Semantics and Pragmatics of NLP
Ambiguity and Underspecification

Alex Lascarides & Ewan Klein

School of Informatics
University of Edinburgh

31 January 2008
1 Representing Ambiguity

2 Conclusion
Operator Ambiguity

Don’t choose the fish starter or order white wine.

1. \( \neg (\text{choose-fish} \lor \text{order-white-wine}) \)
2. \( (\neg \text{choose-fish}) \lor \text{order-white-wine} \)
Operator Ambiguity

Don't choose the fish starter or order white wine.

1. \( \neg (\text{choose-fish} \lor \text{order-white-wine}) \)
2. \( (\neg \text{choose-fish}) \lor \text{order-white-wine} \)
Don't choose the fish starter or order white wine.

1. \( \neg(\text{choose-fish} \lor \text{order-white-wine}) \)
2. \( (\neg\text{choose-fish}) \lor \text{order-white-wine} \)
Quantifier Scope Ambiguity

Every man loves a woman

1. $\forall x (\text{man}(x) \rightarrow \exists y (\text{woman}(y) \land \text{love}(x, y)))$
2. $\exists y (\text{woman}(y) \land \forall x (\text{man}(x) \rightarrow \text{love}(x, y)))$

Semantic scope ambiguity, but:

- Only one syntactic form in most current grammars
- To advocate syntactic ambiguity is:
  - ad hoc
  - computationally problematic
  - inadequate with respect to pragmatics
Quantifier Scope Ambiguity

- Every man loves a woman
  - \(\forall x (\text{man}(x) \rightarrow \exists y (\text{woman}(y) \land \text{love}(x, y)))\)
  - \(\exists y (\text{woman}(y) \land \forall x (\text{man}(x) \rightarrow \text{love}(x, y)))\)

Semantic scope ambiguity, but:

- Only one syntactic form in most current grammars
- To advocate syntactic ambiguity is:
  - *ad hoc*
  - computationally problematic
  - inadequate with respect to pragmatics
Quantifier Scope Ambiguity

Every man loves a woman

1. $\forall x (\text{man}(x) \rightarrow \exists y (\text{woman}(y) \land \text{love}(x, y)))$
2. $\exists y (\text{woman}(y) \land \forall x (\text{man}(x) \rightarrow \text{love}(x, y)))$

Semantic scope ambiguity, but:

- Only one syntactic form in most current grammars
- To advocate syntactic ambiguity is:
  - ad hoc
  - computationally problematic
  - inadequate with respect to pragmatics
Quantifier Scope Ambiguity

- Every man loves a woman
  1. $\forall x (\text{man}(x) \rightarrow \exists y (\text{woman}(y) \land \text{love}(x, y)))$
  2. $\exists y (\text{woman}(y) \land \forall x (\text{man}(x) \rightarrow \text{love}(x, y)))$

Semantic scope ambiguity, but:

- Only one syntactic form in most current grammars
- To advocate syntactic ambiguity is:
  - *ad hoc*
  - computationally problematic
  - inadequate with respect to pragmatics
Build a **partial description** of the LF in the grammar:

- This is called an **underspecified semantic representation** or USR.

- Write an algorithm for working out which FOL formulas a USR describes.
  - More than one FOL formula \(\approx\) semantic ambiguity.

- That is, any FOL formula which satisfies a USR is a possible LF.
Build a partial description of the LF in the grammar:
- This is called an underspecified semantic representation or USR.

Write an algorithm for working out which FOL formulas a USR describes.
- More than one FOL formula $\approx$ semantic ambiguity.
- That is, any FOL formula which satisfies a USR is a possible LF.
Build a **partial description** of the LF in the grammar:
- This is called an **underspecified semantic representation** or USR.

Write an algorithm for working out which FOL formulas a USR describes.
- More than one FOL formula $\approx$ semantic ambiguity.
- That is, any FOL formula which satisfies a USR is a possible LF.
Build a **partial description** of the LF in the grammar:

- This is called an **underspecified semantic representation** or USR.

Write an algorithm for working out which FOL formulas a USR describes.

- More than one FOL formula \(\approx\) semantic ambiguity.

That is, any FOL formula which satisfies a USR is a possible LF.
Build a **partial description** of the LF in the grammar:

- This is called an **underspecified semantic representation** or USR.

Write an algorithm for working out which FOL formulas a USR describes.

- More than one FOL formula $\approx$ semantic ambiguity.

That is, any FOL formula which satisfies a USR is a possible LF.
The two readings again:

1. \( \neg (F \lor W) \)
2. \( (\neg F) \lor W \)

Use \( h_i \) as a variable over sub-formulas:

- \( h_1 \lor W \)
- \( \neg h_2 \)
Use $h_i$ as a variable over sub-formulas:

- $h_1 \lor W$
- $\neg h_2$

Think of $h_i$ as a ‘hole’ in the formula. Possible solutions:

1. (i) $h_1 = F$
   (ii) $h_2 = (F \lor W)$

2. (i) $h_1 = (\neg F)$
   (ii) $h_2 = F$
Back to the fish and wine example, 2

Use $h_i$ as a variable over sub-formulas:

- $h_1 \lor W$
- $\neg h_2$

Think of $h_i$ as a ‘hole’ in the formula. Possible solutions:

1

(i) $h_1 = F$
(ii) $h_2 = (F \lor W)$

2

(i) $h_1 = (\neg F)$
(ii) $h_2 = F$
Back to the fish and wine example, 2

Use $h_i$ as a variable over sub-formulas:

- $h_1 \lor W$
- $\neg h_2$

Think of $h_i$ as a ‘hole’ in the formula. Possible solutions:

1

(i) $h_1 = F$
(ii) $h_2 = (F \lor W)$

2

(i) $h_1 = (\neg F)$
(ii) $h_2 = F$
Use $l_i$ as a label over sub-formulas:

- $l_1 : \neg h_2$
- $l_2 : h_1 \lor W$
- $l_3 : F$

Possible solutions:

1. 
   (i) $h_1 = l_3$
   (ii) $h_2 = l_2$

2. 
   (i) $h_1 = l_1$
   (ii) $h_2 = l_3$
Graphical Representation of Solutions

NB $h_0$ represents ‘widest scope’.
Formulas as Trees

- Mother semantically has scope over daughters
- Left to right order \( \approx \) order of arguments to mother ‘constructor’.
Formulas as Trees

- Mother semantically has scope over daughters
- Left to right order $\approx$ order of arguments to mother 'constructor'.

### Diagram

```
        ~
       /   \
  ¬      v
   /     /  \
 F  v   F  W

        ~
       /   \
  ¬      v
   /     /  \
 F  v   F  W
```
The Strategy

Design a language which can describe these FOL trees.

- Introduce labels to refer to nodes of the tree.
  - To simplify matters, only label nodes which are roots for FOL formulas, e.g.,
  - the nodes that label $\lor$, $\neg$, etc.
- Can express information about:
  - what formula a node labels;
  - which node dominates which other nodes
    (information about relative semantic scope)
The Same Trees with the Labels

\[
\begin{align*}
  l_1 & : \neg \\
  l_2 & : \vee \\
  l_3 & : F \\
  l_4 & : W
\end{align*}
\]
Dominance Constraints

- Partial order $\leq$ between holes and labels.
  - $l_i \leq h_j$: $h_j$ has scope over $l_i$.
  - Note that $\leq$ is transitive.
    - $l_3 \leq h_1$: choose fish ($F$) is in the scope of don’t ($\neg$).
    - $l_3 \leq h_2$: choose fish ($F$) is in the scope of or ($\lor$).
    - $l_1 \leq h_0$: don’t can take widest scope.
    - $l_2 \leq h_0$: or can take widest scope.
Dominance Constraints

- Partial order $\leq$ between holes and labels.
- $l_i \leq h_j$: $h_j$ has scope over $l_i$.
- Note that $\leq$ is transitive.
  - $l_3 \leq h_1$: choose fish ($F$) is in the scope of don’t ($\neg$).
  - $l_3 \leq h_2$: choose fish ($F$) is in the scope of or ($\lor$).
  - $l_1 \leq h_0$: don’t can take widest scope.
  - $l_2 \leq h_0$: or can take widest scope.
Dominance Constraints

- Partial order \( \leq \) between holes and labels.
- \( l_i \leq h_j \): \( h_j \) has scope over \( l_i \).
- Note that \( \leq \) is transitive.
  - \( l_3 \leq h_1 \): choose fish \( (F) \) is in the scope of don’t \( (\neg) \).
  - \( l_3 \leq h_2 \): choose fish \( (F) \) is in the scope of or \( (\lor) \).
  - \( l_1 \leq h_0 \): don’t can take widest scope.
  - \( l_2 \leq h_0 \): or can take widest scope.
Dominance Constraints

- Partial order $\leq$ between holes and labels.
- $l_i \leq h_j$: $h_j$ has scope over $l_i$.
- Note that $\leq$ is transitive.
  - $l_3 \leq h_1$: choose fish ($F$) is in the scope of don’t ($\neg$).
  - $l_3 \leq h_2$: choose fish ($F$) is in the scope of or ($\vee$).
  - $l_1 \leq h_0$: don’t can take widest scope.
  - $l_2 \leq h_0$: or can take widest scope.
Dominance Constraints

- Partial order $\leq$ between holes and labels.
- $l_i \leq h_j$: $h_j$ has scope over $l_i$.
- Note that $\leq$ is transitive.
  - $l_3 \leq h_1$: choose fish ($F$) is in the scope of don’t ($\neg$).
  - $l_3 \leq h_2$: choose fish ($F$) is in the scope of or ($\lor$).
  - $l_1 \leq h_0$: don’t can take widest scope.
  - $l_2 \leq h_0$: or can take widest scope.
Dominance Constraints

- Partial order $\leq$ between holes and labels.
- $l_i \leq h_j$: $h_j$ has scope over $l_i$.
- Note that $\leq$ is transitive.
  - $l_3 \leq h_1$: choose fish ($F$) is in the scope of don’t ($\neg$).
  - $l_3 \leq h_2$: choose fish ($F$) is in the scope of or ($\lor$).
  - $l_1 \leq h_0$: don’t can take widest scope.
  - $l_2 \leq h_0$: or can take widest scope.
Dominance Constraints

- Partial order $\leq$ between holes and labels.
- $l_i \leq h_j$: $h_j$ has scope over $l_i$.
- Note that $\leq$ is transitive.
  - $l_3 \leq h_1$: choose fish ($F$) is in the scope of don’t ($\neg$).
  - $l_3 \leq h_2$: choose fish ($F$) is in the scope of or ($\lor$).
  - $l_1 \leq h_0$: don’t can take widest scope.
  - $l_2 \leq h_0$: or can take widest scope.
Dominance Constraints

\begin{align*}
l_1 &: \neg h_1 \\
l_2 &: h_2 \lor W \\
l_3 &: F
\end{align*}
Solutions and Non-solutions

\[ h_0 \]
\[ l_1 : \neg h_1 \]
\[ l_2 : h_2 \lor W \]
\[ l_3 : F \]

\[ h_0 \]
\[ l_2 : h_2 \lor W \]
\[ l_1 : \neg h_1 \]
\[ l_3 : F \]

\[ h_0 \]
\[ l_1 : \neg h_1 \]
\[ l_3 : F \]

\[ h_0 \]
\[ l_1 : \neg h_1 \]
\[ l_3 : F \]
The USR Language: Predicate Logic Unplugged (PLU)

Have internal holes \( H = \{ h_1, h_2, \ldots \} \) plus ‘top hole’ \( h_0 \)

1. Terms are constants and variables
2. An atomic FOL formula is an atomic PLU formula
3. If \( h \) is an internal hole, then \( h \) is a PLU formula.
4. If \( \phi \) and \( \psi \) are PLU formulas, then so are \( \neg \phi, \phi \rightarrow \psi, \phi \lor \psi, \phi \land \psi \).
5. If \( x \) is a variable and \( \phi \) is a PLU formula, then \( \forall x \phi \) and \( \exists x \phi \) are PLU formulas.
The USRs

A USR is a triple:

1. A set of labels and holes that are used in the USR
2. A set of labelled PLU formulas
3. A set of constraints \( l \leq h \) where \( l \) is a label and \( h \) is a hole (including \( h_0 \)).

\[
\langle \begin{array}{c}
  l_1 \\
  l_2 \\
  l_3 \\
  h_0 \\
  h_1 \\
  h_2
\end{array} , \begin{array}{c}
  l_1 : \neg h_1 \\
  l_2 : h_2 \lor \text{order-white-wine} \\
  l_3 : \text{choose-fish}
\end{array} , \begin{array}{c}
  l_1 \leq h_0 \\
  l_2 \leq h_0 \\
  l_3 \leq h_1 \\
  l_3 \leq h_2
\end{array} \rangle
\]
Reading

- Read section 3.4 of Blackburn & Bos on Hole Semantics
- For a more constrained alternative, see Copestake et al (ACL 2001) — Minimal Recursion Semantics (MRS)
Reading

- Read section 3.4 of Blackburn & Bos on Hole Semantics
- For a more constrained alternative, see Copestake et al (ACL 2001) — Minimal Recursion Semantics (MRS)
Underspecification Recapitulated

- Don’t build LFs in the grammar; build **partial descriptions** of LFs!
- Language for describing LFs
  - **Labels**: name formulas/nodes in structure
  - **Holes**: name arguments with unknown values
- Accumulate constraints in the grammar; this is a USR.
- Scoping algorithm gives all possible readings from the USR, but **not** the preferred readings.
Grammar: supplies constraints on the form of the LF.
Pragmatics: augments these constraints with more constraints.

Logic of USRs is different from the logic of LFs!

<table>
<thead>
<tr>
<th>$\phi \models_{usr} \Phi$</th>
<th>$\mathcal{M'} \models_{fol} \phi'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOL formula $\phi$ satisfies USR $\Phi$</td>
<td>$\mathcal{M'}$ satisfies the FOL formula $\phi'$</td>
</tr>
<tr>
<td>$\phi$ is a finite model</td>
<td>$\mathcal{M'}$ can be infinite</td>
</tr>
<tr>
<td>$\models_{usr}$ doesn’t know about quantifiers.</td>
<td>$\models_{fol}$ knows about quantifiers.</td>
</tr>
</tbody>
</table>

Calculating what is said is easier than checking whether it’s true.