Introduction to Hole Semantics

Johannes Dellert

Seminar für Sprachwissenschaft, Universität Tübingen

May 28, 2008

Underspecified Representation

- using the storage techniques we have introduced so far, we are able to enumerate all the possible semantic readings
- however, we have found out that some of these readings violate semantic constraints
- storage techniques do not allow us to express such constraints
- the solution is to keep the principle of encoding multiple readings in one structure, but to allow a much more general approach to expressing constraints on the possible readings

Why hole semantics?

- the method was devised by Johan Bos, one of the authors of our book
- it uses first-order logic to talk about first-order formulae, so a lot of the technical machinery we will need for implementing it is already at our hands
- the method is still relatively intuitive to understand

URL - Underspecified Representation Language

- the language of the semantic formulae as we know them will from now on be called **SRL - Semantic Representation Language**
- in order to be able to express constraints, we devise a first-order language to describe SRL formula trees and call it **URL - Underspecified Representation Language**
- these formulae do not fully specify one single SRL expression, but their models can contain a number of SRL formula trees

Holes and Labels

- URL formulae specify SRL subformulae in the form of formula tree fragments
- **holes** are basically the leaf nodes of tree fragments that must be filled with other tree fragments
- they could also be seen as variables ranging over subformulae
- to allow the description of full SRL formula trees, tree fragments have identifiers called **labels**
- labels are also used to glue together parts of SRL formulae that stay fixed in every interpretation
- they could also be seen as the elements in a universe that can be assigned to holes

Dominance constraints and pluggings

- the constraints we impose on possible interpretations will occur in the form of **dominance constraints**
- with such a constraint, we can express that a certain hole must be higher in the formula tree than another node
- these dominance constraints will specify how SRL expressions can be plugged together
- a **plugging** is an assignment of labels to holes, specifying which subformulae are to be plugged into which positions

An example

- "Every boxer loves a woman"
- $\exists l_1 \exists l_2 \exists v_1 (l_1{:}ALL(v_1, l_2) \wedge \exists l_3 \exists h_1 (l_2{:}IMP(l_3, h_1) \wedge l_3{:}BOXER(v_1) \wedge \exists l_4 \exists l_5 \exists v_2 (l_4{:}SOME(v_2, l_5) \wedge \exists l_6 \exists h_2 (l_5{:}AND(l_6, h_2) \wedge l_6{:}WOMAN(v_2) \wedge \exists l_7 (l_7{:}LOVE(v_1, v_2) \wedge l_7 \leq h_1 \wedge l_7 \leq h_2 \wedge \exists h_0 (l_1 \leq h_0 \wedge l_4 \leq h_0))))))$
- $\exists h_0 \exists h_1 \exists h_2 \exists l_1 \exists l_2 \exists l_3 \exists l_4 \exists l_5 \exists l_6 \exists l_7 \exists v_1 \exists v_2 (l_1{:}ALL(v_1, l_2) \wedge l_2{:}IMP(l_3, h_1) \wedge l_3{:}BOXER(v_1) \wedge l_4{:}SOME(v_2, l_5) \wedge l_5{:}AND(l_6, h_2) \wedge l_6{:}WOMAN(v_2) \wedge l_7{:}LOVE(v_1, v_2) \wedge l_7 \leq h_1 \wedge l_7 \leq h_2 \wedge l_1 \leq h_0 \wedge l_4 \leq h_0)$

Semantic Resolvers

- semantic resolvers are constraint resolvers specialized on dominance graphs
- as usual, B & B will provide a toy solution in Prolog
- why use external software for this purpose?
- as we have seen, graphical output is extremely helpful for understanding und debugging
- real-world implementations are chart-based
  $\rightarrow$ a Prolog version without dynamic programming will be quite slow for larger applications

The "Swiss Army Knife of Underspecification"

uTool

- the most efficient known solver for underspecified descriptions
- can convert between various formalisms
- developed within the CHORUS project at Saarland University (A. Koller, S. Thater, M. Regneri)
- entirely in Java, distributed under the GPL
- can be run in server mode, illustrative graphical user interface and visualisation module

The `holesem-comsem` input codec

- UTool has an input codec that explicitly processes hole semantics in the format from our book:
  ```
  some(A, some(B, some(C, some(X,
  and(label(A), and(hole(B), and(label(C),
  and(some(A, X, B), and(pred1(C,man,X),
  leq(C,B))))))))))
  ```
- however, as soon as you want to extend the coverage of the hole semantics e.g. to three-place predicates, the input codec will not support such changes any longer
- when talking with the developers about this, they strongly discouraged using this codec outside the context of the book also because of other problems
- their solution proposal: hole semantics formulae can be converted to **dominance graphs** in a relatively straightforward manner

The domcon-oz input codec

- $\exists h_0 \exists h_1 \exists h_2 \exists l_1 \exists l_2 \exists l_3 \exists l_4 \exists l_5 \exists l_6 \exists l_7 \exists v_1 \exists v_2$ $(l_1{:}ALL(v_1, l_2) \wedge l_2{:}IMP(l_3, h_1) \wedge l_3{:}BOXER(v_1) \wedge l_4{:}SOME(v_2, l_5) \wedge l_5{:}AND(l_6, h_2) \wedge l_6{:}WOMAN(v_2) \wedge l_7{:}LOVE(v_1, v_2) \wedge l_7 \leq h_1 \wedge l_7 \leq h_2 \wedge l_1 \leq h_0 \wedge l_4 \leq h_0)$
- this can (relatively) easily be converted into:
- `[label(l1 all(v11 l2)) label(l2 imp(l3 h1)) label(l3 boxer(v12)) label(l4 some(v21 l5)) label(l5 and(l6 h2)) label(l6 woman(v22)) label(l7 love(v13 v23)) dom(h1 l7) dom(h2 l7) dom(h0 l1) dom(h0 l4) label(v11 v1) label(v12 v1) label(v13 v1) label(v21 v2) label(v22 v2) label(v23 v2)]`

Demonstration

Vocabulary of an URL

- assume we have fixed the vocabulary of our SRL
- then the vocabulary of the URL for this SRL consists of the following items:
  - the 2-place predicates :NOT and $\leq$
  - the 3-place predicates
    :IMP, :AND, :OR, :ALL, :SOME and :EQ
  - a constant for every constant in the SRL vocabulary:
    if MIA is an SRL constant, then MIA is also a URL constant
  - a n+1-place relation symbol for every n-place relation symbol in the SRL vocabulary:
    if LOVE is a 2-place relation symbol in the SRL,
    then :LOVE is a 3-place relation in the URL

Sorts in URL

- URLs are first-order languages with three sorts
  - **hole** variables are written h_1, h_2, h_3 etc.
  - **label** variables are written l_1, l_2, l_3 etc.
  - **meta-variables** are written v_1, v_2, v_3 etc.
- holes and labels will be called **nodes**
- meta-variables and constants from the SRL will be called **meta-terms**
- we will not need all the sorted first-order formulae that can be constructed over the vocabulary:
  we are mainly interested in all the *existentially closed conjunctive formulae*

Basic Underspecified Representations

- A **basic USR** is defined as follows:
  - If l is a label, and h is a hole, then $l \leq h$ is a basic USR.
  - If l is a label, and n and n' are nodes, then $l{:}NOT(n)$, $l{:}IMP(n, n')$, $l{:}AND(n, n')$, and $l{:}OR(n, n')$ are basic USRs.
  - If l is a label, and t and t' are meta-terms, then $l{:}EQ(t, t')$ is a basic USR.
  - If l is a label, S is a symbol in the SRL language with arity n, and $t_1 ... t_n$ are meta-terms, then $l{:}S(t_1, ..., t_n)$ is a basic USR.
  - If l is a label, and v is a meta-variable, and n a node, then $l{:}SOME(v, n)$ and $l{:}ALL(v, n)$ are basic USRs.
  - Nothing else is a basic USR.

Underspecified Representations

- We define the remaining USRs as follows:
  - All basic USRs are USRs.
  - If ϕ is a USR, and n is a node then $\exists n\phi$ is a USR.
  - If ϕ is a USR, and v is a meta-variable then $\exists v\phi$ is a USR.
  - If ϕ and ψ are USRs, then $(\phi \wedge \psi)$ is a USR.
  - Nothing else is a USR.
- this definition exactly describes the set of all the *existentially closed conjunctive formulae* of our three-sorted URL language

Pluggings

- a **plugging** is an injective function with the set of holes as domain and the set of labels as codomain
- a plugging is **admissible** if
  - the instantiations of the holes with labels result in a representation in which there is no contradiction with anything demanded by the constraints
  - the resulting representation describes a formula tree, meaning that it must not contain cycles and that it must be connected

References

- Patrick Blackburn & Johan Bos 2003: Representation and Inference for Natural Language, section 3.4
- Johan Bos 1996 : Predicate Logic Unplugged
- A. Koller, J. Niehren, S. Thater 2004: Bridging the Gap Between Underspecification Formalisms: Hole Semantics as Dominance Constraints

Conclusion

Thank you!