Semantics Interpretation

Allen’s Chapter 9
J&M’s Chapter 15
Rule-by-rule semantic interpretation

- Computing Logical forms (i.e., Semantic Interpretation)
- Generating a Syntactic tree from a logical form (i.e., Semantic Realization)
- The meaning of all constituent must be determined
- Using extra features for representing semantics
- Rule-by-Rule semantic interpretation
Semantic interpretation and compositionality

- Semantic interpretation is compositional (similar to parsing)
- The meaning of a constituent is derived from the meaning of its sub constituents
- Interpretations can be built incrementally from the interpretations of sub phrases
- Compositional models make grammars easier to extend and maintain
Semantic interpretation and compositionality

• It is harder than it may seem

• Syntactical structures ≠ Semantic structures

• Syntactical structure of *Jill loves every dog* is: 
  \(((\text{Jill}) \ (\text{loves} \ (\text{every} \ \text{dog})))\)

• Its unambiguous logical form is:
  
  \[(\text{EVERY} \ d1 : (\text{DOG1} \ d1)) \ (\text{LOVES1} \ l1 \ (\text{NAME} \ j1 \ \text{“Jill”}) \ d1)\]
Difficulties of Semantic interpretation via compositionality

• In syntax: *every dog* is a sub constituent of VP,

• In the logical form, the situation is reversed

• How can *every dog* be represented in isolation?

• Using quasi-logical forms is one way around this problem

  (LOVES1 l1 (NAME j1 “Jill”) <EVERY d1 DOG1>)
Problem with Idioms

- Another obstacle for the compositionality assumption is the presence of idioms
  
  *Jack kicked the bucket = Jack died*

- Solution 1: semantic meaning to be assigned to the entire phrase

- What about: *The bucket was kicked by Jack?*

- *Jack kicked the bucket* is ambiguous between:
  - (KICK1 k1 (NAME j1 “Jack”) <THE b1 BUCKET1>)
  - (DIE1 d1 (NAME j1 “Jack”))

- Solution 2: adding a new sense of *die* for the verb *kick* with sub categorization for an object BUCKET1
Semantic interpretation and compositionality

- We should be able to assign a semantic structure to any syntactic constituent
- For instance, assigning a uniform form of meaning to every verb phrase in any rule involving a VP
- The meaning of the VP in *Jack laughed* can be shown by a unary predicate:
  \[(\text{LAUGHED}1 \ 1 \ (\text{NAME} \ j1 \ “Jack”))\]
- The VP in, *Jack loved Sue*, should also be represented by a unary predicate:
  \[(\text{LOVES}1 \ 12 \ (\text{NAME} \ j1 \ “Jack”) \ (\text{NAME} \ s1 \ “Sue”))\]
- **Lambda calculus** provides a formalism for representing such predicates
Lambda calculus

• Using lambda calculus, loved sue is represented as:

\((\lambda x \ (\text{LOVES} \ 1 \ l \ x \ (\text{NAME} \ s \ \text{"SUE"})))\)

• We can apply a lambda expression to an argument, by a process called **lambda reduction**

\((\lambda x \ (\text{LOVES} \ 1 \ l \ x \ (\text{NAME} \ s \ \text{"SUE"}))) (\text{NAME} \ j \ \text{"Jack"})\)

\(= (\text{LOVES} \ 1 \ l \ (\text{NAME} \ j \ \text{"Jack"}) \ (\text{NAME} \ s \ \text{"Sue"}))\)
Lambda calculus

- Lambda-calculus provides a handy tool to couple syntax and semantics
- Verb phrases with even different structures can easily be conjoined

*Sue laughs and opens the door*

\[ (\lambda \ a \ (\text{LAUGHES}1 \ \text{l1} \ a)) \text{ and } (\lambda \ a \ (\text{OPENS}1 \ \text{o1} \ a) \ <\text{THE} \ d1 \ \text{DOOR1}>)) \text{ can be conjoined into:} \\
(\lambda \ a \ (& \ (\text{LAUGHES}1 \ \text{l1} \ a) \ (\text{OPENS}1 \ \text{o1} \ a) \ <\text{THE} \ d1 \ \text{DOOR1}>)\) \\
Applying it to (NAME s1 “Sue”) would produce the logical form:

\[ (& \ (\text{LAUGHES}1 \ \text{l1} \ (\text{NAME} \ s1 \ “Sue”)) \ (\text{OPENS}1 \ \text{o1} \ (\text{NAME} \ s1 \ “Sue”)) \ <\text{THE} \ d1 \ \text{DOOR1}>)\]
Lambda calculus (Cont.)

- Propositional phrase modifiers in noun phrases could be handled in different ways
  1. Search for location of modifiers in noun phrases and incorporate them into the interpretations
    - Works for *the man in the store*, but not for *the man is in the store* or *the man was thought to be in the store*
  2. Give an independent meaning to prepositional phrases
    - *in the store* is represented by a unary predicate
      \[(\lambda o \ (\text{IN-LOC1}\ o \ <\text{THE}\ s1\ \text{STORE1}>)\)\]
    - *The man in the store*
      \[(<\text{THE}\ m1\ (\&\ (\text{MAN1}\ m1)\ (\text{IN-LOC1}\ m1\ <\text{THE}\ s1\ \text{STORE1}>)>)\)]
    - *The man is in the store*
      \[\text{IN-LOC1} <\text{THE}\ m1\ \text{MAN1}> <\text{THE}\ s1\ \text{STORE1}>\)
Compositional approach to semantics

- In general, each major syntactic phrase corresponds to a particular semantic construction:
  - VPs and PPs map to unary predicates,
  - Sentences map to propositions,
  - NPs map to terms, and
  - minor categories are used in building major categories
A simple grammar and lexicon for semantic interpretation

- Logical forms can be computed during parsing
- The main extension needed is a SEM feature, which is added to each lexical entry and rule
- Example:
  
  \[(S \text{ SEM } (?\text{semvp } ?\text{semnp})) \rightarrow (NP \text{ SEM } ?\text{semnp}) (VP \text{ SEM } ?\text{semvp})\]

  - NP with SEM (NAME m_1 “Mary”)
  - VP with SEM (\(\lambda\ a (\text{SEES1} e_8 a (\text{NAME} j_1 “Jack”)))

- Then the SEM feature of S is:
  
  \[((\lambda\ a (\text{SEES1} e_8 a (\text{NAME} j_1 “Jack”))) (\text{NAME} m_1 “Mary”)), \text{is reduced to}\]
  
  \[(\text{SEES1} e_8 (\text{NAME} m_1 “Mary”) (\text{NAME} j_1 “Jack”)}\]
Mary sees Jack

S SEM (SEES1 e8 (NAME m1 "Mary") (NAME j1 "Jack"))

NP SEM (NAME m1 "Mary")  VP SEM (λ a (SEES1 e8 a NAME j1 "Jack"))

V SEM SEES1  NP SEM (NAME m1 "Jack")

Figure 9.1 A parse tree showing the SEM features
Sample lexicon with SEM features

a (art AGR 3s SEM INDEF1)
can (aux SUBCAT base SEM CAN1)
car (n SEM CAR1 AGR 3s)
cry (v SEM CRY1 VFORM base SUBCAT _none)
decide (v SEM DECIDES1 VFORM base SUBCAT _none)
decide (v SEM DECIDES-ON1 VFORM base SUBCAT _pp:on)
dog (n SEM DOG1 AGR 3s)
fish (n SEM FISH1 AGR 3s)
fish (n SEM (PLUR FISH1) AGR 3p)
house (n SEM HOUSE1 AGR 3s)
has (aux VFORM pres AGR 3s SUBCAT pastprt SEM PERF)
he (pro SEM HE1 AGR 3s)
in (p PFORM {LOC MOT} SEM IN-LOC1)
Jill (name AGR 3s SEM “Jill”)
man (n SEM MAN1 AGR 3s)
men (n SEM (PLUR MAN1) AGR 3p)
on (p PFORM LOC SEM ON-LOC1)
saw (v SEM SEES1 VFORM past SUBCAT _np AGR ?a)
see (v SEM SEES1 VFORM base SUBCAT _np IRREG-PAST + EN-PASTPRT +)
she (pro AGR 3s SEM SHE1)
the (art SEM THE AGR {3s 3p})
to (to AGR – VFORM inf)

Figure 9.2  A small lexicon showing the SEM features
A sample grammar with SEM and VAR features

1. \((S \ SEM \ (?semvp \ ?semnp)) \rightarrow \ (NP \ SEM \ ?semnp) \ (VP \ SEM \ ?semvp)\)
2. \((VP \ VAR \ ?v \ SEM \ (\lambda \ a2 \ (?semv \ ?v \ a2))) \rightarrow \ (V[_\ none] \ SEM \ ?semv)\)
3. \((VP \ VAR \ ?v \ SEM \ (\lambda \ a3 \ (?semv \ ?v \ a3 \ ?semnp))) \rightarrow \ (V[_\ np] \ SEM \ ?semv) \ (NP \ SEM \ ?semnp)\)
4. \((NP \ WH-\ VAR \ ?v \ SEM \ (PRO \ ?v \ ?sempro)) \rightarrow \ (PRO \ SEM \ ?sempro)\)
5. \((NP \ VAR \ ?v \ SEM \ (NAME \ ?v \ ?semname)) \rightarrow \ (NAME \ SEM \ ?semname)\)
6. \((NP \ VAR \ ?v \ SEM \ <\semart \ ?v \ (?semcnp \ ?v)> \rightarrow \ (ART \ SEM \ ?semart) \ (CNP \ SEM \ ?semcnp)\)
7. \((CNP \ SEM \ ?semn) \rightarrow \ (N \ SEM \ ?semn)\)

Head features for S, VP, NP, CNP: VAR

Grammar 9.3  A simple grammar with SEM features

- A feature called VAR is also needed to store the discourse variable that corresponds to the constituents
- The VAR feature is automatically generated when a lexical constituent is constructed from a word
- Then the VAR feature is passed up the parse tree by being treated as a head feature
- This guarantees that the discourse variables are always unique
- If ?v is m1, SEM of the man is: \(<\text{THE m1 (MAN1 m1)}>\)
Using Chart Parsing for semantic interpretation

• Only two simple modifications are needed to use the standard Chart Parser to handle semantic interpretation:
  • When a lexical rule is instantiated, the VAR feature is set to a new discourse variable
  • Whenever a constituent is built, its SEM is simplified by possible lambda reductions
Jill saw the dog

Figure 9.5 The parse of Jill saw the dog showing the SEM and VAR features
Handling Prepositional Phrases

• Propositional phrases play two different semantic roles:
  
  1. PP can be a modifier to a noun phrase or a verb phrase (cry in the corner)
  
  2. PP may be needed by a head word, and the preposition acts more as a term than as an independent predicate (Jill decided on a couch)
PP as a modifier of a noun phrase

• The appropriate rule for interpreting PP is:
  (PP SEM (λ y (?semp y ?semnp)))) → (P SEM ?semp) (NP SEM ?semnp)
• if SEM of P is: IN-LOC1, and SEM of NP is: <THE c1 (CORNER1 c1)>,
• Then the SEM of PP in the corner will be: (λ y (IN-LOC1 y <THE c1 CORNER1>)),

PP as a modifier of a noun phrase

• If the PP is used in the rule:
  \[(\text{CNP SEM } (\lambda n (\& (?\text{semcnp } n) (?\text{sempp } n)))) \rightarrow (\text{CNP SEM } ?\text{semcnp}) (\text{PP SEM } ?\text{sempp})\]

• Then SEM of CNP *man in the corner* will be:
  \[(\lambda n (\& (\text{MAN}1 n) ((\lambda y (\text{IN-LOC}1 y <\text{THE c1 CORNER1}>)) n)))\]

• That is reduced to:
  \[(\lambda n (\& (\text{MAN}1 n) (\text{IN-LOC}1 n <\text{THE c1 CORNER1}>)))\]

• Then using rule
  \[(\text{NP VAR } ?v \text{ SEM } <?\text{semart } ?v (> (?\text{semcnp } ?v)>)) \rightarrow (\text{ART SEM } ?\text{semart}) (\text{CNP SEM } ?\text{semcnp})\]

• The SEM of *the man in the corner* will be:
  \[<\text{THE } m1 ((\lambda y (\& (\text{MAN}1 y) (\text{IN-LOC}1 y <\text{THE c1 CORNER1}>)))) m1>\]

• Which is reduced to:
  \[<\text{THE } m1 (\& (\text{MAN}1 m1) (\text{IN-LOC}1 m1 <\text{THE c1 CORNER1}>))>\]
PP as a modifier of a verb phrase

• Appropriate rule is:
  
  \[
  (\text{VP VAR } ?v \ \text{SEM} \ (\lambda \ x \ (\& \ (?\text{semvp} \ x) \ (?\text{sempp} \ ?v)))) \to (\text{VP VAR } ?v \ \text{SEM} \ ?\text{semvp}) \ (\text{PP SEM} \ ?\text{sempp})
  \]

• SEM of PP \textit{in the corner} is:
  \[
  (\lambda \ y \ (\text{IN-LOC} \ y \ <\text{THE c1 CORNER1}>)),
  \]

• SEM of VP of \textit{cry} is:
  \[
  (\lambda x \ (\text{CRIES} \ e1 \ x)),
  \]

• SEM of VP of \textit{cry in the corner} will be:
  \[
  (\lambda a \ (\& \ (\text{CRIES} \ e1 \ a) \ (\text{IN-LOC} \ e1 \ <\text{THE c1 CORNER1}>)))
  \]
Parse tree of *Cry in the corner*

\[
\begin{align*}
VP & \text{SEM} \left( \lambda x \ (\& \ (\text{CRIES} \ e_1 \ x)) \ (\text{IN-LOC}1 \ e_1 \ <\text{THE} \ c_1 \ (\text{CORNER} \ c_1)>) \right) \\
\text{VAR} & \ e_1 \\
\end{align*}
\]

\[
\begin{align*}
V & \text{SEM} \left( \lambda x \ (\text{CRIES}1 \ e_1 \ x)) \right) \\
\text{VAR} & \ e_1 \\
\end{align*}
\]

\[
\begin{align*}
VP & \text{SEM} \left( \lambda y \ (\text{IN-LOC}1 \ y \ <\text{THE} \ c_1 \ (\text{CORNER} \ c_1)>) \right) \\
\text{NP} & \text{SEM} \ <\text{THE} \ c_1 \ \text{CORNER1} \ c_1 > \ \\
\text{VAR} & \ c_1 \\
\end{align*}
\]

\[
\begin{align*}
\text{P} & \text{SEM} \ \\
\text{ART} & \text{SEM} \ \text{THE} \\
\text{N} & \text{SEM} \ \text{CORNER1} \\
\text{VAR} & \ c_1 \\
\end{align*}
\]

\[
\begin{align*}
\text{cry} & \\
in & \\
\text{the} & \\
corner & \\
\end{align*}
\]

Figure 9.6 Using the VAR feature for PP modifiers of VPs
**PP as a sub constituent of a head word**

- Jill decided on a couch is ambiguous:
  1. Jill made a decision while she was on a couch
  2. Jill made a decision about a couch
- The rule for the second one is:
  - VP $\rightarrow$ V[pp:on] NP PP[on]
    - The desired logical form for this is
      \[ (\lambda s (\text{DECIDES-ON1} d1 s <A c1 (\text{COUCH1} c1)>)) \]
- Here the preposition “on” is only a term
- A binary feature PRED is needed (i.e., a predicate (+) or a term(-))
Rules for handling PPs

8. \((\text{PP PRED } + \text{ SEM }) (\lambda x (?\text{ semp } x ?\text{ semnp})) \rightarrow (P \text{ SEM } ?\text{ semp}) (\text{NP SEM } ?\text{ semnp})\)

9. \((\text{PP PRED } - \text{ PFORM } ?\text{ pf } \text{ SEM } ?\text{ semnp}) \rightarrow (P \text{ ROOT } ?\text{ pf}) (\text{NP SEM } ?\text{ semnp})\)

10. \((\text{VP VAR } ?\text{ v } \text{ SEM } (\lambda \text{ ag} 1 (& (?\text{ semvp ag} 1) (?\text{ sempp } ?\text{ v}))) \rightarrow (VP \text{ SEM } ?\text{ semvp}) (\text{PP PRED } + \text{ SEM } ?\text{ sempp})\)

11. \((\text{VP VAR } ?\text{ v } \text{ SEM } (\lambda \text{ ag} 2 (?\text{ semv } ?\text{ v } \text{ ag} 2 ?\text{ sempp})) \rightarrow (V[\_np_{pp}:\text{on SEM } ?\text{ semv}) (\text{PP PRED } - \text{ PFORM on SEM } ?\text{ sempp})\)

12. \((\text{VP SEM } (\lambda \text{ a} 1 (?\text{ semaux } (?\text{ semvp a} 1))) \rightarrow (\text{AUX SUBCAT } ?\text{ v } \text{ SEM } ?\text{ semaux}) (\text{VP VFORM } ?\text{ v } \text{ SEM } ?\text{ semvp})\)

13. \((\text{CNP SEM } (\lambda \text{ n} 1 (& (?\text{ semcnp n} 1) (?\text{ sempp n} 1))) \rightarrow (\text{CNP SEM } ?\text{ semcnp}) (\text{PP PRED } + \text{ SEM } ?\text{ sempp})\)

Head features for PP: PFORM
Head features for VP, CNP: VAR

Grammar 9.7  Rules to handle PPs in verb phrases
Different parse trees for the VP decide on the coach

**Figure 9.8** Two possible parse trees for the VP *decide on a couch*
### Handling simple questions

<table>
<thead>
<tr>
<th>Rule</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.</td>
<td>((S \text{ INV} - \text{ SEM} \ (\text{WH-query } ?\text{sems})) \rightarrow)</td>
</tr>
<tr>
<td></td>
<td>((\text{NP WH Q AGR } ?a \text{ SEM } ?\text{semp}))</td>
</tr>
<tr>
<td></td>
<td>((S \text{ INV} + \text{ SEM } ?\text{sems GAP} \ (\text{NP AGR } ?a \text{ SEM } ?\text{semp})))</td>
</tr>
<tr>
<td>15.</td>
<td>((S \text{ INV} + \text{ GAP } ?g \text{ SEM} \ (?\text{semaux } (?\text{semvp } ?\text{semp})))) \rightarrow)</td>
</tr>
<tr>
<td></td>
<td>((\text{AUX AGR } ?a \text{ SUBCAT } ?s \text{ SEM } ?\text{semaux}))</td>
</tr>
<tr>
<td></td>
<td>((\text{NP AGR } ?a \text{ GAP } - \text{ SEM } ?\text{sempp}))</td>
</tr>
<tr>
<td></td>
<td>((\text{VP VFORM } ?s \text{ GAP } ?g \text{ SEM } ?\text{semvp}))</td>
</tr>
<tr>
<td>16.</td>
<td>((\text{NP WH Q VAR } ?v \text{ SEM } &lt;\text{WH } ?v (\text{?sempro } ?v)&gt;)) \rightarrow)</td>
</tr>
<tr>
<td></td>
<td>((\text{PRO WH Q SEM } ?\text{sempro}))</td>
</tr>
</tbody>
</table>

**Grammar 9.10** Rules to handle simple wh-questions

- The lexical entry for the wh-words are extended with the SEM feature:
  - who: \((\text{PRO WH } \{Q R\} \text{ SEM WHO1 AGR } \{3s 3p\})\)
- But, how SEM and GAP features interact?
Who did Jill see?

Figure 9.11 The parse tree for *Who did Jill see?*
Prepositional phrase Wh-Questions

- Questions can begin with prepositional phrases:
  - In which box did you put the book?
  - Where did Jill go?
- The semantic interpretation of these questions depend on the type of PPs

(S INV – SEM (WH-query ?sems)) →
(S INV + SEM ?sems GAP (PP PRED ?p PFORM ?pf SEM ?sempp))
Prepositional phrase Wh-Questions

• To handle wh-terms like *where* the following rule is also needed:

\[(\text{PP PRED } ?p \text{ PFORM } ?pf \text{ SEM } ?sempp) \rightarrow (\text{PP-WRD PRED } ?p \text{ PFORM } ?pf \text{ SEM } ?sempp)\]

• There would also be two lexical entries:

\[(\text{PP-WRD PFORM } \{\text{LOC MOT}\} \text{ PRED } \text{– VAR } ?v \text{ SEM } <\text{WH } ?v \text{ (LOC1 } ?v)>\)]

\[(\text{PP PRED } \text{+ VAR } ?v \text{ SEM } (\lambda x (\text{AT-LOC } x <\text{WH } ?v \text{ (LOC1 } ?v)>)))\]
Where did Jill go?

But, Wh-questions starting with a PP with + PRED cannot be handled, yet

The difficulty comes from the restriction that in rule VP → VP PP, the GAP is passed only into the VP sub constituent

Thus there is still no way to create a PP gap that modifies a verb phrase
Relative Clauses

- Relative clauses and wh-questions are similar at the semantic level, too.
- The logical form of a relative clause is a unary predicate:

\[
(CNP \ SEM (\lambda \ ?x (\& (\semcnp \ ?x) (\semrel \ ?x)))) \rightarrow \\
(CNP \ SEM \ ?\semcnp) (REL \ SEM \ ?\semrel)
\]
Who Jill saw

- The wh-term variable is specified in a feature called RVAR in the rule

(REL SEM (\(\lambda \ ?v \ ?\text{sems}\))) →
(NP WH R RVAR ?v AGR ?a SEM ?semnp)
Semantic Interpretation by Features Unification versus Lambda reductions

- Semantic interpretation can be performed by just using feature values and variables
- Using new features instead of lambda reductions

1. \((S \text{ VFORM past SEM } ?\text{semvp}) \rightarrow \text{(NP SEM } ?\text{semnp}) \ (VP \text{TNS past SUBJ } ?\text{semnp SEM } ?\text{semvp})\)
2. \((VP \text{TNS } ?\text{tns SUBJ } ?\text{semssubj SEM } ?\text{semv}) \rightarrow \text{(V[_none] VAR } \text{?v SUBJ } ?\text{semssubj TNS } ?\text{tns SEM } ?\text{semv})\)
3. \((VP \text{TNS } ?\text{tns SUBJ } ?\text{subj SEM } ?\text{semv}) \rightarrow \text{(V[_np] VAR } \text{?v SUBJ } ?\text{subj OBJVAL } \text{?obj TNS } ?\text{tns SEM } ?\text{semv})\)
   \text{(NP SEM } ?\text{obj})\)
Jill saw the dog

Figure 9.13  The parse tree for Jill saw the dog using the SUBJ feature
Features versus Lambda expressions

1. \((S \ SEM \ ?semvp) \rightarrow\)
   \((NP \ SEM \ ?semsubj) \ (VP \ SUBJ \ ?semsubj \ SEM \ ?semvp)\)

2. \((VP \ VAR \ ?v \ SUBJ \ ?semsubj \ SEM \ (?semv \ ?v \ ?semsubj)) \rightarrow\)
   \((V[\_\_none] \ SEM \ ?semv)\)

3. \((VP \ VAR \ ?v \ SUBJ \ ?semsubj \ SEM \ (?semv \ ?v \ ?semsubj \ ?semnp)) \rightarrow\)
   \((V[\_np] \ SEM \ ?semv) \ (NP \ SEM \ ?semnp)\)

4. \((NP \ VAR \ ?v \ SEM \ (PRO \ ?v \ ?sempro)) \rightarrow \ (PRO \ SEM \ ?sempro)\)

5. \((NP \ VAR \ ?v \ SEM \ (NAME \ ?v \ ?semname)) \rightarrow \ (NAME \ SEM \ ?semname)\)

6. \((NP \ VAR \ ?v \ SEM \ (<\semart \ ?v \ ?semcnp>) \rightarrow\)
   \((ART \ SEM \ ?semart) \ (CNP \ SEM \ ?semcnp)\)

7. \((CNP \ VAR \ ?v \ SEM \ (?semn \ ?v)) \rightarrow \ (N \SEM \ ?semn)\)

Head features for S, VP, NP, CNP: VAR

Grammar 9.14  A simple grammar with SEM features
Features versus Lambda expressions

- No special mechanism (e.g., Lambda Reduction) is needed for semantic interpretation
- Grammar is reversible (can be used to generate sentences)
- Not all lambda expressions can be eliminated using this technique
- In conjoined subject phrases (e.g., Sue and Sam saw Jack), SUBJ variable need to unify with both Sue and Sam (is not possible)
Generating sentences from Logical Forms

- Intuitively, it should be easy to reverse a grammar and use it for generation:
- Decompose the logical form of each constituent into a series of lexical constituents
- But, not all grammars are reversible: e.g., a grammar with Lambda reduction
Generating sentences from Logical Forms

• Consider (<PAST SEES1> s1 (NAME j1 “Jill”) <THE d1 (DOG1 d1)>)

• Rule S with SEM (?semvp ?semnp) cannot be unified

• The problem: lambda reduction was used to convert the original logical form:

  ((λa (<PAST SEES1> d1 a <THE d1 (DOG1 d1)>))
   (NAME j1 “Jill”))

• Lambda abstraction can be used, but there are three possible lambda abstractions:

  – (λe (<PAST SEES1> e (NAME j1 “Jill”) <THE d1 (DOG1 d1)>)
  – (λa (<PAST SEES1> s1 a <THE d1 (DOG1 d1)>))
  – (λo (<PAST SEES1> s1(NAME j1 “Jill”) o))
Realization versus Parsing

• Parsing and realization both can be viewed as building a syntactic tree

• A parser starts with the words and tries to find a parse tree

• A realizer starts with a logical form and tries to find a tree and determine the words to realize it
Realization and Parsing

• Standard top-down parsing algorithm is extremely inefficient
  Consider \(<\text{PAST SEES1} \ s1 \ (\text{NAME} \ j1 \ “Jill”) \ <\text{THE} \ d1 \ (\text{DOG1} \ d1)>)\)
  \(\text{(NP SEM } ?\text{semsubj})\)
  \(\ (\text{VP SUBJ } ?\text{semsubj})
  \quad \text{SEM } (\text{<PAST SEES1} \ s1 \ (\text{NAME} \ j1 \ “Jill”) \ <\text{THE} \ d1 \ (\text{DOG1} \ d1)>)\)

• The problem is that the SEM of NP is unconstrained

• Solution: expand the constituents in a different order
Head Driven Realization algorithm

Initialization: Set L to a list containing the constituent that you wish to generate.

Do until L contains no nonlexical constituents:

1. If L contains a constituent C that is marked as a nonlexical head,
2. Then use a rule in the grammar to rewrite C. Any variables in C that are bound in the rewrite should be instantiated throughout the entire list.
3. Else choose a nonlexical constituent C, giving preference to one whose SEM feature is bound, if one exists. Use a rule in the grammar to rewrite C. Any variables in C that are bound in the rewrite should be instantiated throughout the entire list.

Figure 9.15  A head-driven realization algorithm
Head driven Realization

• Using rule 3 of grammar 9.14
  
  \[ ?semv = \langle \text{PAST SEES1} \rangle, \]
  \[ ?v = s1, \]
  \[ ?semsubj = (\text{NAME j1 "Jill"}), \]
  \[ ?semnp = \langle \text{THE} d1 \langle \text{DOG1} d1 \rangle \rangle \]

• After rewriting the VP, the following list of constituents is obtained:

  (NP SEM (NAME j1 "Jill"))
  (V [np] SEM \langle \text{PAST SEES1} \rangle)
  (NP SEM \langle \text{THE} d1 \langle \text{DOG1} d1 \rangle \rangle)
Head driven Realization

- Since there is no non-lexical head, the algorithm picks any non-lexical constituent with a bound SEM (e.g., the first NP)
- Using rule 5, it yields to (NAME SEM “Jill”)
- Selecting the remaining NP, and using rules 6, and then 7, produce the following list:
  (NAME SEM “Jill”), (V [_np] SEM <PAST SEES1>),
  (ART SEM THE), (N SEM DOG1)
- Now it is easy to produce Jill saw the dog