Semantics and Pragmatics of NLP
Ambiguity and Underspecification

Alex Lascarides & Ewan Klein

School of Informatics
University of Edinburgh

31 January 2008

Operator Ambiguity

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

1. $\neg$(choose-fish $\vee$ order-white-wine)
2. ($\neg$choose-fish) $\vee$ order-white-wine

Operator Ambiguity

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

1. $\neg$(choose-fish $\vee$ order-white-wine)
2. ($\neg$choose-fish) $\vee$ order-white-wine

Operator Ambiguity

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

1. $\neg$(choose-fish $\vee$ order-white-wine)
2. ($\neg$choose-fish) $\vee$ order-white-wine

Quantifier Scope Ambiguity

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

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. $\forall x(man(x) \rightarrow \exists y(woman(y) \wedge love(x, y)))$
  2. $\exists y(woman(y) \wedge \forall x(man(x) \rightarrow love(x, y)))$

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. $\forall x(\mathrm{man}(x) \rightarrow \exists y(\mathrm{woman}(y) \wedge \mathrm{love}(x, y)))$
  2. $\exists y(\mathrm{woman}(y) \wedge \forall x(\mathrm{man}(x) \rightarrow \mathrm{love}(x, y)))$

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. $\forall x(man(x) \rightarrow \exists y(woman(y) \wedge love(x, y)))$
  2. $\exists y(woman(y) \wedge \forall x(man(x) \rightarrow love(x, y)))$

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

Underspecification

- 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.

Underspecification

- 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.

Underspecification

- 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.

Underspecification

- 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.

Underspecification

- 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.

Back to the fish and wine example, 1

The two readings again:

1. $\neg(F \vee W)$
2. $(\neg F) \vee W)$

Use h_i as a variable over sub-formulas:

- $h_1 \vee W$
- $\neg h_2$

Back to the fish and wine example, 2

Use h_i as a variable over sub-formulas:

- $h_1 \vee W$
- $\neg h_2$

Think of h_i as a 'hole' in the formula. Possible solutions:

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

- $h_1 \vee W$
- $\neg h_2$

Think of h_i as a 'hole' in the formula. Possible solutions:

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

- $h_1 \vee W$
- $\neg h_2$

Think of h_i as a 'hole' in the formula. Possible solutions:

1. (i) $h_1 = F$
   (ii) $h_2 = (F \vee W)$
2. (i) $h_1 = (\neg F)$
   (ii) $h_2 = F$

Labels and Holes

Use l_i as a label over sub-formulas:

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

Possible solutions:

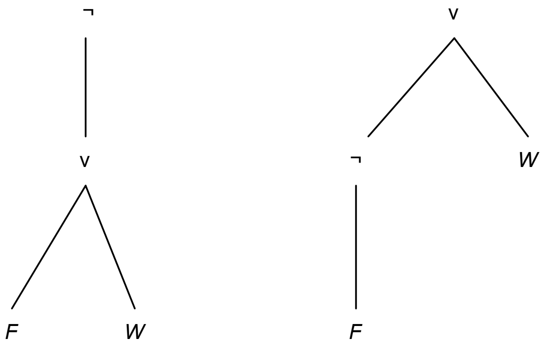
1. (i) $h_1 = l_3$
   (ii) $h_2 = l_2$
2. (i) $h_1 = l_1$
   (ii) $h_2 = l_3$

Graphical Representation of Solutions

$$\begin{tikzcd}
h_0 \arrow[d] & h_0 \arrow[d] \\
l_1: \neg h_1 \arrow[d] & l_2: h_2 \vee W \arrow[d] \\
l_2: h_2 \vee W \arrow[d] & l_1: \neg h_1 \arrow[d] \\
l_3: F & l_3: F
\end{tikzcd}$$

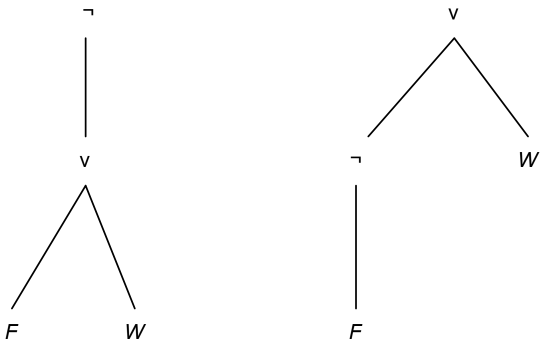
NB h_0 represents ‘widest scope’.

Formulas as Trees



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

Formulas as Trees



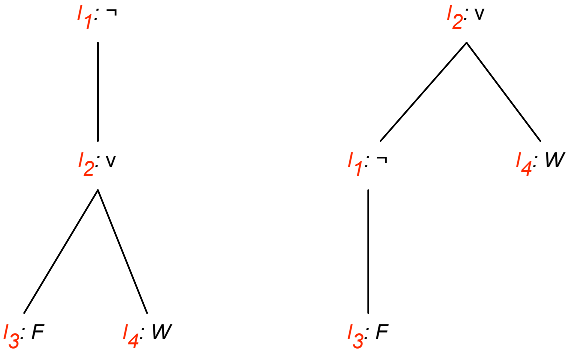
- Mother semantically has scope over daughters
- Left to right order $\approx$ order of arguments to mother 'constructor'.

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 $\vee$, $\neg$, 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



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* ($\vee$).
  - $l_1 \leq h_0$: *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* ($\vee$).
  - $l_1 \leq h_0$: *don't* can take widest scope.
  - $l_2 \leq h_0$: *or* can take widest scope.

Dominance Constraints

SPNLP: Ambiguity and Underspecification

Lascarides & Klein

- 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* ($\vee$).
  - $l_1 \leq h_0$: *don't* can take widest scope.
  - $l_2 \leq h_0$: *or* can take widest scope.

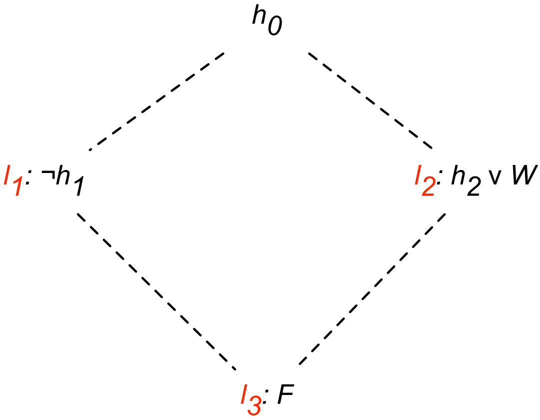
Dominance Constraints

- 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* ($\vee$).
  - $l_1 \leq h_0$: *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* ($\vee$).
  - $l_1 \leq h_0$: *don't* can take widest scope.
  - $l_2 \leq h_0$: *or* can take widest scope.

Dominance Constraints



Solutions and Non-solutions

$$h_0 \rightarrow l_1: \neg h_1 \rightarrow l_2: h_2 \vee W \rightarrow l_3: F$$

$$h_0 \rightarrow l_2: h_2 \vee W \rightarrow l_1: \neg h_1 \rightarrow l_3: F$$

$$h_0 \rightarrow l_1: \neg h_1 \rightarrow l_3: F$$

$$h_0 \rightarrow l_1: \neg h_1 \rightarrow l_1: \neg h_1 \rightarrow l_3: F$$

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 ϕ and ψ are PLU formulas, then so are $\neg\phi$, $\phi \rightarrow \psi$, $\phi \vee \psi$, $\phi \wedge \psi$.
5. If x is a variable and ϕ 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).

$$\left\langle \left\{ \begin{array}{l} l_1 \\ l_2 \\ l_3 \\ h_0 \\ h_1 \\ h_2 \end{array} \right\}, \left\{ \begin{array}{l} l_1 : \neg h_1 \\ l_2 : h_2 \vee \text{order-white-wine} \\ l_3 : \text{choose-fish} \end{array} \right\}, \left\{ \begin{array}{l} l_1 \leq h_0 \\ l_2 \leq h_0 \\ l_3 \leq h_1 \\ l_3 \leq h_2 \end{array} \right\} \right\rangle$$

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.

Architecture

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!

$\phi \models_{usr} \Phi$	$M' \models_{fol} \phi'$
FOL formula ϕ satisfies USR Φ	M' satisfies the FOL formula ϕ'
ϕ is a finite model	M' can be infinite
$\models_{usr}$ doesn't know about quantifiers.	$\models_{fol}$ knows about quantifiers.

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