Second order inference in NL semantics

Stephen Pulman

Computational Linguistics Group,
Department of Computer Science,
Oxford University.
stephen.pulman@cs.ox.ac.uk

Aug 2012
Aims

- Capture some apparently trivial NL inferences (and lack of inferences).
- Assumptions: syntax-driven composition semantics producing logical form for a (disambiguated) parsed sentence
- Logical forms sent to automated theorem prover (resolution, tableau...).
- Statements added as ‘axioms’, questions (inferences) treated as ‘theorems’ to be proved.
- Yes/no questions: yes if there is a proof; Wh-questions: if a proof, return unifying substitutions as wh-value answers.

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: $\exists x.\text{rich}(x)$
Adjective inferences

Jones is a Welsh rugby player
\[\models\] Jones is Welsh
\[\models\] Jones is a rugby player

All rugby players are beer drinkers
\[\models\] Jones is a Welsh beer drinker, etc.

Minnie is a large mouse
\[\models\] Minnie is a mouse

All mice are animals
\[\models\] Minnie is an animal
\[\not\models\] Minnie is a large animal

Tony Blair is a former Prime Minister
\[\not\models\] Tony Blair is a Prime Minister

Smith showed an apparent proof of the theorem
\[\not\models\] Smith showed a proof of the theorem
Possessive inferences

Smith is Jones’s plumber
   \[\models\] Smith is a plumber

Smith is also a decorator
   \[\not\models\] Smith is Jones’s decorator

John’s wooden toy broke
   \[\models\] John’s toy broke
   \[\models\] a wooden toy broke
   \[\models\] a toy broke

A student’s textbook’s cover intrigued Jones
   \[\models\] A textbook’s cover intrigued Jones
   \[\models\] A cover intrigued Jones

John’s mother or father phoned
   John’s mother or John’s father phoned. etc.
Adjective semantics

Customary to distinguish three subclasses:

- **Intersective**: dead, Welsh, wooden, foreign...
  Adj N implies both Adj and N
- **Subsective**: tall, old, green, rigid...
  Adj N implies N, but only Adj-for-an-N, not Adj in general
- **Privative**: apparent, fake, former, alleged...
  Adj N does not imply N (may even imply not-N)

**Truth conditions**: \( D(x) = \text{‘denotation of } x\text{’} \)

Intersective: ‘Jones is a Welsh rugby-player’ true iff
\[
D(\text{Jones}) \in D(\text{Welsh}) \cap D(\text{rugby-player})
\]

Subsective: ‘Minnie is a large mouse’ true iff
\[
D(\text{Minnie}) \in \{ X \mid X \text{ a mouse larger than relevant standard} \}
\]

Privative: varies - ‘X is a former Y’ true iff
\[
D(\text{X}) \in D(\text{Y at earlier time}), \text{ etc.}
\]
Logical forms

Fine if we are just doing linguistics, but for computational purposes we need a logical form that will support the relevant inferences proof theoretically. No syntactic difference between types of Adj so build semantic differences into their LFs directly:

Assume syntax/semantics rules like:

- **NP** → Det N’ \(Det(N’)\)
- **N’** → Adj N’ \(Adj(N’)\)
- **N’** → N \(N\)
- **Adj** → wooden etc. \(\lambda P x.\text{wooden}(x) \land P(x)\)
- **Adj** → small, etc. \(\lambda P x.\text{small}(x, P)\)
- **Adj** → apparent, etc. \(\lambda P x.\text{apparent}(x, P)\)

- intersectional: we get the inferences we want immediately
- subjunctive: \(\text{small}(x, P)\) = ‘small by the standards relevant for \(P\)’.
  To get the inference that \(P(x)\) we add an axiom for each adj:
  \(\forall x P \text{adj}(x, P) \rightarrow P(x)\)
- privative: we do not add these axioms and so we (correctly) cannot infer from \(\text{apparent}(x, P)\) that \(P(x)\)
But the non-intersective Adj have second order arguments

Like this analysis, many natural language constructs are intrinsically higher order:

- Most dogs bark = $most(dog, bark)$
  \[\text{type}(\text{most}) = (et)(et)t, \text{type}(\text{dog}) = \text{type}(\text{bark}) = et\]
- John is very tall = $very(tall)(john)$
  \[\text{type}(\text{very}) = (et)(et), \text{type}(\text{tall}) = et, \text{type}(\text{john}) = e\]
- But we only have automated theorem provers for first order logic

Reification or ‘ontological promiscuity’ attempts to avoid the problem: e.g. event analyses of adverbs:

- John runs quickly = $quickly(run)(john)$
  \[\Rightarrow \exists e.\text{run}(e, john) \land \text{quick}(e)\]
- or the ‘standard translation’ of modal logic:
  \[\Box p \Rightarrow \forall x. R(thisWorld, x) \rightarrow P(x)\]
Reification of standards of Adj-ness?

But it’s not obvious how such a strategy could help here:

John is a tall man =?

\[ \exists s. \text{tall}(john, s) \land \text{man}(john) \land \text{tallness-for-men}(s) \]

Not clear how to model interaction with related predicates:

John is a tall man \( \models \) John is not a short man

\[ \forall xyz. \text{tall}(x, y) \land \text{tallness-for-men}(y) \rightarrow \\
\neg (\text{short}(x, z) \land \text{shortness-for-men}(z)) \]

Potentially infinite number of such ‘s’ predicates, and therefore such relatedness axioms (ignore conjunctive readings):

this is an old building, an old American building

this is an old English building, old Anglo-Saxon religious site...

Whereas the higher order version generalises cleanly:

...old American building = old(this, \lambda x. \text{American}(x) \land \text{building}(x))

and the interaction with related predicates only needs one axiom:

\[ \forall x P. \text{old}(x, P) \rightarrow \neg (\text{young}(x, P)) \]
Possessive semantics

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.

The precise possessive relation is usually contextually inferred:

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...

Relational vs. sortal nouns: if there is a relational noun, that usually provides the relation, but not invariably:

*The history teacher had an argument with one of his parents.*
(parents’ evening context: parents who came to see him)
*This time, Maria’s evil daughter will be Goneril*
(acting King Lear context: daughter played by Maria)
An initial simple analysis

A simple analysis (e.g. [Bos et al., 2004, Steedman, 2012]) takes the possessive morpheme ’s (or just ’ for plurals) to be a function from NP meanings to Det meanings introducing an abstract ‘of’ or ‘poss’ relation:

John’s friend is Bill = ∃x. friend(x) ∧ of(x, John) ∧ x=Bill
But this analysis is wrong!

A: John’s brother is Bill =
\[ \exists x. \text{brother}(x) \land \text{of}(x, \text{John}) \land x=\text{Bill} \]
\[ \Rightarrow= \text{brother}(\text{Bill}) \land \text{of}(\text{Bill},\text{John}) \]

B: Bill is a doctor =
\[ \text{doctor}(\text{Bill}) \]

C: Is Bill John’s doctor? =
\[ \exists x. \text{doctor}(x) \land \text{of}(x, \text{John}) \land x=\text{Bill} \]
\[ \Rightarrow= \text{doctor}(\text{Bill}) \land \text{of}(\text{Bill},\text{John}) \]

- but now C is provable from A and B, incorrectly.
Contextual interpretation

More sophisticated analyses ([Partee and Borschev, 2003], [Peters and Westerståhl, 2006]) require contextual interpretation of a predicate variable $R_{gen}$ or $Poss$.

If we interpret ‘of’/‘Poss’/$R_{gen}$’ in A as the two-place relation ‘brother(Bill,John)’, and as something else in C, then the incorrect inference will not be made.

A: John’s brother is Bill =
$$\exists x. brother(x) \land Poss(x, John) \land x=Bill$$
$$\Rightarrow = brother(Bill) \land brother(Bill,John)$$

B: Bill is a doctor =
$$doctor(Bill)$$

C: Is Bill John’s doctor? =
$$\exists x. doctor(x) \land Poss(x, John) \land x=Bill$$
$$\Rightarrow = doctor(Bill) \land doctor-employed-by(Bill,John)$$
Contextual interpretation

Although our invalid inference will not go through when relational nouns are involved, we cannot always guarantee this for sortal nouns (or relational nouns when interpreted sortally):

A: Smith is Bill’s plumber = (interpret ‘Poss’ as ‘works-for’)
   \[\text{plumber}(Smith) \land \text{works-for}(Smith, Bill)\]

B: Smith is also a decorator
   \[\text{decorator}(Smith)\]

C: Is Smith Bill’s decorator?
   \[\text{decorator}(Smith) \land \text{works-for}(Smith, Bill)\]

It’s surely difficult to argue that ‘Poss’ should be instantiated differently in A and C.
One solution: an ‘of’ relation with a second order argument

\[ 's = \lambda OPQ. \exists x. P(x) \land O(\lambda y. of(x,y,P)) \land Q(x) \]

A: Smith is Bill’s plumber =

\[ \text{plumber}(Smith) \land of(Smith,Bill,\text{plumber}) \]

B: Smith is also a decorator =

\[ \text{decorator}(Smith) \]

C: Is Smith Bill’s decorator? =

\[ \text{decorator}(Smith) \land of(Smith, Bill,\text{decorator}) \]

Now the unwanted inference does not go through. We can remove the duplicate ‘P’ with additional axioms:

\[ 's = \lambda OPQ. \exists x. O(\lambda y. of(x,y,P)) \land Q(x) \]

Smith is Bill’s plumber = \( of(Smith,Bill,\text{plumber}) \)

A: \( \forall xy P. of(x,y,P) \rightarrow P(x) \) (for sortal N)

B: \( \forall xy P. of(x,y,P) \rightarrow P-of(x,y) \) (for relational N)

We don’t even have to resolve ‘of’ to avoid bad inferences, and we don’t need to distinguish sortal and relational N syntactically.
Generalises to complex \( N' \)

John’s wooden toy disappeared.

\[
\exists x. \text{of}(x, \text{John}, \lambda y. \text{wooden}(y) \land \text{toy}(y)) \land \text{disappeared}(x)
\]

Via axiom A we can deduce: \(...[\lambda y. \text{wooden}(y) \land \text{toy}(y)](x)\)... and then by \( \beta \)-reduction that:

\[
\exists x. \text{of}(x, \text{John}, \lambda y. \text{wooden}(y) \land \text{toy}(y)) \land \text{disappeared}(x) \\
\land \text{wooden}(x) \land \text{toy}(x)
\]

Now we can capture the inferences:

A toy disappeared: \( \exists x. \text{toy}(x) \land \text{disappeared}(x) \)

and:

Something wooden disappeared: \( \exists x. \text{wooden}(x) \land \text{disappeared}(x) \)

Semantically (though not yet proof-theoretically) it also follows that:

John’s toy disappeared: \( \exists x. \text{of}(x, \text{John}, \text{toy}) \land \text{disappeared}(x) \)

To capture this we need an axiom to the effect that if \( \text{of}(x, y, P) \) and if property P entails property Q, then \( \text{of}(x, y, Q) \)
Linguistically fine, but computationally OK?

Unhappily not, because our ‘of’ predicate has a second order argument, like our analysis of adjective modification, so we cannot do any automated inference directly with it.

Could reification help us here?

Johan Bos ([Bos, 2009]) has suggested a reification approach to this problem.

We translate sentences like ‘Vincent is Mia’s husband’ as:

$$\text{person}(Vincent) \land \exists y.\text{role}(Vincent,y) \land \text{husband}(y) \land \text{of}(y,Mia)$$

paraphrased as something like ‘Vincent is a person who is playing the role of Mia’s husband’.

(NB Bos attributes the idea to Yuliya Lierler and Vladimir Lifschitz)
I don’t find this satisfactory

- Compositionality: where do ‘role’ and ‘person’ come from?
- Blocks unwanted inference, but does not support a valid inference: If Vincent is Mia’s husband, then Vincent is a husband: 
  \[ \text{husband}(Vincent) \].
- Only obvious way to restore the missing inference would be to add a (second order) axiom: 
  \[ \forall xy P.\text{role}(x,y) \land P(y) \rightarrow P(x) \]
- And not clear how to extend to complex N’: John’s wooden toy disappeared 
  \[ \exists x.\text{thing}(x) \land \exists y.\text{role}(x,y) \land \text{wooden}(y) \land \text{toy}(y) \ldots \]
- We want to infer ‘Something wooden disappeared’, but can roles be wooden?
Suggestion 1: encode (some) HOL as FOL

Two examples we want to capture:

Bill is John’s dentist $\models$ Bill is a dentist

$$\text{of}(\text{Bill}, \text{John}, \text{dentist}) \models \text{dentist}(\text{Bill})$$

John’s wooden toy disappeared $\models$ A toy/ John’s toy disappeared

$$\exists x . \text{of}(x, \text{John}, \lambda y . (\text{wooden}(y) \land \text{toy}(y))) \land \text{disappeared}(x) \models$$

$$\exists x . \text{toy}(x) \land \text{disappeared}(x)$$

$$\exists x . \text{toy}(x) \land \text{of}(x, \text{John}, \text{toy}) \land \text{disappeared}(x)$$

Axioms $+$ beta-reduction:

A: $\forall xyP . \text{of}(x,y,P) \rightarrow P(x)$

B: $\forall xyPQ . \text{of}(x,y,\lambda z . P(z) \land Q(z)) \rightarrow \text{of}(x,y,P) \land \text{of}(x,y,Q)$ etc.

(It would be nice if someone added these (not very) higher order features to a FOL theorem prover. But in the meantime...)
Encoding: the gory details

Represent literals in applicative form:
\[ \text{sleep}(\text{john}) = p(a(\text{sleep}, \text{john})) \]
\[ \text{like}(\text{john}, \text{jane}) = p(a(a(\text{like}, \text{john}), \text{jane})) \]

Now we can represent predicate variables: \( P(j) = p(a(P, j)) \)

What’s the ‘p’ doing? Well, ‘a’ is actually a function symbol, so to respect FOL syntax we wrap a dummy predicate ‘p’ around the translation.

\[ \text{of}(x, y, \lambda z. P(z) \land Q(z)) = p(a(a(\text{of}, x), y), \lambda z. a(a(\text{and}, a(P, z)), a(Q, z))) \]

Now eliminate lambda expressions by using combinators:
\[
\begin{align*}
Ix &= x \\
Kxy &= x \\
Sxyz &= xz(yz) \\
Cfxy &= fyx \\
Bfgx &= f(gx)
\end{align*}
\]

\(^1\)The basic idea is taken from [Hurd, 2002]
Define function $T$ (translate), lambda-term to combinators:

- $T[x] \Rightarrow x$
- $T[(E_1 E_2)] \Rightarrow (T[E_1] \ T[E_2])$
- $T[\lambda x.E] \Rightarrow (K \ T[E])$ (if $x$ is not free in $E$)
- $T[\lambda x.x] \Rightarrow I$
- $T[\lambda x.\lambda y.E] \Rightarrow T[\lambda x.T[\lambda y.E]]$ (if $x$ is free in $E$)
- $T[\lambda x.(E_1 E_2)] \Rightarrow (S \ T[\lambda x.E_1] \ T[\lambda x.E_2])$ (if $x$ is free in both $E_1$ and $E_2$)
- $T[\lambda x.(E_1 E_2)] \Rightarrow (C \ T[\lambda x.E_1] \ T[E_2])$ (if $x$ is free in $E_1$ but not $E_2$)
- $T[\lambda x.(E_1 E_2)] \Rightarrow (B \ T[E_1] \ T[\lambda x.E_2])$ (if $x$ is free in $E_2$ but not $E_1$)
- $T[\lambda x.(E \ x)] \Rightarrow T[E]$ (if $x$ is not free in $E$: this is eta reduction)

Now our axioms look like this:

A: $p(a(a(a(of,X),Y),Q)) \rightarrow p(a(Q,X))$

B: $p(a(a(a(of,X),Y),a(a(S,a(a(B,and),Q)),R)))) \rightarrow$

\[ p(a(a(a(of,X),Y),Q)) \land p(a(a(a(of,X),Y),R)) \]
Capturing our inferences

A: \( p(a(a(a(of,X),Y),Q)) \rightarrow p(a(Q,X)) \)

B: \( p(a(a(a(of,X),Y),a(a(S,a(a(B,and),Q)),R))) \rightarrow p(a(a(a(of,X),Y),Q)) \wedge p(a(a(a(of,X),Y),R)) \)

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

Using axiom A we deduce \( p(a(dentist,Bill)) = dentist(Bill) \)

John’s wooden toy disappeared =
\[ \exists x. of(x,John,\lambda y.(wooden(y) \wedge toy(y))) \wedge disappeared(x) \]

Using axiom B we can deduce \( ...of(x,John,toy)\) and from A \( ...toy(x)\)
enabling us to prove the queries:
\[ \exists x. toy(x) \wedge disappeared(x) \]
\[ \exists x. toy(x) \wedge of(x,John,toy) \wedge disappeared(x) \]
Adjective inferences

We can encode our adjective inferences in the same way:

\[ \forall x P. \text{small}(x,P) \rightarrow P(x) \Rightarrow \]
\[ p(a(a(\text{small},X),P)) \rightarrow p(a(P,X)) \]
\[ \forall xPQ. \text{small}(x,\lambda y.P(y) \land Q(y)) \rightarrow P(x) \land Q(x) \Rightarrow \]
\[ p(a(a(\text{small},X),a(a(S,a(a(B,\text{and}),a(a(B,P),I))),a(a(B,Q),I)))),a(a(\text{and},a(P,X)),a(Q,X)))) \rightarrow \]
\[ p(a(a(\text{and},a(P,X)),a(Q,X)))) \]

These axioms and others will enable us to capture inferences like:
Jones is a short Welsh rugby-player
\[ \models \text{Jones is Welsh} \]
\[ \models \text{Jones is a rugby-player} \]
\[ \models \text{Jones is not a tall Welsh rugby-player} \]
(Does it follow that Jones is not a tall rugby player? Strictly speaking, no!)
All these examples work (NB translations and proofs tested with Prover9). But I find this method is clumsy:

- Logical form $\Rightarrow$ Applicative form $\Rightarrow$ Combinator form $\Rightarrow$ Clausal form
- To interpret the results we often need to partly reverse the process (e.g. for wh-answers that may be complex functions)
- Probably inefficient on a large scale (predicates all the same, deeply nested functions)
- And not complete: note that in the final form of the literals we still have logical connectives.
- We have to axiomatise the inferences associated with connectives inside lambda terms...
Suggestion 2: second order axioms as schemata

Partly higher order logical form ⇒ Forward chaining schemata ⇒ Expanded set of first order LFs ⇒ Theorem prover
Reinterpret axioms as rewriting schemata: match to input LF using higher order matching, then beta-reduce results:
\[ \forall x P. \text{small}(x, P) \rightarrow P(x) \Rightarrow \]
A: \text{small}(x, P) ⇒ \text{small}(x, \text{hash}(P)) \land P(x)
‘hash’ is something which produces a unique symbol of type e for its argument (same arg, same symbol).

Harvard is an old American university
\[ \text{old}(\text{harvard}, \lambda y. \text{american}(y) \land \text{university}(y)) \Rightarrow (\text{via A}) \]
\[ \text{old}(\text{harvard}, \text{AU}) \land [\lambda y. \text{american}(y) \land \text{university}(y)](\text{harvard}) \]
\[ \text{old}(\text{harvard}, \text{AU}) \land \text{american}(\text{harvard}) \land \text{university}(\text{harvard}) \]
And via our earlier axiom, which can now be interpreted as a first order one:
\[ \text{old}(X, P) \rightarrow \neg \text{young}(X, P) \models \ldots \neg \text{young}(\text{harvard}, \text{AU}) \]
Combining possessive and adjective inferences

Our main axiom for possessives is now:

\[ B: \text{of}(x,y,P) \Rightarrow \text{of}(x,y,\text{hash}(P)) \land P(x) \]

This interacts with axiom A for adjectives, and we have to recursively apply these rewritings:

John’s old wooden toy broke =

\[ \exists x. \text{of}(x,john, \lambda y.\text{old}(y, \lambda z.\text{wooden}(z) \land \text{toy}(z))) \land \text{broke}(x) \]

via B:

\[ \exists x. \text{of}(x,john,\text{OWT}) \land [\lambda y.\text{old}(x,\lambda z.\text{wooden}(z) \land \text{toy}(z))](x) \land \text{broke}(x) \]

\[ \exists x. \text{of}(x,john,\text{OWT}) \land \text{old}(x,\lambda z.\text{wooden}(z) \land \text{toy}(z)) \land \text{broke}(x) \]

via A:

\[ \exists x. \text{of}(x,john,\text{OWT}) \land \text{old}(x,\text{WT}) \land [\lambda z.\text{wooden}(z) \land \text{toy}(z)](x) \land \text{broke}(x) \]

\[ \exists x. \text{of}(x,john,\text{OWT}) \land \text{old}(x,\text{WT}) \land \text{wooden}(x) \land \text{toy}(x) \land \text{broke}(x) \]
Connectives again

These schemata will still not allow us to capture inferences like

John’s wooden toy broke \(\models\) John’s toy broke

We need an axiom to the effect that if \(of(x,y,P)\) and if property P entails
property Q, then \(of(x,y,Q)\). Recall that in our combinator treatment we
had a special case of such an axiom:

\[
\forall xy PQ. of(x,y,\lambda z. P(z) \land Q(z)) \rightarrow of(x,y,P) \land of(x,y,Q) \text{ etc.}
\]

While this will generalise to multiple conjunctions, it will not handle
analogous cases with disjuncts:

John’s mother or father phoned

\(\models\) John’s mother or John’s father phoned

What we really need is to frame our axioms as schemata that are able to
quantify over more than second order.
Go even higher order?

Intuitively, what we need to do is to ‘raise’ the internal logical structure of a second-order lambda-term argument over the propositional level. Schematically:

\[ C_n: \text{of}(x,y,\lambda z. P(z) C Q(z)) \Rightarrow \text{of}(x,y,P) C \text{of}(x,y Q) \]

(‘C’ can be \& or \lor - but linguistically (I think!) we will never get \rightarrow).

Smith’s mother or father phoned

\[ \exists x.\text{of}(x,\text{smith},\lambda y. \text{mother}(y) \lor \text{father}(y)) \land \text{phoned}(x) \]

via \(C_n: \exists x. (\text{of}(x,\text{smith},\text{mother}) \lor \text{of}(x,\text{smith},\text{father})) \land \text{phoned}(x)\)

Now we can handle our earlier example:

\[ \exists x.\text{of}(x,\text{john}, \lambda z. \text{wooden}(z) \land \text{toy}(z))) \land \text{broke}(x) \]

by \(B: \exists x.\text{of}(x,\text{john}, WT) \land \text{wooden}(x) \land \text{toy}(x) \land \text{broke}(x)\)

by \(C_n\) also: \(\exists x.\text{of}(x,\text{john}, \text{wooden}) \land \text{of}(x,\text{john},\text{toy}) \land \text{broke}(x)\)

by \(B: \exists x.\text{of}(x,\text{john}, W) \land \text{of}(x,\text{john}, T) \land \text{wooden}(x) \land \text{toy}(x) \land \text{broke}(x)\)

which \(\models \) John’s toy broke = \(\exists x.\text{of}(x,\text{john}, T) \land \text{toy}(x) \land \text{broke}(x)\)
Conclusions

- There are some simple but central NL constructs that seem to need second order inference
- It may be possible to reduce to first order via reification, but I haven’t been able to
- Translation to FOL via combinators may work, but is a little clumsy and may not be possible in full generality
- A better approach seems to be to pre-process the higher order LFs using second (or higher) order matching, rewriting in a forward-chaining manner to add extra first-order LFs
- Health warning: I haven’t implemented the second (rewriting) approach yet, so there is sure to be some problem I haven’t foreseen...
References

Computing genitive superlatives.

Wide-coverage semantic representations from a CCG parser.

Hurd, J. (2002).
An LCF-style interface between HOL and first-order logic.

Genitives, relational nouns, and argument-modifier ambiguity.

Quantifiers in Language and Logic.

Taking Scope.
MIT Press.