Saxon religious sites of all periods, and so on. The recursive nature of adjectival modification means that there is no limit in principle to the number of such standards and so we cannot just define them all in advance, nor can we list in advance all the required axioms connecting antonyms.

However, our second order analysis of these adjectives generalises quite cleanly to this case, without requiring separate axioms for each further combination:

(19) a. This is an old American building. =
b. old(this,λx.American(x) $\wedge$ building(x))

(20) a. This is an old mid-period Anglo-Saxon religious site. =
b. old(this,λx.mid-period(x, λy.Anglo-Saxon(y) $\wedge$ religious(y) $\wedge$ site(y)))

and the interaction with related predicates only needs one (second order) axiom (again generated from a schema, we assume), which quantifies over every possible standard of comparison:

(21) $\forall$xP. old(x,P) $\rightarrow$ $\neg$(young(x,P))

(22) $\forall$xP. tall(x,P) $\rightarrow$ $\neg$(short(x,P))

While this is satisfactory from the point of view of linguistic analysis, we are unfortunately still no nearer to a solution to the problem of how to automate inferences involving these logical forms: they are still second order.

3 POSSESSIVES

We turn now to possessive determiners, an apparently simple construction, but one which on closer inspection has several interesting properties. There are a number of relevant well-known properties of possessives for us to bear in mind when trying to uncover their inferential properties, as well as some less well-known properties. It is a striking fact, discussed further below, that that all of the well-known analyses of possessives sanction invalid inferences involving them.

Firstly, an obvious point to make is that the relation between possessor and possessed can vary and is not just restricted to a small set of semantic notions like 'ownership', 'part of', and the like; rather, it can depend on almost any feature of the linguistic or non-linguistic context:

The table's leg...	Monday's lecture...
America's invasion of Iraq...	John's measles...
John's dog...	John's brother...
John's portrait...	etc.

For example, 'John's dog' can mean the dog that John owns, the dog that John has just sold, the dog that John has just bet on to win in a race, etc. This wide contextual dependence, as with adjectives, makes it perfectly reasonable to adopt an analysis on which logical forms are relatively simple, and all the heavy lifting is done by setting of various contextual parameters. But as we argued when discussing adjectives, this is not a stance that is open to anyone wanting an implementable account of the inferences associated with such constructions.

Secondly, what we are calling the possessive comes in various syntactic forms:

(23) a. John's picture/team/sister
b. a picture/team/sister of John's
c. a picture/*team/sister of John
d. That picture/team/sister is John's

As the 'team' examples show, there are some acceptability variations associated with the difference between what are often called 'relational' and 'sortal' nouns. Relational nouns implicitly correspond to two-argument predicates whereas sortal nouns are more naturally modelled as one-argument predicates. As we will see later, this does not necessarily correspond to a syntactic distinction.

Third, relational and sortal nouns also seem to sanction inference patterns of differing acceptability (de Bruin and Scha (1988)), where we interpret 'has' in the following as denoting the same relation as the possessive:

(24) a. John's cars are wrecks. $\models$
b. Some wrecks of John's are cars; Some wrecks/cars are John's.
c. John has some wrecks; John has some cars.

(25) a. John's brothers are musicians. $\models$
b. ?Some musicians of John's are brothers.
c. ?Some musicians/brothers are John's.
d. ?John has some musicians; John has some brothers.

Despite these differences in acceptability, I would prefer not to distinguish relational vs. sortal nouns syntactically. This is because of the fourth observation: that all relational nouns can be interpreted as sortal in the right context, as many people have pointed out:

(26) The headmaster has difficulty dealing with his parents.

(Parents' evening context: headmaster is talking to parents of the children in his school.)

(27) John's famous wife is Victoria Beckham.

(John is one of several journalists tasked with writing a piece about famous men with equally famous wives.)

In the following, we will not attempt to capture all of the interesting properties of possessives in our analysis. Instead, we will concentrate on a quite modest ambition: we would like an implemented analysis of the possessive which allows us to avoid the invalid inferences of existing analyses, to be described later, and to capture valid inferences like the following:

(28) Smith is Jones's plumber. ⊨ Smith is a plumber.

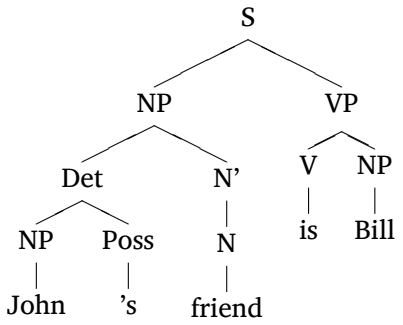
(29) a. John's old wooden toy broke. ⊨
b. John's wooden toy broke.
c. A wooden toy broke.
d. A toy broke.

(30) a. The student's essay's title intrigued Jones. ⊨
b. An essay's title intrigued Jones.
c. A title intrigued Jones.

(31) a. All John's brothers are rich.
b. Bill is John's brother.
c. ⊨ Bill is rich.

3.1 *An initial simple analysis*

A simple analysis (variants of which can be found in many places, for example Bos *et al.* (2004), or more recently, Steedman (2012)) takes the possessive morpheme *'s* (or just *'* for plurals) to be a function from NP meanings to Det meanings introducing an abstract two-place 'of' or 'poss' relation, usually assumed to be subject to further contextual resolution. In the illustrative framework we are using this would be implemented as follows:



(32) John's friend is Bill. = $\exists x.\text{friend}(x) \wedge \text{of}(x, \text{John}) \wedge x=\text{Bill}$

We need to add some rules to our earlier fragment to produce such an analysis:

Det → NP poss : $poss_{((et)t)(et)(et)t}(NP_{(et)t})$
poss → ' or 's : $\lambda O_{(et)t} P_{et} Q_{et}.O(\lambda y.\exists x.P(x) \wedge of_{eet}(x,y)) \wedge Q(y)$

In principle, we ought to be able to leave the 'of' predicate unresolved – not the least because this kind of contextually sensitive resolution is a completely unsolved computational inference problem – and still get most of the inferences we would like to get. But it turns out that this will lead us astray. If we leave 'of' unresolved, we will sanction some incorrect inferences:

A: John's brother is Bill. $= \exists x.brother(x) \wedge of(x, John) \wedge x=Bill$
$\Rightarrow_{=} brother(Bill) \wedge of(Bill,John)$

B: Bill is a doctor. $= \exists x.doctor(x) \wedge x=Bill$
$\Rightarrow_{=} doctor(Bill)$

C: Bill is John's doctor. $= \exists x.doctor(x) \wedge of(x, John) \wedge x=Bill$
$\Rightarrow_{=} doctor(Bill) \wedge of(Bill,John)$

Now C is provable from the conjunction of A and B, incorrectly; whereas C is not a valid inference from A and B.

Perhaps it was a mistake to leave 'of' unresolved? 'Of' can be contextually interpreted as 'has', 'owns', or as an arbitrarily complex context-dependent relation like 'bet-on-by', or as the relation associated with a relational noun, if present:

(33) a. John's dog won. =
b. $\exists x.dog(x) \wedge owned\text{-}by(x, John) \wedge won(x)$
c. $\exists x.dog(x) \wedge bet\text{-}on\text{-}by(x, John) \wedge won(x)$
d. etc.

(34) a. John's brother arrived. =
b. $\exists x.brother(x) \wedge brother\text{-}of(x,John) \wedge arrived(x)$

If we now interpret 'of' in A above as the two-place relation 'brother(Bill,John)', and as something else in C (for example, 'treated-by'), then the incorrect inference will not be made.

Unfortunately, contextual interpretation doesn't always solve this problem. Although our invalid inference will not go through when relational nouns are involved (at least if we use them as the source for the contextually dependent resolution option) we cannot always

guarantee validity for examples involving sortal nouns. Consider the following example:

A: Smith is Bill's plumber. = (interpret 'of' as 'works-for')
$\Rightarrow_{=}$ plumber(Smith) $\wedge$ works-for(Smith,Bill)

B: Smith is also a decorator.
$\Rightarrow_{=}$ decorator(Smith)

C: Smith is Bill's decorator.
$\Rightarrow_{=}$ decorator(Smith) $\wedge$ works-for(Smith, Bill)

It's surely impossible to argue that 'of', interpreted as a contextually dependent 'works-for', or 'employed-by' relation, should be instantiated differently in A and C, and under these interpretations the unwanted inference will still go through. The analogous bad inference will also go through even where we do have a relational noun but where it is interpreted sortally. If we interpret the possessive as something like 'taught by' in:

A: The noisy class were Mr Smith's children.

B: The noisy class are also the Latin class.

C: The noisy class are Mr Smith's Latin class.

then C should not follow from A and B, but it will do so given the logical forms assigned by this analysis, even after resolution.

3.2 *Two further, more sophisticated, analyses*

In Partee and Borschev's analysis (Partee and Borschev (2003)), the possessive morpheme introduces a lot more structure:

(35) John's = λN.λP. $\exists$x.[Sort(N)](x) $\wedge$ R_{gen}(x,John) $\wedge$ P(x)

In their analysis, relational and sortal nouns are assigned to different types: for example, brother_{eet} and team_{et}. In order to keep the types straight in composition they define a 'typeshifting' operator, 'Sort', defined thus:

(36) a. A: $\text{Sort}(N_{eet}) = \lambda x.\exists y.N(x,y)$
b. B: $\text{Sort}(N_{et}) = N$

Applying clause A of the definition to a relational noun like brother_{eet} = $\lambda x.\lambda y.\text{brother}_2(x,y)$ produces a one-argument version with the same type as the corresponding sortal noun: $\text{brother}_1 = \lambda x.\exists y.\text{brother}_2(x,y)$.

The relation "R_{gen}" is their version of our 'of' relation, to be contextually interpreted, but with a default preference for the relational version of a noun N_2 if N_1 is present. This approach gives analyses like the following:

(37) a. John's team won. =

b. $[[\lambda N.\lambda P.\exists x.(Sort(N))(x) \wedge R_{gen}(x,John) \wedge P(x)](\lambda y.team(y))](won)$

c. $\Rightarrow_{\beta} \exists x.team(x) \wedge R_{gen}(x,John) \wedge won(x)$

The relation 'R_{gen}' can then be contextually interpreted as something like 'played-in-by' or 'supported-by', as appropriate. For the relational case:

(38) a. John's brother arrived. =

b. $[[\lambda N.\lambda P.\exists x.(Sort(N))(x) \wedge R_{gen}(x,John) \wedge P(x)](\lambda y.\lambda z.brother_2(y,z))](arrived)$

c. $\Rightarrow_{\beta} \exists x.brother_1(x) \wedge R_{gen}(x,John) \wedge arrived(x)$

then R_{gen} can be instantiated to the original relational version of 'brother':

(39) $\exists x.brother_1(x) \wedge brother_2(x,John) \wedge arrived(x)$

I do not find this a particularly satisfying or elegant analysis. Note that for sortal nouns, this is just a variant of our first simple analysis, and so it will also sanction the same set of invalid inferences. In the case of relational nouns, the treatment is surely very clumsy. Initially, the contextually appropriate two-place relation is accounted for in the analysis, but transformed to a 1-place relation to keep the types straight. Then the original two-place relation has to be recovered again by inference. Furthermore, the strategy of giving relational nouns a different type from sortal nouns means that everything that can combine with N (Det, Adj, etc.) will now have to be polymorphic, i.e. set up to expect two different types: `eet` and `et`, or alternatively have the 'Sort' operator wrapped around it to coerce two-argument predicates to one-argument predicates. This seems a high price to pay, both linguistically and computationally.

An influential alternative analysis by Peters and Westerståhl (2006) (see also Peters and Westerståhl (2013)) makes several perceptive contributions to our understanding of possessives. In their

analysis, possessives involve two quantifiers, one associated with the NP in the possessive Det phrase, and the other either explicit, as in:

(40) Several of each farmer's sheep are infected.

or contextually inferred:

(41) a. John's fingers are clean. (all of them)
b. John's fingers are dirty. (just some)

A second concern in their analysis is to capture the phenomenon of 'narrowing', as in:

(42) a. Most people's grandchildren like them.
b. Many planets' moons are visible.

In these examples, 'most/many' are quantifying over 'people with grandchildren' or 'planets with moons' rather than just 'people' or 'moons'. A related phenomenon is discussed by Bos (2009) noting that possessives involving superlatives:

(43) a. London's most expensive restaurant...
b. Milan's best player...

require a comparison set that involves the possessor as well as the possessed.

Peters and Westerståhl give truth conditions for two variants of the possessive construction (their account is couched in model theoretic terms), with or without an explicit quantifier (Q_2) in a predeterminer position:

(44) a. Q_1 C's As are B
b. Q_2 of Q_1 C's As are B

Peters and Westerståhl define a 'Poss' higher order operator (distinct from the "poss" morpheme) which has four arguments: (i) an explicit (sometimes implicit) quantifier in the possessive determiner phrase, (ii) the explicit or contextually inferred predeterminer quantifier, (iii) the possessed nominal relation, and (iv) a two-place relation 'R' corresponding to our 'of' and also a placeholder for a contextually inferred relation. They further define, for a two-place relation R and a set A:

(45) a. R_a = {b : R(a,b)} or in our logical form notation λb.R(a,b)
b. $dom_A(R)$ = {a: A ∩ $R_a \neq \emptyset$} or λa.$\exists$b.A(b) ∧ R(a,b)

The expression in (a) denotes the set of things possessed by *a* and in (b) the set of objects that possess something in A. Now the truth conditions for an expression involving 'Poss' are defined as:

(46) $Poss(Q_1,C,Q_2,R)(A,B) = Q_1(C \cap dom_A(R), \{a: Q_2(A \cap R_a, B)\})$

Q_2, as above, is the inferred or predeterminer quantifier. Read this expression as:

(47) Q_1 C x that 'possess' an A are such that Q_2 A that x 'possesses' are B

It is assumed that even where Q_1 is not explicit, as in 'John's...' there is an implicit non-vacuous universal quantifier involved. The syntactic structure assumed for the case where there is an explicit predeterminer quantifier is illustrated in Figure 1. The case where the second quantifier is implicit is illustrated in Figure 2, and a concrete example of this phenomenon is offered in Figure 3.

In Figure 3, interpreting 'R' as 'own', we arrive at the interpretation:

(48) a. Poss(most,students,every,own)(cars,rusty)
b. = most(students ∩ dom_{car}(own), {a: every(car ∩ own_a, rusty)})

Translated to our logical form notation:

(49) most(λx.student(x) ∧ $\exists$b.car(b) ∧ own(x b),
λa.every(λy.car(y) ∧ own(a,y),rusty))

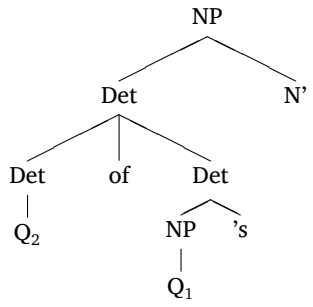


Figure 1:
Case with explicit predeterminer quantifier

Figure 2:
Case with implicit predeterminer quantifier

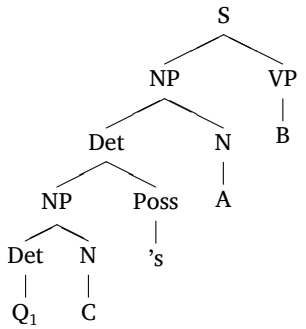
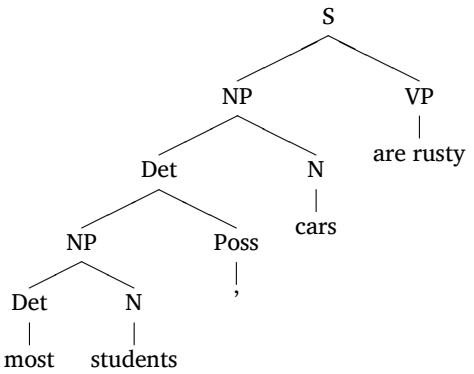


Figure 3:
An example of the phenomenon in Figure 2, where 'R' is interpreted as 'own'



While this is one of the most sophisticated analyses of the possessive in the literature, there are still a number of problems with it. For example, it is not clear that the 'implicit predeterminer quantifier' is specific to possessives or is an instance of the implicit quantification that is needed anyway for bare plural nouns (Lauri Carlson, p.c.). Furthermore, as the authors point out, this account is not fully compositional, since 'Poss' needs access to the components of its sister NP separately in order to build 'narrowing' into the truth conditions. In Peters and Westerståhl (2013) this is described as "second level" compositionality, accessing immediate constituents of immediate constituents, as opposed to "first level" compositionality. (We will ignore narrowing in what follows, for simplicity.)

For our purposes the most salient shortcoming of this analysis is that, however R is contextually interpreted, provided it is interpreted consistently, the unwanted inference in our 'plumber' and 'decorator' case will still go through on Peters and Westerståhl's analysis. This

is, as we shall see shortly, because any binary relation R is incapable of capturing the dependencies involved in blocking the invalid inferences.

Johan Bos (Bos (2009)) is aware of the problem we have signalled and suggests an analysis (that he attributes to Yuliya Lierler and Vladimir Lifschitz) in which we translate sentences like 'Vincent is Mia's husband' as:

(50) person(Vincent) ∧ ∃y.role(Vincent,y) ∧ husband(y) ∧ of(y,Mia)

paraphrased as 'Vincent is a person who is playing the role of Mia's husband'.

His suggestion is not sketched in full detail, and it may be possible to extend it to overcome these objections, but as it stands this analysis leaves much to be desired, in my view. To begin with, it is highly non-compositional: there are no words in the sentence corresponding to the logical form predicates 'role' and 'person'. Secondly, although the analysis certainly blocks our unwanted invalid inference, it also *fails* to sanction a basic and valid inference that we want to capture: If Vincent is Mia's husband, then Vincent is a husband: husband(Vincent). Thirdly, it is not clear how to extend the analysis to complex nominals: Bos's discussion suggests that we would get something like the following analysis:

(51) a. John's wooden toy disappeared. =
b. ∃x.thing(x) ∧ ∃y.role(x,y) ∧ wooden(x) ∧ toy(y) ∧ disappear(x)

We can successfully infer that 'Something wooden disappeared', but not that 'A toy disappeared', only that 'Something with the role of a toy disappeared'.

4 A HIGHER ORDER $\mathsf{of}_{\mathsf{ee(et)t}}$ RELATION

There is a relatively simple solution, linguistically at least, to this problem. Intuitively it is clear that what allows the invalid inferences to go through is that any binary possessive relation simply relates possessor and possessed, but does not capture in what respect the possessive relation holds. This respect is the property denoted by the N' constituent following the possessive 'NP's' determiner (which may have

to be recovered by ellipsis in some cases). If we make our possessive morpheme in our grammar fragment slightly more complex by giving it a semantics as follows:

(52) $\text{poss} = \lambda O_{(et)t} P_{et} Q_{et}.O(\lambda y.\exists x.P(x) \wedge \text{of}_{ee(et)t}(x,y,P)) \wedge Q(x)$

then we now make the respect in which the possessive relation holds an explicit argument of the 'of' placeholder relation. The "of" predicate is now of type ee(et)t: a function from individuals to individuals to properties to truth values. This is sufficient to block our unwanted inference:

(53) a. A: Smith is Bill's plumber.
b. $\Rightarrow_{=}$ plumber(Smith) ∧ of(Smith,Bill,plumber)

(54) a. B: Smith is also a decorator.
b. $\Rightarrow_{=}$ decorator(Smith)

(55) a. C: Smith is Bill's decorator?
b. $\Rightarrow_{=}$ decorator(Smith) ∧ of(Smith, Bill,decorator)

Now the unwanted inference does not go through. Note that this analysis is rather uncompositional, by the criteria we outlined earlier, in that one word corresponds to two identical non-logical constants in the logical form. We can make the analysis simpler and more compositional by dropping the repetition at the cost of two additional second order axioms:

(56) $\text{poss} = \lambda OPQ.\exists x.O(\lambda y.\text{of}(x,y,P)) \wedge Q(x)$

(57) Smith is Bill's plumber/brother. $\Rightarrow_{=}$ of(Smith,Bill,plumber/brother)

In order to recover the inference that Smith is a plumber, or where the noun is relational and interpreted relationally, as in 'a brother of Bill' we need these axiom schemata:

(58) a. A: $\forall xyP.\text{of}(x,y,P) \rightarrow P(x)$ ***(sortal N)***
b. B: $\forall xyP.\text{of}(x,y,P) \rightarrow P\text{-of}(x,y)$ ***(relational N)***

Note that we do not have to resolve 'of' to avoid bad inferences, and we do not need to distinguish sortal and relational N syntactically: these axioms capture their semantic differences.

Note also that unlike Bos's suggested analysis, ours generalises smoothly to complex nominal cases:

(59) a. John's wooden toy disappeared.
b. $\exists$x.of(x,John,λy.wooden(y) $\wedge$ toy(y)) $\wedge$ disappeared(x)

Via axiom A we can deduce:

(60) $\exists$x.[λy.wooden(y) $\wedge$ toy(y)](x) $\wedge$ disappeared(x)

and then by β-reduction and conjunction both that:

(61) a. A toy disappeared.
b. $\exists$x.toy(x) $\wedge$ disappeared(x)

and that:

(62) a. Something wooden disappeared.
b. $\exists$x.wooden(x) $\wedge$ disappeared(x)

Semantically it also follows from this last sentence that:

(63) John's toy disappeared: $\exists$x.of(x,John,toy) $\wedge$ disappeared(x)

However, in order for us to be able to show this proof-theoretically we need something more elaborate. Intuitively, we want to be able to say that if of(x,y,P) and P implies Q, then also of(x,y,Q). If Fido is John's cat, and all cats are animals, then Fido is John's animal. Something like the following would suffice for this particular case, where the entailment involves conjunction, but later we will need something more general:

(64) $\forall$xyPQ. of(x,y,λz.P(x) $\wedge$ Q(z)) $\rightarrow$ of(x,y,P) $\wedge$ of(x,y,Q)

We will return to such cases below.

5 COMPUTATIONAL IMPLICATIONS

Our second-order analysis may be linguistically fine, but computationally it does not yet solve our problems. As already remarked, we cannot do second (or higher) order logic theorem proving automatically

except for some very special restricted cases, beyond which our analysis lies. So although we cannot hope for a fully general solution to the problem of automated inference for analyses like the ones developed so far using second order logical forms, in this section we will explore some heuristic techniques which may enable us to implement a special purpose first order solution, still bearing in mind that we only have computationally efficient reasoning for fragments of first order logic.

In this section we explore two alternatives, both of which try to make our second order reasoning look like first order proofs.

5.1 *Encoding via combinators*

As illustrative examples let us focus on some simple possessive inferences we want to capture:

(65) Bill is John's dentist $\models$ Bill is a dentist

Our earlier analysis, after equality simplications, will give these sentences the following logical forms:

(66) of(Bill,John,dentist) $\models$ dentist(Bill)

A second slightly more complex example we would like to be able to handle is:

(67) a. John's wooden toy disappeared. $\models$
b. John's toy disappeared.
c. A toy disappeared.

(68) a. $\exists$x.of(x,John,λy.(wooden(y) $\wedge$ toy(y))) $\wedge$ disappeared(x) $\models$
b. $\exists$x.toy(x) $\wedge$ of(x,John,toy) $\wedge$ disappeared(x)
c. $\exists$x.toy(x) $\wedge$ disappeared(x)

We will assume the following axioms, introduced earlier:

Axiom A: $\forall$xyP.of(x,y,P) $\rightarrow$ P(x)
Axiom B: $\forall$xyPQ.of(x,y,λz.P(z) $\wedge$ Q(z)) $\rightarrow$ of(x,y,P) $\wedge$ of(x,y,Q)

Notice that in the intended application of these axioms, applicability would be determined by higher order matching (which **is** decidable: Stirling (2010)) and thus would generalise to sequences of three or

more conjuncts inside the lambda term. However, our approximation will not behave in this way and so in reality we will need a version for 2, 3, etc. conjuncts.

The basic idea is to encode higher order terms as first order expressions via combinators, following Hurd (2002) who used this technique to automate some of the components of a human-assisted higher order proof. Our logical expressions are less general than those treated by Hurd, since only second order arguments are involved, and they are either single predicate constants, or lambda terms with complex terms formed by connectives, but no quantifiers, in their body. We can thus apply the usual first order normal form transformations needed for a resolution or tableau theorem prover to our logical forms to obtain clauses (disjunctions of literals), with the extra feature that any second order arguments like those we are using are 'frozen': their outermost lambda functor will be regarded as a function symbol and no transformations will take place inside that lambda term.

We now transform each literal. The first step is to represent literals in applicative form, using a two-argument functor 'a' meaning 'apply'. Since 'a' is a function symbol and not a predicate, to respect first order syntax and semantics we have to wrap a dummy predicate 'p' around the translation:

(69) a. sleep(john) = p(a(sleep,john))
b. like(john,jane) = p(a(a(like,john),jane))

Now we can represent predicate variables as ordinary first order variables, so that for example ∃P.P(j) = ∃P.p(a(P,j)), where the occurrences of P on the right hand side are first order.

Our axiom A, in implicational rather than clausal form, now looks like this:

(70) p(a(a(a(of,X),Y),Q)) → p(a(Q,X))

However, the more complex axiom B has the following form at this stage, still containing a lambda expression:

(71) p(a(a(a(of,X),Y),λZ.a(a(and,a(P,Z)),a(Q,Z)))) →
p(a(a(a(of,X),Y),Q)) ∧ p(a(a(a(of,X),Y),R))

We need to eliminate all lambda expressions, of course. It is well known that we can completely eliminate variables from a lambda-

Figure 4: Translation function T: eliminating variables

T[x]	⇒ x	
T[(E1 E2)]	⇒ (T[E1] T[E2])	
T[λx.E]	⇒ (**K** T[E])	(if x is not free in E)
T[λx.x]	⇒ **I**	
T[λx.λy.E]	⇒ T[λx.T[λy.E]]	(if x is free in E)
T[λx.(E1 E2)]	⇒ (**S** T[λx.E1] T[λx.E2])	(if x is free in both E1 & E2)
T[λx.(E1 E2)]	⇒ (**C** T[λx.E1] T[E2])	(if x is free in E1 but not E2)
T[λx.(E1 E2)]	⇒ (**B** T[E1] T[λx.E2])	(if x is free in E2 but not E1)
T[λx.(E x)]	⇒ T[E]	(if x is not free in E: this is eta reduction)

calculus based logic by using 'combinators'. We can use this fact to further try to squeeze our second order expression into something that can be handled by a first order prover. There are many variant formulations of variable-free combinator calculi, but we will use a familiar one, also used by Hurd:

Ix	=	x (identity)
Kxy	=	x (make constant function)
Sxyz	=	xz(yz) (generalised application)
Cfxy	=	fyx (special case of S)
Bfgx	=	f(gx) (special case of S)

For completeness, we give the usual definition of a translation function **T** that will eliminate lambda terms and their variables (Figure 4), where E1 and E2 are any well formed HOL expression.

Now our axiom B looks like this, in implicational form:

(72) p(a(a(a(of,X),Y),a(a(**S**,a(a(**B**,and),Q)),R))) →
p(a(a(a(of,X),Y),Q)) ∧ p(a(a(a(of,X),Y),R))

Given axiom B and the applicative logical form for 'Bill is John's dentist':

(73) of(Bill,John,dentist) = p(a(a(a(of,Bill),John),dentist))

it is (relatively!) easy to see that the applicative version of this logical form will (first order) unify with the antecedent of the implication in the applicative form of axiom B, with bindings X=Bill, Y=John, Q=dentist allowing us to deduce:

(74) p(a(dentist,Bill)) = dentist(Bill)

Perhaps less easy to see, as the applicative forms become less human readable, is that we can also make some of the deductions we wanted from:[5]

(75) a. John's wooden toy disappeared.

b. $\exists x.of(x,John,\lambda y.(wooden(y) \wedge toy(y))) \wedge disappeared(x)$

Using axiom B (in applicative form) we can deduce the equivalent of ...of(x,John,toy)... and from axiom A ...toy(x)... enabling us to prove the queries:

(76) a. $\exists x.toy(x) \wedge of(x,John,toy) \wedge disappeared(x)$

b. $\exists x.toy(x) \wedge disappeared(x)$

5.2 *Adjective inferences*

We can encode our adjective inferences in the same way:

(77) a. $\forall xP.\ small(x,P) \rightarrow P(x) \Rightarrow$

b. $p(a(a(small,X),P)) \rightarrow p(a(P,X))$

(78) a. $\forall xPQ.\ small(x,\lambda y.P(y) \wedge Q(y)) \rightarrow P(x) \wedge Q(x) \Rightarrow$

b. $p(a(a(small,X),a(a(\mathbf{S},a(a(\mathbf{B},and),a(a(\mathbf{B},P),\mathbf{I}))),a(a(\mathbf{B},Q),\mathbf{I}))))$

c. $\rightarrow a(P,X) \wedge a(Q,X)$

These axioms and others will enable us to capture inferences like:

(79) a. Jones is a short red-haired farmer. $\models$

b. Jones is red-haired.

c. Jones is a farmer.

d. Jones is not a tall red-haired farmer.

Note that it does not on the intended readings of these sentences automatically follow that 'Jones is not a tall farmer'.

5.3 *An alternative approach*

All these examples so far work, but I find in general that this method is clumsy, for a number of reasons. Firstly, we have a rather cumbersome sequence of translations to carry out: from logical form to clausal form, then to applicative form and finally to combinator form. And in order

[5] Translations and proofs tested with Prover9.

to interpret the answers we get from our first order prover we need to reverse this process, particularly for cases where we are trying to answer a wh-question. To do this adequately we need to keep track of the unifying substitutions that allow the proof to go through.

Secondly, while this is an engineering rather than a theoretical problem, it is likely that on a large scale this approach would be very inefficient at the theorem proving stage: most (particularly Prolog-inspired) theorem provers rely heavily on predicate indexing for efficient search among a large set of clauses, and all our literals have the same dummy 'p' predicate. It is easy to think of other indexing schemes that would help, but they are not necessarily straightforward to add to existing systems.

Finally, notice that in the final form of the literals we may still have logical connectives. In order to capture all the inferences associated with these (we encoded a few in a flat-footed and uneconomical way a little earlier) we would have to efficiently axiomatise the inferences associated with connectives inside lambda terms. This is a little reminiscent of what would be needed to axiomatise various forms of property theory (Chierchia *et al.* (1989); Turner (1992); Fox and Lappin (2005)) and would lead to an explosion of low level axioms that carry no weight theoretically but are disastrous computationally.

It may therefore be worth exploring an alternative approach, which combines some of the features of the techniques already described. Looking at the properties of the inference examples discussed so far it seems we need to be able to do several things:

1. replace second order terms by some kind of first order constant which retains a unique link to the second order term that it replaces,
2. be able to reason using the internal structure of the second order term where it is more complex than a predicate constant,
3. ensure that this reasoning does not go beyond FOL.

One way of achieving this is to regard our second order axioms as rewriting or translation schemata which are applied to the compositionally derived logical form in order to produce one or more "compiled" first order equivalents. This has some features of a kind of on-the-fly reification of the type discussed earlier but one which does not require pre-computation.

This leads operationally to a picture like the following:

Partly second order logical form ⇒
Second order rewriting schemata ⇒
Expanded set of first order LFs ⇒
FOL Theorem prover

We reinterpret our existing axioms as rewriting schemata: we match them to an input logical form using higher order matching (which as already remarked, is decidable), and then if necessary beta-reduce the results. We also need a "reification" function: a kind of hash function guaranteed to give a unique first order constant for each different second order argument we give it. To illustrate, we take one of our earlier axioms (there will be one for each relevant adjective of this semantic type), which says that if you are old for a P, then you are a P:

(80) $\forall xP.\ old(x,P) \rightarrow P(x)$

We reconstrue this as a rewriting rule:

(81) Adj(old): $old(x,P) \Rightarrow old(x,hash(P)) \wedge P(x)$

In order for this to give us the results we want, we have to define 'hash' as a function which produces a unique symbol of type *e* for its argument (i.e. the same argument gives the same symbol guaranteed to be unique to that argument). More on this in a moment, but first, to illustrate:

(82) Harvard is an old American university:
$old(harvard, \lambda y.\ american(y) \wedge university(y)) \Rightarrow$ (via A)
$old(harvard,*AU*) \wedge [\lambda y.\ american(y) \wedge university(y)](harvard) \Rightarrow_{beta}$
$old(harvard,*AU*) \wedge american(harvard) \wedge university(harvard)$

'*AU*' is of course the constant produced by 'hash(λy. american(y) $\wedge$ university(y))'. We can think of such constants as denoting a first-order proxy for the property described by the second order argument to 'hash', reminiscent of the output of nominalisation operators in property theory.

We cannot give a sound and complete definition for a function such as 'hash' exactly, because ideally we want it to give the same result for logically equivalent lambda-terms, and of course we cannot fully compute this logical equivalence. But we can approximate

by doing various preprocessing operations: (i) inside lambda terms, reducing expressions involving connectives to some kind of normal form, and (ii) imposing a lexicographic ordering on predicates inside disjunctions and conjunctions, so that, for example λx.P(x) $\wedge$ Q(x) and λx.Q(x) $\wedge$ P(x) will count as the same. There may be other useful heuristics, too: this is essentially the 'equivalence of logical form problem' often discussed in the sentence generation literature (Shieber (1993)).

We can now reinterpret our earlier axioms capturing the relation between, say, antonymous adjectives by treating all the variables in them as first order:

(83) old(harvard,*AU*) $\wedge$ $\forall$xy.old(x,y) $\rightarrow$ $\neg$young(x,y) $\models$
$\neg$young(harvard,*AU*)

These can stay as axioms, added as background knowledge: they are not needed in the rewriting process.

We can deal with combinations of possessive and adjective inferences in the same way. Our main rewriting schema for possessives is now:

(84) Possessive: of(x,y,P) $\Rightarrow$ of(x,y,hash(P)) $\wedge$ P(x)

This interacts with Adj(old), an output of the rewriting schemata for adjectives, and so we have to recursively apply these rewritings:

(85) John's old wooden toy broke. =
$\exists$x.of(x,john, λy.old(y, λz.wooden(z) $\wedge$ toy(z))) $\wedge$ broke(x)

via Possessive:

(86) $\exists$x.of(x,john,*OWT*)$\wedge$[λy.old(x,λz.wooden(z)$\wedge$toy(z))](x)$\wedge$broke(x)
$\Rightarrow_{\beta}$
$\exists$x.of(x,john,*OWT*) $\wedge$ old(x,λz.wooden(z) $\wedge$ toy(z)) $\wedge$ broke(x)

via A:

(87) $\exists$x.of(x,john,*OWT*) $\wedge$ old(x,*WT*) $\wedge$ [λz.wooden(z) $\wedge$ toy(z)](x) $\wedge$ broke(x)

which beta-reduces to:

(88) $\exists$x.of(x,john,*OWT*) $\wedge$ old(x,*WT*) $\wedge$ wooden(x) $\wedge$ toy(x) $\wedge$ broke(x)

With this machinery we can capture the following inferences:

(89) Harvard is an old American university. ⊨
Harvard is a university.
Harvard is American.
Harvard is an American university.

but, correctly, it does not follow that:

(90) Harvard is an old university.

Similarly, we can capture:

(91) John's old wooden toy broke. ⊨
A toy broke.
A wooden toy broke.

However, we cannot yet capture inferences to:

(92) a. John's toy broke.
b. John's wooden toy broke.

or inferences such as:

(93) Bush is a former US President. ⊨
Bush is a former President.

Earlier, we had an axiom which in effect distributed over conjunctions: $\forall xyPQ.of(x,y,\lambda z.P(z) \wedge Q(z)) \rightarrow of(x,y,P) \wedge of(x,y,Q)$ We could introduce an analogous axiom for privative adjectives like "former", at least for those for which inferences within their complement are transparent.[6] If we assume that "US president" is to be analysed as involving an implicit possessive, then our example will be analysed as follows:

(94) a. Bush is a former US president.
b. former(Bush,λx.of(x,US,president))

In order to capture the inference we will need an axiom like:

(95) $\forall xyP.p\text{-}adjective(x,\lambda y.of(x,y,P)) \rightarrow p\text{-}adjective(x,P)$

[6] I am at a loss to provide a characterisation of the property that will distinguish apparently valid privative inferences – like the Bush one – from the invalid "former fussy eater" to "former eater" example discussed earlier.

Noting that "US president" entails "president", and that $\lambda z.P(z) \wedge Q(z)$ entails both P and Q we might be tempted to propose more general axioms like:

(96) a. $\forall xypq.\ of(x,y,p) \wedge entails(p,q) \rightarrow of(x,y,q)$

b. $\forall xPQ.\ p\text{-}adjective(x,P) \wedge entails(P,Q) \rightarrow p\text{-}adjective(x,Q)$

However, these axioms are much *too* general. Suppose it is true that all footballers are also gamblers. Then "Smith is a former footballer" would wrongly entail that "Smith is a former gambler". Likewise, suppose it is true that all members of parliament are lawyers. Then "Smith is my member of parliament" would wrongly entail "Smith is my lawyer".[7] What we need to do is restrict the type of entailment considered to entailments valid simply on the basis of the logical forms of the second order predicates we are dealing with. This will usually involve reconstructing the inferences that are implicit in relations between our hash generated constants like *OWT*, *WT*, and so on.

Again we seem to run up against an irreducible case of inference involving second order properties. But we can, in this type of case at least, take advantage of the fact that the lambda terms involved are only a few beta-reductions away from something that is first order. If these terms are predicated of a first order entity then after beta-reduction the resulting formulae will also be first order. This suggests that we might be able to take these second order properties and use them to construct something that we can evaluate with our theorem prover.

We can write the axioms we need quite literally as:

(97) a. $\forall xypq.\ of(x,y,p) \wedge hash\text{-}entails(p,q) \rightarrow of(x,y,q)$

b. $\forall xpq.\ p\text{-}adjective(x,p) \wedge hash\text{-}entails(p,q) \rightarrow p\text{-}adjective(x,q)$

assuming that we are applying this to the output of our rewriting schemata, so that "p" and "q" are first order variables that will range over the constants generated by the "hash" function. We will treat the predicate 'hash-entails' as interpreted partly by a 'procedural attachment' (an old idea, but one recently used in natural language inference by Waldinger and Shrager (2008)) in our base theorem prover. The procedurally attached predicate will be evaluated by calling a

[7] Thanks to a referee for suggesting such examples.

separate instantiation of that theorem prover, in which any general background knowledge axioms are available, but none of the linguistically derived information related to the top level inference in which we are currently engaged.

In order to operationalise the "hash-entails" predicate we also need some housekeeping. We need the "hash" function to record the connection between the second order term it takes as input and the first order constant it gives as output. We will assume that this is achieved via a predicate recording the inputs and outputs to the hash function during the application of the various schemata described above, e.g.

(98) hashOutput(*AU*, λx.American(x) $\wedge$ university(x))

It is necessary to do this recursively to include any other hash-generated constants, for example:

(99) a. hashOutput(*OWT*, λA.old(A,*WT*) $\wedge$ wooden(A) $\wedge$ toy(A))

b. hashOutput(*WT*, λA.wooden(A) $\wedge$ toy(A))

We will also have need of a default case for those sentences in which the second order argument of 'of' or adjectives is not itself complex, as for example:

(100) hashOutput(toy_e, toy_{et})

We can now define the 'hash-entails' predicate as follows:

(101) $\forall p_e q_e R_{et} S_{et}$. hash-entails(p,q) $\Longleftrightarrow$
hashOutput(p,R) $\wedge$ hashOutput(q,S) $\wedge$ prove($\neg(\exists$x.R(x) $\wedge$ $\neg$(S(x))))

where 'prove' represents a call to a separate instance of our theorem prover as described above. The assumption is that 'R(x)' and 'S(x)' etc. represent a full beta reduction of 'R' and 'S' applied to 'x', so that the formula to be proved ends up as strictly first order.

Now the inference we want will go through, with logical forms as shown:

(102) a. John's old wooden toy disappeared.

b. $\exists$x.of(x,john,*OWT*) $\wedge$ old(x,*WT*) $\wedge$ wooden(x) $\wedge$ toy(x) $\wedge$ disappear(x)

(103) a. Did John's toy disappear?

b. $\exists$x.of(x,john,toy) $\wedge$ toy(x) $\wedge$ disappear(x)

The relevant instantion of the axiom involving 'hash-entails' will be:

(104) of(x,John,*OWT*) ∧ hash-entails(*OWT*,toy) → of(x,John,toy)

The definition of "hash-entails" will give us:

(105) hashOutput(*OWT*,λx.old(x,*WT*) ∧ wooden(x) ∧ toy(x)) ∧
hashOutput(toy,toy) ∧
prove(¬(∃y. (old(y, *WT*) ∧ wooden(y) ∧ toy(y)) ∧ (¬toy(y))))

and the inference that calling the procedural predicate 'hash-entails' checks via 'prove' will be:

(106) ¬(∃y.(old(y,*WT*) ∧ wooden(y) ∧ toy(y)) ∧ ¬(toy(y)))

which is clearly almost trivially valid.

There is a minor wrinkle in applying the axiom concerning privative adjectives. Recall that we accounted for the invalidity of the inference from "Bush is a former US president" to "Bush is a president" by not allowing the axiom ∀xP.adj(x,P) → P(x) to apply to such adjectives. In our rewriting framework this means that we do not rewrite adj(x,P) as adj(x,hash(P)) ∧ P(x). Since "hash-entails" is a relation between first-order entities, there will be nothing for it to work with. The solution is to add a rewrite specific to this class of adjectives to yield former(Bush, *USP*). We will also need to arrange for "hashOutput" to recursively apply even where the usual rewriting has not taken place, to give:

(107) a. hashOutput(*USP*,λx.of(x,US,president$_{et}$))
b. hashOutput(president$_{e}$,president$_{et}$)

This approach has been fully implemented using a unification grammar to produce the logical forms, and a combination of two resolution theorem provers to carry out the inferences, with one being called during the evaluation of the "entails" predicate. Appendix gives a FraCaS style corpus of the natural language inferences described in this paper, all of which are successfully handled by this system.

6 CONCLUSIONS

There are some simple but quite central linguistic constructs that seem to need second order inference. It may be possible to reduce the nec-

essary inferences to those capturable in first order logic via some standard variant of reification, but the apparent requirement for a potentially infinite number of types of new first order individuals caused by recursive adjective modification seems a barrier to this. An alternative approach using translation to FOL via combinators may work, but is a little clumsy and may not generalise fully.

A better approach seems to be to pre-process the second order logical forms using second (or perhaps higher) order matching, rewriting in a forward-chaining manner to produce first-order logical forms in which second order arguments are represented by first order constants: a different type of reification, in some sense. The inferential content of these particular originally second order terms can be recaptured via a subsidiary set of first order inferences using a procedurally attached predicate which calls a separate instance of a theorem prover, after suitable beta-reductions produce first order forms.

It is an interesting question as to what extent this strategy, or variants of it, can be used to handle other types of second or perhaps higher order inference. Extensions to cover various forms of the comparative construction seem straightforward. It remains to be seen whether other second order inference phenomena such as intensional verbs may also yield to this approach.

7 ACKNOWLEDGEMENTS

This paper has been a long time in gestation. Talks based on parts of it have been given to meetings of the MOLTO project and to the Controlled Natural Language conference in Zurich, 2012; to the Computational Linguistics seminars at Kings College London and at Oxford University, 2013; and to Nuance Research Labs in Sunnyvale, CA, in 2013. I am grateful for comments received on all these occasions (particularly from Emmon Bach in Oxford), for some suggestions from Ash Asudeh, and also to Johan Bos for useful discussion of these and similar topics over many years. Four referees and two editors made many helpful criticisms and suggestions for which I am also grateful.

APPENDIX

Phenomenon	Expected answer
Intersective adj:	
Jones is a Welsh musician.	
Is Jones Welsh?	Yes
Is Jones a musician?	Yes
All musicians are teachers.	
Is Jones a teacher?	Yes
Is Jones a Welsh teacher?	Yes
Gradable adj:	
Mickey is a large mouse.	
Is Mickey a mouse?	Yes
Is Mickey large?	No proof found
All mice are animals.	
Is Mickey an animal?	Yes
Is Mickey a large animal?	No proof found
All mice are small animals.	
Is Mickey a small animal?	Yes
Mickey isn't a large animal?	Yes
Mickey isn't a small mouse?	No proof found
Privative type 1:	
Jones is a former Welsh minister.	
Is Jones a minister?	No proof found
Is Jones a Welsh minister?	No proof found
Is Jones a former minister?	Yes
Jones isn't a Welsh minister?	No proof found
Privative type 2:	
Jones owns a fake diamond.	
Does Jones own a diamond?	No proof found
Zirconia is a fake diamond.	
Zirconia isn't a diamond?	Yes
Interaction with antonyms:	
John is a tall man.	
John isn't a short man?	Yes
Bill isn't a short man.	
Is Bill a tall man?	No proof found

Recursive adj modification:

Harvard is an old American university.	
Harvard is a university?	Yes
Harvard is an American university?	Yes
Harvard is an old university?	No proof found

Possessives:

John's brother is Bill.	
Bill is a doctor.	
Is Bill John's doctor?	No proof found
Smith is Bill's plumber.	
Is Smith a plumber?	Yes
Smith is a decorator.	
Is Smith Bill's decorator?	No proof found
Smith's essay's title intrigued Jones.	
An essay's title intrigued Jones?	Yes
A title intrigued Jones?	Yes
An essay intrigued Jones?	No proof found
Smith intrigued Jones?	No proof found

Combination of adj and possessive:

John's old wooden toy broke.	
Did John's toy break?	Yes
Did John's wooden toy break?	Yes
Did John's old toy break?	No proof found
Did an old wooden toy break?	Yes
Did an old toy break?	No proof found
Did a wooden toy break?	Yes
Did a toy break?	Yes

REFERENCES

Marilisa AMOIA and Claire GARDENT (2007), A First Order Semantic Approach to Adjectival Inference, in *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 185–192, Association for Computational Linguistics, Prague, http://www.aclweb.org/anthology/W07-1430.

Franz BAADER, Diego CALVANESE, Deborah L. MCGUINNESS, Daniele NARDI, and Peter F. PATEL-SCHNEIDER, editors (2003), *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, New York, NY, USA, ISBN 0-521-78176-0.

Johan BOS (2009), Computing Genitive Superlatives, in *Proceedings of the Eighth International Conference on Computational Semantics*, IWCS-8 '09, pp. 18–32, Association for Computational Linguistics, Stroudsburg, PA, USA, ISBN 978-90-74029-34-6, http://dl.acm.org/citation.cfm?id=1693756.1693763.

Johan BOS, Stephen CLARK, Mark STEEDMAN, James R. CURRAN, and Julia HOCKENMAIER (2004), Wide-Coverage Semantic Representations from a CCG Parser, in *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, pp. 1240–1246, Geneva, Switzerland.

Stergios CHATZIKYRIAKIDIS and Zhaohui LUO (2014), Natural Language Inference in Coq, *Journal of Logic, Language and Information*, 23(4):441–480, ISSN 0925-8531, doi:10.1007/s10849-014-9208-x, http://dx.doi.org/10.1007/s10849-014-9208-x.

Gennaro CHIERCHIA, Barbara H. PARTEE, and Raymond TURNER (1989), Introduction, in Gennaro CHIERCHIA, Barbara H. PARTEE, and Raymond TURNER, editors, *Properties, Types and Meaning. Volume I: Foundational Issues*, pp. 1–16, Kluwer, Dordrecht.

Robin COOPER, Dick CROUCH, Jan VAN EIJCK, Chris FOX, Josef VAN GENABITH, Jan JASPARS, Hans KAMP, David MILWARD, Manfred PINKAL, Massimo POESIO, and Steve PULMAN (1996), *Using the Framework*, LRE 62-051, The FraCaS Consortium, ftp://ftp.cogsci.ed.ac.uk/pub/FRACAS/del16.ps.gz.

Donald DAVIDSON (1967), The Logical form of Action Sentences, in Nicholas RESCHER, editor, *The Logic of Decision and Action*, pp. 81–95, University of Pittburgh Press: Pittsburgh.

Jos DE BRUIN and Remko SCHA (1988), The Interpretation of Relational Nouns, in *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, pp. 25–32, Association for Computational Linguistics, Buffalo, New York, USA, doi:10.3115/982023.982027, http://www.aclweb.org/anthology/P88-1004.

Guillermo DEL PINAL (2015), Dual Content Semantics, Privative Adjectives, and Dynamic Compositionality, *Semantics and Pragmatics*, 8(Article 7):1–53.

Chris Fox and Shalom Lappin (2005), *Foundations of Intensional Semantics*, Blackwell.

Jerry R. Hobbs (1985), Ontological Promiscuity, in *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pp. 60–69, Association for Computational Linguistics, Chicago, Illinois, USA, doi:10.3115/981210.981218, http://www.aclweb.org/anthology/P85-1008.

Gerard Huet (1975), A Unification Algorithm for Typed λ-Calculus, *Theoretical Computer Science*, 1:27–57.

Joe Hurd (2002), An LCF-Style Interface between HOL and First-Order Logic, in Andrei Voronkov, editor, *Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings*, volume 2392 of *Lecture Notes in Computer Science*, pp. 134–138, Springer, ISBN 3-540-43931-5.

Johannes A. W. Kamp (1975), Two Theories about Adjectives, in Edward L. Keenan, editor, *Formal Semantics of Natural Language*, pp. 123–155, Cambridge University Press, Cambridge.

Daniel Lassiter and Noah D. Goodman (2017), Adjectival Vagueness in a Bayesian Model of Interpretation, *Synthese*, 194(10):3801–3836, doi:10.1007/s11229-015-0786-1, https://doi.org/10.1007/s11229-015-0786-1.

Bill MacCartney and Christopher Manning (2008), Modeling Semantic Containment and Exclusion in Natural Language Inference, in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 521–528, Coling 2008 Organizing Committee, http://aclweb.org/anthology/C08-1066.

William McCune (2005–2010), Prover9 and Mace4, http://www.cs.unm.edu/~mccune/prover9/.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki (2015), Higher-order Logical Inference with Compositional Semantics, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2055–2061, Association for Computational Linguistics, Lisbon, Portugal, https://aclweb.org/anthology/D15-1244.

Marcin Morzycki (2014), Modification, https://www.msu.edu/~morzycki/work/papers/modification_book.pdf, accessed Jan 3 2015.

Barbara H. Partee (2007), Compositionality and Coercion in Semantics: The Dynamics of Adjective Meaning, in Gerlof Bouma, Irene Krämer, and Joost Zwarts, editors, *Cognitive foundations of interpretation*, p. 145–161, University of Chicago Press.

Barbara H. PARTEE and Vladimir BORSCHEV (2003), Genitives, Relational Nouns, and Argument-modifier Ambiguity, in Ewald LANG, Claudia MAIENBORN, and Cathrine FABRICIUS-HANSEN, editors, *Modifying Adjuncts*, Interface Explorations, pp. 67–112, Mouton de Gruyter, Berlin.

Stanley PETERS and Dag WESTERSTÅHL (2006), *Quantifiers in Language and Logic*, Clarendon Press, Oxford.

Stanley PETERS and Dag WESTERSTÅHL (2013), The Semantics of Possessives, *Language*, 89(4):713–759.

Jessica RETT (2014), *The Semantics of Evaluativity*, Oxford Studies in Theoretical Linguistics, Oxford University Press.

Stuart M. SHIEBER (1993), The Problem of Logical-form Equivalence, *Computational Linguistics*, 19(1):179–190, ISSN 0891-2017, `http://dl.acm.org/citation.cfm?id=972450.972460`.

Mark STEEDMAN (2012), *Taking Scope*, MIT Press.

Colin STIRLING (2010), Introduction to Decidability of Higher-Order Matching, in Luke ONG, editor, *Foundations of Software Science and Computational Structures*, volume 6014 of *Lecture Notes in Computer Science*, pp. 1–1, Springer Berlin Heidelberg, ISBN 978-3-642-12031-2, doi:10.1007/978-3-642-12032-9_1, `http://dx.doi.org/10.1007/978-3-642-12032-9_1`.

Zoltán Gendler SZABÓ (2017), Compositionality, in Edward N. ZALTA, editor, *The Stanford Encyclopedia of Philosophy*, Metaphysics Research Lab, Stanford University, summer 2017 edition.

Raymond TURNER (1992), Properties, Propositions and Semantic Theory, in Mike ROSNER and Rod JOHNSON, editors, *Computational Linguistics and Formal Semantics*, pp. 159–180, Cambridge University Press, Cambridge.

Richard WALDINGER and Jeff SHRAGER (2008), Answering Science Questions: Deduction with Answer Extraction and Procedural Attachment, *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*.