Prolog and its Applications

Lecture 12: Natural Language Parsing II

Ian Pratt-Hartmann

Room: KB 2.38
email: ipratt@cs.man.ac.uk

2016-7
Outline

Deep structure versus surface structure

Specifying generation

Specifying movement
Deep structure versus surface structure

Specifying generation

Specifying movement

• The sentences we have parsed so far contain no **auxiliary verbs** *(auxiliaries)*:
  
  John loves Mary  
  Every boy kissed a girl

• But many sentences do:
  
  John **does** love Mary  
  Every boy **can** kiss some girl.

• What is the phrase-structure of sentences with auxiliaries?
• The standard analysis takes the auxiliaries to be **inflections** (I).
• Sentences containing them are said to be **inflection phrases** (IPs).

```
IP
   NP        I'
   Det    N    V P
   Every  boy  can
         V
         V'
         NP
         kiss  Det N
         some  girl
```
• But what now is the phrase-structure of sentences without auxiliaries?

• Here is the standard account:
• That is, the grammar postulates a two-stage process of sentence formation:
  • generation, producing the deep structure;
  • movement, producing the surface structure.

When no auxiliaries are present, the verb moves from its position under V to join onto the inflexion under I.

• This approach to linguistic theory is known as transformational grammar, and was developed in Noam Chomsky’s book Syntactic Structures (1957).
Outline

Deep structure versus surface structure

Specifying generation

Specifying movement
• We begin with the auxiliary case: consider

\[\text{Every boy can kiss some girl}\]

• We want to generate the (deep=表面) structure:
• Here are the basic dcs for our new grammar

• The rules for NP (cosmetic changes)

```prolog
dp(np(np(DetSyn,NbarSyn)) -->
    det(DetSyn), nbar(NbarSyn).
dbar(nbar(NSyn)) -->
    n(NSyn).
n(n(Noun)) -->
    [Noun], {isNoun(Noun)}.

det(det(Det)) -->
    [Det], {isDet(Det)}. ```
• The rules for VP (cosmetic changes)

\[
\begin{align*}
vp(vp(VbarSyn)) & \rightarrow \\
vbar(VbarSyn). & \\
vbar(vbar(VSyn,NPSyn)) & \rightarrow \\
v(VSyn), np(NPSyn). & \\
v(v(Verb)) & \rightarrow \\
[Verb], \{isVerb(Verb)\}. & 
\end{align*}
\]
• The lexicon (as before)
  isNoun(boy).
  isNoun(girl).
  isVerb(kiss).
  isDet(some).
  isDet(every).
  isAux(can).
The rules for IP (new):

\[
\text{ip}(\text{Sentence}) :\neg \\
\quad \text{setof}(\text{IPSem}, \\
\quad \quad \text{ip}(\text{IPSem}, \text{Sentence}, []), \\
\quad \quad \text{IPSems}), \\
\quad \text{treeP}(\text{IPSems}).
\]

\[
\text{ip}(\text{ip}(\text{NPSyn}, \text{VPSyn})) \rightarrow \\
\quad \text{np}(\text{NPSyn}), \text{ibar}(\text{VPSyn}).
\]

\[
\text{ibar}(\text{ibar}(\text{ISyn}, \text{VPSyn})) \rightarrow \\
\quad \text{i}(\text{ISyn}), \text{vp}(\text{VPSyn}).
\]

\[
\text{i}(\text{i}(\text{Aux})) \rightarrow [\text{Aux}], \{\text{isAux}(\text{Aux})\}.
\]
• When we run the program, we get
  
  \[- \text{ip([every, boy, can, kiss, some, girl])}.
  \text{[ip(np(det(every))}
          \nbar(n(boy)))
  \text{ibar(i(can)}
          \vp(vbar(v(kiss))
          \text{np(det(some)}
          \nbar(n(girl))))]]

• which is just the tree

```
          IP
             / \            /  \
            NP   I'        VP   NP
            / \            /   /  \
           Det N    I    V  Det N
           |    |    |   |    |   |   |
         Every boy can kiss some girl
```
• Interlude: note the nice calling function

    ip(Sentence):-
        setof(IPSem,
            ip(IPSem,Sentence,[]),IPSems),
        treeP(IPSems).

• The predicate treeP/1 is worth taking a look at .... See the auxilliary file lab3_aux.pl.
Deep structure versus surface structure

Specifying generation

Specifying movement
• Movement is easy to specify in dcg’s
• To keep things simple, suppose we are only interested in past-tense sentences
• Rules for NPs are as before
• We assume a predicate which deals with verb inflections:

?– pastInfl(kiss, PastTense).

PastTense = kissed ;

No

?= pastInfl(Root, kissed).

Root = kiss ;

No
New rules for IP and VP are:

\[
\text{ip}(\text{ip}(\text{NPSyn}, \text{VPSyn})) \rightarrow \text{np}(\text{NPSyn}), \text{ibar}(\text{VPSyn}).
\]

\[
\text{ibar}(\text{ibar}(\text{ISyn}, \text{VPSyn})) \rightarrow
\]

\[
\text{i}(\text{ISyn}, \text{MvdVbL}), \text{vp}(\text{VPSyn}, \text{MvdVbL}).
\]

\[
\text{i}(\text{i}(\text{Aux}), []) \rightarrow [\text{Aux}], \{\text{isAux}(\text{Aux})\}.
\]

\[
\text{i}(\text{i}(\text{pastInfl}), [\text{Verb}]) \rightarrow
\]

\[
[\text{InflVerb}], \{\text{pastInfl}(\text{Verb}, \text{InflVerb}), \text{isVerb}(\text{Verb})\}.
\]

\[
\text{vp}(\text{vp}(\text{VbarSyn}, \text{MvdVbL})) \rightarrow \text{vbar}(\text{VbarSyn}, \text{MvdVbL}).
\]

\[
\text{vbar}(\text{vbar}(\text{VSyn}, \text{NPSyn}), \text{MvdVbL}) \rightarrow
\]

\[
\text{v}(\text{VSyn}, \text{MvdVbL}), \text{np}(\text{NPSyn}).
\]

\[
\text{v}(\text{v}(\text{Verb}), []) \rightarrow [\text{Verb}], \{\text{isVerb}(\text{Verb})\}.
\]

\[
\text{v}(\text{v}(\text{MvdVb}), [\text{MvdVb}]) \rightarrow [].
\]

The rules for np are unaffected.
Here’s what is happening pictorially in the auxiliary (no-movement) case:

\[
\text{MvdVL} = []
\]
Here’s what is happening pictorially in the no-auxiliary case, after movement:

\[ \text{MvdVL} = [\text{kiss}] \]

\[ \text{MvdVb} = \text{kiss} \]
Summary:

- Deep structure and surface structure in transformational grammar.
- Implementation in Prolog.

What should I do next?

- Revise everything you have read in Learn Prolog Now!
- Start the third (final) programming exercise.