Slot Grammar

A specific type of Dependency Grammar

Kurt Eberle

October 26 2017
Outline

Slot Grammar

- Background
- Design
- Literature
- Dependency Trees
- Two Level Analysis
- Realization
  - Data Structures and Parsing
  - Lexicon
  - Grammar
Outline

Slot Grammar
  Background
  Design
  Literature
  Dependency Trees
  Two Level Analysis
  Realization
    Data Structures and Parsing
    Lexicon
    Grammar
A version of Dependency Grammar
Michael McCord
from the eighties on
IBM Thomas J. Watson Research Center
IBM research centers Europe
Spinoff companies: Synthema (IT), linguatec, lingenio (DE)
projects:
Logic based Machine Translation (LMT),
Deep analysis in Watson question answering system
Grammars for many languages... ESG, GSG, FSG, ... Italian, Spanish, Danish, Hebrew, Arabian, ...
Design

- Design closely related to implementation issues
- Implementation in Prolog
- ...and later in C
Design

- Own set of fine-grained dependency relations classified into
  - subcategorized argument slots
    - subj, obj, auxcomp, nobj, ...
  - free modifier slots
    - vadv, vsubconj, ndet, nadj, ...
- lexicalized theory
  - passivization rule, ...
- Exptraposition handling

...
## Literature

<table>
<thead>
<tr>
<th>Year</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1980</td>
<td>Slot Grammars</td>
</tr>
<tr>
<td></td>
<td>Computational Linguistics, vol. 6, pp. 31-43.</td>
</tr>
<tr>
<td>1982</td>
<td>Using Slots and Modifiers in Logic Grammars for Natural Language</td>
</tr>
<tr>
<td></td>
<td>Artificial Intelligence, vol. 18, pp. 327-367.</td>
</tr>
<tr>
<td>1989</td>
<td>Design of LMT: A Prolog-based Machine Translation System</td>
</tr>
<tr>
<td>1989</td>
<td>A New Version of Slot Grammar</td>
</tr>
<tr>
<td></td>
<td>Research Report RC 14506, IBM Research Division, Yorktown Heights</td>
</tr>
<tr>
<td>1990</td>
<td>Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars</td>
</tr>
<tr>
<td>2006</td>
<td>The Slot Grammar Lexical Formalism</td>
</tr>
<tr>
<td></td>
<td>Research Report RC 23977, IBM Research Division, Yorktown Heights</td>
</tr>
<tr>
<td>2007</td>
<td>An Arabic Slot Grammar Parser</td>
</tr>
<tr>
<td></td>
<td>Proceedings of the 5th Workshop on Important Unresolved Matters, pp. 81-88, Prague, Czech Republic</td>
</tr>
<tr>
<td>2010</td>
<td>Using Slot Grammar</td>
</tr>
<tr>
<td></td>
<td>Research Report RC 23978, IBM Research Division, Yorktown Heights</td>
</tr>
<tr>
<td>2012</td>
<td>Deep Parsing in Watson</td>
</tr>
<tr>
<td></td>
<td>IBM J. RES. &amp; DEV. VOL. 56 NO. 3/4</td>
</tr>
</tbody>
</table>
Dependency Trees

Example (’Watson Tree’)

Figure 1

ESG parse of the Jeopardy! question “Chandeliers look great but nowadays do not usually use these items from which their name is derived.”
Dependency Trees

Tree visualization:

- 90° rotation, left-to-right = top-to-bottom
- 4 'columns':
  - dependency relations
  - labels (slots)
  - node descriptions (predicates),
  - node descriptions (morphosyntactic and semantic features)
Dependency Trees

'Columns':
- slots
  - slot names
    arguments (1-pl): subj, obj, comp, auxcomp, pred, objprep, ...
    modifiers (0-pl): lconj, rconj, vadv, ndet, nrel, ...
  - slot fillers
    n (nominal), a (adjectival), en (participle),
    binf (bare infinitive), p (prepositional), ...
- word 'predicates'
  - word reading
    (optionally with link to the corresponding word sense in the lexicon)
  - 'functional'/semantic arguments
    (positions of the corresponding (head) words)
- node descriptions
Dependency Trees

'Columns':

- slots
- word 'predicates'
- node descriptions
  - morphosyntactic features (PoS + subclassification + agreement) noun cn pl, verb vfin pres pl, ...
  - semantic types physobj artf, sta, neg, ...

Slot Grammar

Dependency Trees
Example (LMT Tree')

:: Chandeliers look great but nowadays do not usually use these items from which their name is derived.  

Syntactic analysis no. 1. Evaluation = 5.31323 ...
Two Level Analysis

- syntactic level
- 'functional' semantic level

Figure 3: Ingredients of a Slot Grammar analysis structure
**Two Level Analysis**

Example: Passive Voice

'Deep' and 'Surface' Structure

![Diagram of a passive sentence parse]

*Figure 4: Parse of a passive sentence*
Two Level Analysis

used for ... control phenomena

Peter wants to devour a beefsteak.

Syntactic analysis no. 1. Evaluation = 0.221 ...

```
subj(n)  Peter527815(1)  noun(prop,[nom:sg],nwh)
top     want1339028(2,1,1,4) verb(fin[[pers3:sg],[pres,X1]])
preinf  preinf(3)        preinf
comp(inf) devour752172(4,1,6) verb(inf[full])
det     a851651(5)       det(sg, indef)
obj(n)  beefsteak886906(6) noun(cn,[acc:sg],nwh)
```
Two Level Analysis

used for ... control phenomena: subject control vs object control

:\Peter expects John to devour a beefsteak.

\textbf{Syntactic analysis no. 1. Evaluation = 0.321 ...}

\begin{verbatim}
subj(n)  Peter527815\{1\}  noun(prop,[nom:sg],nwh)
  top   expect1007620\{2,1,3,5\}  verb(fin[[pers3:sg],pres,X1])
  obj(n) John366059\{3\}  noun(prop,[acc:sg],nwh)
    preinf  preinf\{4\}  preinf
  comp<inf> devour\{5,3,7\}  verb(inf[full])
    ndet  a851651\{6\}  det(sg,indef)
     obj(n)  beefsteak886906\{7\}  noun(cn,[acc:sg],nwh)
\end{verbatim}

:\Peter promises John to devour a beefsteak.

\textbf{Syntactic analysis no. 1. Evaluation = 0.321 ...}

\begin{verbatim}
subj(n)  Peter527815\{1\}  noun(prop,[nom:sg],nwh)
  top   promise1196757\{2,1,5,3\}  verb(fin[[pers3:sg],pres,X1])
  iobj(n) John366059\{3\}  noun(prop,[acc:sg],nwh)
    preinf  preinf\{4\}  preinf
  comp<inf> devour\{5,1,7\}  verb(inf[full])
    ndet  a851651\{6\}  det(sg,indef)
     obj(n)  beefsteak886906\{7\}  noun(cn,[acc:sg],nwh)
\end{verbatim}
Two Level Analysis

used for . . .
control phenomena in extended auxiliary structures (with/without passive)

Figure 5: Bob may have been being taken to the station.
Two Level Analysis

used for ... (local and long-distance) extraposition

!: Who did Alice try to find?

Syntactic analysis no. 1. Evaluation = -0.39 ...

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>obj&lt;n&gt;</td>
<td>who785192&lt;1&gt;</td>
<td>noun&lt;pron&lt;wh&gt;,[X3!X4],wh&lt;X5&gt;&gt;</td>
</tr>
<tr>
<td>top</td>
<td>do1&lt;2,3,4&gt;</td>
<td>verb&lt;fin[[X1:sg],past,ind:q:wh<a href="">X2:1</a>&gt;]</td>
</tr>
<tr>
<td>subj&lt;n&gt;</td>
<td>Alice20514&lt;3&gt;</td>
<td>noun&lt;prop,[nom&lt;sg],nwh&gt;</td>
</tr>
<tr>
<td>auxcomp&lt;binf&gt;</td>
<td>try1316975&lt;4,3,6&gt;</td>
<td>verb&lt;inf&lt;bare&gt;&gt;</td>
</tr>
<tr>
<td>preinf</td>
<td>preinf&lt;5&gt;</td>
<td>preinf</td>
</tr>
<tr>
<td>obj&lt;inf&gt;</td>
<td>find1019159&lt;6,3,1,u&gt;</td>
<td>verb&lt;inf&lt;full&gt;&gt;</td>
</tr>
</tbody>
</table>
Two Level Analysis

used for ...
combining extraposition and control

`: Which horse do you want to win?

| Syntactic analysis no. 1. Evaluation = −0.68 ...
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ndet</td>
</tr>
<tr>
<td>obj(n)</td>
</tr>
<tr>
<td>top</td>
</tr>
<tr>
<td>subj(n)</td>
</tr>
<tr>
<td>auxcomp(binf)</td>
</tr>
<tr>
<td>preinf</td>
</tr>
<tr>
<td>comp&lt;inf&gt;</td>
</tr>
</tbody>
</table>

| Syntactic analysis no. 2. Evaluation = −0.68 ...
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ndet</td>
</tr>
<tr>
<td>obj(n)</td>
</tr>
<tr>
<td>top</td>
</tr>
<tr>
<td>subj(n)</td>
</tr>
<tr>
<td>auxcomp(binf)</td>
</tr>
<tr>
<td>preinf</td>
</tr>
<tr>
<td>comp&lt;inf&gt;</td>
</tr>
</tbody>
</table>
used for ...
extraposition over several levels

| Syntactic analysis no. 1 | Evaluation = -0.67869 ...
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>objprep&lt;n&gt;</td>
<td>who785192&lt;1&gt;</td>
</tr>
<tr>
<td>top</td>
<td>do1&lt;2,3,4&gt;</td>
</tr>
<tr>
<td>subj&lt;n&gt;</td>
<td>we&lt;3&gt;</td>
</tr>
<tr>
<td>auxcomp&lt;binf&gt;</td>
<td>think1302411&lt;4,3,6,u&gt;</td>
</tr>
<tr>
<td>subj&lt;n&gt;</td>
<td>Anna867386&lt;5&gt;</td>
</tr>
<tr>
<td>obj&lt;fin&gt;</td>
<td>claim926809&lt;6,5,7,u&gt;</td>
</tr>
<tr>
<td>obj&lt;fin&gt;</td>
<td>thatconj&lt;7,9&gt;</td>
</tr>
<tr>
<td>subj&lt;n&gt;</td>
<td>David147104&lt;8&gt;</td>
</tr>
<tr>
<td>fscomp</td>
<td>give1039044&lt;9,8,11,u,1&gt;</td>
</tr>
<tr>
<td>ndet</td>
<td>the1301140&lt;10&gt;</td>
</tr>
<tr>
<td>obj&lt;n&gt;</td>
<td>book896591&lt;11,u&gt;</td>
</tr>
<tr>
<td>iobj&lt;to&gt;</td>
<td>to1306632&lt;12,1&gt;</td>
</tr>
</tbody>
</table>
used for ... extraposition over several levels + passivization ...

: Who do we think Anna claimed that the book was written by?

Syntactic analysis no. 1. Evaluation = -0.67878 ...

```
objprep(n) who785192(1) noun(pron(wh),[X6!X7],wh(X8))
top do1(2,3,4) verb(fin([X1|pl],pres,ind:q:wh(X2:1)))
subj(n) we(3) noun(pron(pers1),[nom|pl],nwh)
auxcomp(binf) think1302411(4,3,6,u) verb(inf(bare))
subj(n) Anna867386(5) noun(prop,[nom:sg],nwh)
obj{fin} claim926809(6,5,7,u) verb(fin([X11|isg],past,dep:dcl:nwh))
subj(n) thatconj(7,10) fsubconj
nget the1301140(8) det(sg,def)
subj(n) book896591(9,u) noun(cn,[nom:sg],nwh)
fscomp be885385(10,9,11) verb(fin([X12|sg],past,dep:dcl:nwh))
pred write1350167(11,1,9,u,u) verb(pastpart)
agent by906279(12,1) prep(by,X17,1)
```
Two Level Analysis

used for ... coordination (scope and re-entrancy) ...

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>npt</td>
<td>the1301140&lt;1&gt;</td>
<td>det(sg,def)</td>
</tr>
<tr>
<td>subj</td>
<td>man456867&lt;2&gt;</td>
<td>noun(cn,[nom</td>
</tr>
<tr>
<td>lconj</td>
<td>see1241326&lt;3,2,8&gt;</td>
<td>verb(fin([pers3</td>
</tr>
<tr>
<td>top</td>
<td>and</td>
<td>verb(fin([pers3</td>
</tr>
<tr>
<td>vadu</td>
<td>p[b(probable);ly],5&gt;</td>
<td>adv&lt;X8,X9&gt;</td>
</tr>
<tr>
<td>rconj</td>
<td>hear1052900&lt;6,2,8,u&gt;</td>
<td>verb(fin([pers3</td>
</tr>
<tr>
<td>npt</td>
<td>the1301140&lt;7&gt;</td>
<td>det(sg,def)</td>
</tr>
<tr>
<td>objn</td>
<td>car911995&lt;8,u&gt;</td>
<td>noun(cn,[acc</td>
</tr>
</tbody>
</table>

The man sees and probably hears the car.

Syntactic analysis no. 1. Evaluation = 1.23...
used for ... extraposition and coordination ...

?!: Who did Alice say Bob wanted to find and talk with?

Syntactic analysis no. 1.  Evaluation = 2.6361 ...

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>obj(n)</strong></td>
<td><strong>wh085192(1)</strong></td>
<td><strong>noun[pron[[wh],[X9],[X10],[wh],[X11]]]</strong></td>
</tr>
<tr>
<td><strong>top</strong></td>
<td><strong>do1(2,3,4)</strong></td>
<td><strong>verb[fin[[X1][sg]],past,ind:q:[wh],[X2,1]]</strong></td>
</tr>
<tr>
<td><strong>subj(n)</strong></td>
<td><strong>Alice20514(3)</strong></td>
<td><strong>noun[prop,[nom][sg],nwh]</strong></td>
</tr>
<tr>
<td><strong>auxcomp(binf)</strong></td>
<td><strong>say1697011(4,3,6,u,u)</strong></td>
<td><strong>verb[inf[bare]]</strong></td>
</tr>
<tr>
<td><strong>subj(n)</strong></td>
<td><strong>Bob114968(5)</strong></td>
<td><strong>noun[prop,[nom][sg],nwh]</strong></td>
</tr>
<tr>
<td><strong>obj(fin)</strong></td>
<td><strong>want1339028(6,5,5,9)</strong></td>
<td><strong>verb[fin[[X12][sg]],past,dep:dc1:nwh]</strong></td>
</tr>
<tr>
<td><strong>preinf</strong></td>
<td><strong>preinf([7])</strong></td>
<td><strong>preinf</strong></td>
</tr>
<tr>
<td><strong>lconj</strong></td>
<td><strong>find1019159(8,5,1,1)</strong></td>
<td><strong>verb[inf[full]]</strong></td>
</tr>
<tr>
<td><strong>comp[inf]</strong></td>
<td><strong>and</strong></td>
<td><strong>verb[inf[full]]</strong></td>
</tr>
<tr>
<td><strong>rconj</strong></td>
<td><strong>talk1293520(10,5,u,1)</strong></td>
<td><strong>verb[inf[full]]</strong></td>
</tr>
<tr>
<td><strong>comp[p(with)]</strong></td>
<td><strong>with1347183(11,1)</strong></td>
<td><strong>prep[with,[X15,1]]</strong></td>
</tr>
</tbody>
</table>
Realization

- lexicon and grammar
- data structures
- parsing
Data Structures and Parsing

- \textit{words} (\textit{'wframe'})
- \textit{phrases}

- Parser turns lexical \textit{words} into \textit{phrases} and combines these phrases to phrases representing substrings of the sentence.
- Only one type of phrase:
  - 'categorial' differences expressed by PoS-type and by different/differently instantiated lists of arguments.
  - Saturated phrases: list of subcategorized arguments completely instantiated.
- Modular composition (using filler rules, extraposition rules, coordination rules, ordering constraints . . .)
- Extraposition done via a second list consisting of locally not realized and (presumably) extraposed elements.
Example

"I: He sees and was seen."

Syntactic analysis no. 1. Evaluation = 3.61 ...
Phrases

\[ \text{phrase}(\text{Span}, \text{Sense}, \text{Feas}, \text{Frame}, \text{Ext}, \text{Mods}) \]

\[ [LB, H|RB] \quad \text{mods}(LM, RM) \]

\[ \text{LM} = [\text{slot:phrase}(\ldots), \text{slot:phrase}(\ldots)] \]

\[ \text{LM} = [\text{slot:phrase}(\ldots), \text{slot:phrase}(\ldots)] \quad \text{(inverse Ordnung)} \]
Example

3.61: phrase[[1,3], coord<and,s<see,1241326>,s<be,885385>], verb<fin[[pers3],pres,X1: [slot<subj(n),ob,subj(n):1,t>], [vst[[nocont,thatcpref,sentsubj]]]],
mods<
  subj(n): phrase[[1,1], s<he,he>, noun<pron<def>,[nom:sg],nwh>,
  ,
  mods[[],[]]
>
  ,
  IC conj: phrase[[2,2], s<see,1241326>, verb<fin[[pers3],pres,X1:X2:X3>>,
  [slot<subj(n),ob,subj(n):1,t>,slot<obj<n ing>fin<wh>,op,X4,X5]],[
  ,
  mods[[],[]]
>,
  ,
  r conj: phrase[[4,4], s<be,885385>, verb<fin[[pers3],pres, X1:X2:X3>>,
  [slot<subj(n):v>,ob,subj(n):1,t>,slot<pred,op,pred:5,t]],
  [vst[[nocont,thatcpref,sentsubj]]]],
mods<
  ,
  pred: phrase[[5,5], s<see,1241326>, verb<pastpart>,
  [slot<agent,op,subj(n):X6,t>,slot<subj(n),op,subj(n):1,X7]],[
  ,
  mods[[],[]]
>
result(1,0,eval(head(noun,1), 0.0E+00), 0, phrase([1,1], s<he, he>, noun(pron(def), [nom|sg], A), [1,1], mods([], [])),
result(2,1,eval(head(fin,2), 0.0E+00), 0, phrase([2,2], s<see, 1241326>, verb(fin([pers3|sg], pres,A)), [slot(subj='''fin''|''he''|''op,C,D'',1,[1],mods([[],[]])]),
result(2,0,eval(head(fin,2), 1.0E-01), 0, phrase([1,2], s<see, 1241326>, verb(fin([pers3|sg], pres,A)), [slot(subj='''fin''|''wh''|''op,B,C'',1,[1],mods([subj(n):phrase([1,1], s<he, he>, noun(pron(def), [nom|sg], nwh), [1,1], mods([], [])),
result(2,1,eval(head(fin,2), 0.0E+00), 0, phrase([2,2], s<see, 1241340>, verb(fin([pers3|sg], pres,A)), [slot(subj='''fin''|''he''|''op,C,D'',1,[1],mods([[],[]])]),
result(3,2,eval(head(special,3), 0.0E+00), 0, phrase([3,3], and, special, [1,1], mods([[],[]])]),
result(4,3,eval(head(fin,4), 0.0E+00), 0, phrase([4,4], s<be, 885305>, verb(fin(A,past,B)), [slot(subj(n)'''fin''|''he''|''op,C,D'',1,[1],mods([[],[]])])]).
Lexicon

- lexical entries
- lexical rules
Words

- Lexicon
  different formats
    - (Prolog) data base with word records (using defaults, see below: 'Watson'-version)
    - Relational data base with tables for different types of lexical information

→ morphological analysis
→ *wframe*: interface to grammar
Slot Grammar-words ('Watson'-version)

Word  <  PoS Slot Frame SynDescrs SemDescrs

Slots for verbs
- subj, obj, iobj, pred, auxcomp, comp
- subj needs not to be specified
- comp
  - *Alice drove Betty to the store* vs. *Alice drove Betty crazy*
  - drive < v obj (comp lo a)
- pred
  - in ESG only used by the word *be*
  - be < v (subj n v) pred
  - *Bob is a teacher*
- auxcomp
  - mainly for modal verbs
  - *Bob may have been taken to the station*
Slot Grammar-words

ambiguities represented by alternative descriptions

talk < v (obj n (p about)) (comp (p to with))
    < v obj1 (compl (p into))
    < n nsubj (nobj n (p about))
      (ncomp (p to with))
Slot Grammar - words

- **Word** < PoS SlotFrame SynDescrs SemDescrs
- **PoS**: v, n, adj, adv, ...
- **SlotFrame**: [SlotNameOb(Opts) : SLConds, ...]
- **Ob**: obligatory (=1) / optional (∅): ex: (obj1(n) vs obj(n))
- **Opts**: n | v | p(... ) . . . (PoS of filler)
- **SynDescrs**: Boolean combination of syntactic types (ss, so, attr, ...)
- **SemDescr**: conjunction of semantic types (sent_obj & artefact)
- **SLConds**: Boolean combination of semantic types, SynDescrs, WordDescrs
Syntactic Types

Adverb features.

badadjmod. Preferred not to modify adjective.
compar. Comparative.
detadv. Can modify a determiner.
interj. An interjection.
initialmod. Modifies on left only as first modifier.
introadv. Adverb like *hello* that easily left-coordinates by comma-coordination.
invertadv. Adverb that allows certain constructions *Adv Verb Subj*.
loadv. Easily modifies a locational prep or adverb (particle).
locadv. Locative adverb like *above*.
noadvpre. Cannot have (qualifier) premodifier.
nopadv. Cannot modify preposition.
notadjmod. Cannot modify an adjective.
notinitialmod. Cannot appear clause-initially.
notleftmod. Cannot modify verb on left.
notnadv. Cannot premodify a noun.
Semantic Types

coll. Collection.


cpropn. Meant to suggest “common proper noun”. A propn, like “German” that can name a class, and so behaves also like a common noun.

cst. State or province in a country.

ctitle. A postposed company title, like “Inc.” or “Co.”

ctry. Named country (propn).

cty. Named city.

discipline. Branch of knowledge.


dy. Day. Named weekdays like Tuesday or named holiday days.

dmeas. Unit of electromagnetic measure.

event. Event.

f. Female.

feeling. Feeling, emotion.

geoarea. Geographical area.
Relational Database

LMT/Lingenio version

**Struktur**

**Lemma** (Bsp. "Wort")

**Homonym** ("Wort -> Wörter") HID₁, Stem

**Homonym** ("Wort -> Worte") HID₂, Stem

**Morph** (HID₂ -> MID₂), Flexion₂

**Homsense** (HID₂ -> SID₂), Variante

**Homsense** (HID₁ -> SID₁), Hauptform

**Homsense** (HID₁ -> SID₁), Hauptform

**Secstem** (MID₂ -> StID₂)

**Concept** (SID₁ -> CID₁”), Slots₁

**Concept** (SID₂ -> CID₂”), Slots₂

**Texts** (SID₂)
Relational Database

'Secondary' Table (built from 'primary tables, used during morphological analysis)
Relational Database

'Secondary' Table (built from 'primary tables, used during morphological analysis)

```sql
SELECT * FROM tlex.lmte WHERE word = 'want'
```
Morphological Analysis

Result (\(=\) input to syntactic analysis)

- type information
- \textit{wframe} representation:

\[
\textit{wframe}(\text{Position, Sense, FeatureDescription, SlotFrame})
\]
Morphological Analysis

Result (= input to syntactic analysis)

```plaintext
:: Peter walked.
lexisa(s(Peter,527815),m)
lexisa(s(Peter,527815),h)
lexisa(s(Peter,527815),uname)
lexisa(s(walk,1338444),transalt)
lexisa(s(walk,1338444),dirv)
wframe(1,s(Peter,527815),noun(prop,[_3739:sg],nwh),[])
wframe(2,s(walk,1338444),verb(fin(_3978,past,_3997)),[slot(subj(n),ob,_4056,t),slot(obj(n),op,_4115,_4126),slot(comp(p),op,_4177,_4188)])
wframe(2,s(walk,1338444),verb(pastpart),[slot(agent,op,subj(n):_4582,t),
slot(subj(n),op,subj(n):_4661,_4672),slot(comp(p),op,_4723,_4734)])
```
Lexical rules

→ applied in morphological analysis

!:: Peter presumably walked.
lexisa(s(Peter, 527815), m)

lexisa(s(Peter, 527815), h)

lexisa(s(Peter, 527815), vname)

lexisa(s(walk, 1338444), translalt)

lexisa(s(walk, 1338444), dirv)

wframe(1, s(Peter, 527815), noun(prop, [3908, isg, nwh], []))

wframe(2, s([b( presumable), ly], 749836), adv(_4187, _4198), []))

wframe(3, s(walk, 1338444), verb(fin(_4417, past, _4436), [slot(subj(n), _495, t), slot(obj(n), op, _4554, _4565), slot(comp(p), op, _4616, _4627)]))

wframe(3, s(walk, 1338444), verb(pastpart), [slot(agent, op, subj(n):_5022, slot(subj(n), op, obj(n):_5100, _5111), slot(comp(p), op, _5162, _5173)])
Slot Grammar phrase composition

Different types of rules (stored in modules)

▶ declaration of adjuncts
▶ Slot filler rules
▶ or-slot rules
▶ slot ordering rules
▶ declaration of obligatory slots
▶ extraposition rules
▶ coordination rules
▶ punctuation rules
▶ evaluation rules
Slot Grammar-composition

Adjunct declaration

Feas adjuncts {……}

noun(Ntype,NFeas,Wh) adjuncts

( ndet.ndetgen.ngen.nadj.nnoun.napos.
naposo.nprep.nrel.ncompar.nlselfst.pnspec.
ncompound.npbrk.ncorrel.nadv.nper).
Slot Grammar-composition

Slot filler rules

Slot  ==>  Conds

ndet ==> 
  f(det(Feas,Dtype)) , 
  hf(noun(NType,[Cases,Gen,StA|ST],Wh)) , 
  detagree(Feas,Dtype,Cases,Cases I), 
  hcf(noun(NType,[Cases I,Gen,StA|det],Wh)).
Slot Grammar-composition

Slot ordering rules

Slot on_left (:- Conds).
Slot on_right (:- Conds).

LSlot <= RSlot (:- Conds).
Slot Grammar-extraposition

Who did Alice say Bob wanted to find and talk with?

Syntactic analysis (3.3231) ...

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>who(X2)</td>
<td>noun</td>
</tr>
<tr>
<td></td>
<td>do1(X1,X3,X4)</td>
<td>verb</td>
</tr>
<tr>
<td>top</td>
<td>Alice(X3)</td>
<td>noun</td>
</tr>
<tr>
<td>subj(n)</td>
<td>say(X4,X3,X10,u)</td>
<td>verb</td>
</tr>
<tr>
<td>auxcmp(bare)</td>
<td>Bob(X6)</td>
<td>noun</td>
</tr>
<tr>
<td>subj(n)</td>
<td>want(X10,X6,X6,X11)</td>
<td>verb</td>
</tr>
<tr>
<td>obj(fin)</td>
<td>preinf(X11)</td>
<td>preinf</td>
</tr>
<tr>
<td>preinf</td>
<td>find(X12,X6,X2,u)</td>
<td>verb</td>
</tr>
<tr>
<td>lconj</td>
<td>and(X11,X12,X14)</td>
<td>verb</td>
</tr>
<tr>
<td>comp(ing)</td>
<td>talk(X14,X6,X2)</td>
<td>verb</td>
</tr>
<tr>
<td>rconj</td>
<td>with(X16,X2)</td>
<td>prep</td>
</tr>
</tbody>
</table>

Figure 1.
Slot Grammar-extraposition

- Declaration of extraposer slots
  \( extraposer(obj) :- Conds. \)
- Extraposition rules

Extraposition rules

- \text{Slot ext Lev} \implies Conds.
- \text{Slot exta} \implies Conds.
Slot Grammar-extraposition

Example

```
  rconj : phrase([10,10:11], s(talk,1293520), verb<inf(full)>,
     [slot<subj(n)>:op, subj<n>:5,t>, slot<obj>n<phrase>:about/of/in/on>, op,X13,X14>,
     slot<comp<phrase>:to\with>, op,c>,
     [slot<objprep<phrase>:about/of/in/on>],
     mods<

  comp<phrase>:to\with> : phrase([11,11:11], s(with,1347183), prep<with,X15,1>,
     [slot<objprep<phrase>:about/of/in/on>],
     mods<[],[],>

  >```

Weighting of rules

Evaluation Rules
Slot +++ Val (:- Conds)

obj(_) +++ 3.