Feature Structures and Unification Grammars

11-711 Algorithms for NLP
21 November 2017 – Part II
Linguistic features

• (Linguistic “features” vs. ML “features”.)
• Human languages usually include agreement constraints; in English, e.g., subject/verb
  – I often swim
  – He often swims
  – They often swim
• Could have a separate category for each minor type: N1s, N1p, ..., N3s, N3p, ...
  – Each with its own set of grammar rules!
A day without features...

• NP1s $\rightarrow$ Det-s N1s
• NP1p $\rightarrow$ Det-p N1p
  ...
• NP3s $\rightarrow$ Det-s N3s
• NP3p $\rightarrow$ Det-p N3p
  ...
• S1s $\rightarrow$ NP1s VP1s
• S1p $\rightarrow$ NP1p VP1p
• S3s $\rightarrow$ NP3s VP3s
• S3p $\rightarrow$ NP3p VP3p
Linguistic features

• *Could* have a separate category for each minor type: N1s, N1p, ... , N3s, N3p, ...
  – *Each* with its own set of grammar rules!

• Much better: represent these regularities using independent *features*: number, gender, person, ...

• Features are typically introduced by lexicon; checked and propagated by constraint equations attached to grammar rules
Feature Structures (FSs)

Having multiple orthogonal features with values leads naturally to **Feature Structures**:

\[
\text{[Det} \\
\text{[root: } a] \\
\text{[number: sg } ]\]
\]

A feature structure’s values can in turn be FSs:

\[
\text{[NP} \\
\text{[agreement: } [[[\text{number: sg}] \\
\text{[person: 3rd}]]]]
\]

Feature Path: <NP agreement person>
Adding constraints to CFG rules

• \( S \rightarrow NP\ VP \)
  \(<\text{NP\ number}> = <\text{VP\ number}>\)

• \( NP \rightarrow \text{Det Nominal} \)
  \(<\text{NP\ head}> = <\text{Nominal\ head}>\)
  \(<\text{Det\ head\ agree}> = <\text{Nominal\ head\ agree}>\)
FSs from lexicon, constrs. from rules

Lexicon entry:

[Det
 [root: a]
 [number: sg ]]

Rule with constraints:

NP → Det Nominal

<NP number> = <Det number>

<NP number> = <Nominal number>

• Combine to get result:

[NP [Det
 [root: a]
 [number: sg ]]
 [Nominal [number: sg] ...]
 [number: sg]]
Similar issue with VP types

Another place where grammar rules could explode:

Jack laughed

$\text{VP} \rightarrow \text{Verb}$  \textit{for many specific verbs}

Jack found a key

$\text{VP} \rightarrow \text{Verb} \ \text{NP}$  \textit{for many specific verbs}

Jack gave Sue the paper

$\text{VP} \rightarrow \text{Verb} \ \text{NP} \ \text{NP}$  \textit{for many specific verbs}
Verb Subcategorization

Verbs have sets of allowed args. Could have many sets of VP rules. Instead, have a SUBCAT feature, marking sets of allowed arguments:

+none -- Jack laughed
+np -- Jack found a key
+np+np -- Jack gave Sue the paper
+vp:inf -- Jack wants to fly
+np+vp:inf -- Jack told the man to go
+vp:ing -- Jack keeps hoping for the best
+np+vp:ing -- Jack caught Sam looking at his desk
+np+vp:base -- Jack watched Sam look at his desk
+np+pp:to -- Jack gave the key to the man

+pp:loc -- Jack is at the store
+np+pp:loc -- Jack put the box in the corner
+pp:mot -- Jack went to the store
+np+pp:mot -- Jack took the hat to the party
+adjp -- Jack is happy
+np+adjp -- Jack kept the dinner hot
+sfor -- Jack believed that the world was flat
+sfor -- Jack hoped for the man to win a prize

50-100 possible frames for English; a single verb can have several. (Notation from James Allen “Natural Language Understanding”)
Frames for “ask”  
*(in J+M notation)*

<table>
<thead>
<tr>
<th>Subcat</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Quo</em></td>
<td>asked [<em>{Quo} “What was it like?”</em>]</td>
</tr>
<tr>
<td><em>NP</em></td>
<td>asking [_{NP} a question]</td>
</tr>
<tr>
<td><em>Sw(h)</em></td>
<td>asked [_{Sw(h)} what trades you’re interested in]</td>
</tr>
<tr>
<td><em>Sto</em></td>
<td>ask [_{Sto} him to tell you]</td>
</tr>
<tr>
<td><em>PP</em></td>
<td>that means asking [_{PP} at home]</td>
</tr>
<tr>
<td><em>V(to)</em></td>
<td>asked [_{V(to)} to see a girl called Evelyn]</td>
</tr>
<tr>
<td><em>NP Sif</em></td>
<td>asked [<em>{NP} him] [</em>{Sif} whether he could make]</td>
</tr>
<tr>
<td><em>NP NP</em></td>
<td>asked [<em>{NP} myself] [</em>{NP} a question]</td>
</tr>
<tr>
<td><em>NP Sw(h)</em></td>
<td>asked [<em>{NP} him] [</em>{Sw(h)} why he took time off]</td>
</tr>
</tbody>
</table>
Adding transitivity constraint

• $S \rightarrow NP \ VP$
  
  $<NP \ number> = <VP \ number>$

• $NP \rightarrow \text{Det Nominal}$
  
  $<NP \ head> = <\text{Nominal} \ head>$
  
  $<\text{Det} \ head \ agree> = <\text{Nominal} \ head \ agree>$

• $VP \rightarrow \text{Verb} \ NP$
  
  $<VP \ head> = <\text{Verb} \ head>$
  
  $<VP \ head \ subcat> = +np$ (which means transitive)
Applying a verb subcat feature

Lexicon entry:

[Verb

  [root: found]

  [head: find]

  [subcat: +np ]]

• Combine to get result:

  [VP [Verb

    [root: found]

    [head: find]

    [subcat: +np ]]

  [NP ...]

  [head: [find [subcat: +np]]]]

Rule with constraints:

VP → Verb   NP

<VP head> = <Verb head>

<VP head subcat> = +np
Relation to LFG constraint notation

• $\text{VP } \rightarrow \text{Verb} \quad \text{NP}$
  \[
  \langle \text{VP head} \rangle = \langle \text{Verb head} \rangle
  \]
  \[
  \langle \text{VP head subcat} \rangle = +\text{np}
  \]

  *from JM book is the same as the LFG expression*

• $\text{VP } \rightarrow \text{Verb} \quad \text{NP}$
  \[
  (↑ \text{head}) = (↓ \text{head})
  \]
  \[
  (↑ \text{head subcat}) = +\text{np}
  \]
Unification

• Merging FSs (and failing if not possible) is called **Unification**

• Simple FS examples:

  \[\text{[number sg]} \cup \text{[number sg]} = \text{[number sg]}\]
  \[\text{[number sg]} \cup \text{[number pl]} \text{ FAILS}\]
  \[\text{[number sg]} \cup \text{[number []]} = \text{[number sg]}\]
  \[\text{[number sg]} \cup \text{[person 3rd]} = \text{[number sg, person 3rd]}\]
Recap: applying constraints

Lexicon entry:

[Det
  [root: a]
  [number: sg ]]

• Combine to get result:

[NP [Det
  [root: a]
  [number: sg ]]
  [Nominal [number: sg] ...]
  [number: sg]]

Rule with constraints:

NP → Det Nominal

<NP number> = <Det number>
<NP number> = <Nominal number>
Turning constraint eqns. into FS

Lexicon entry:

[Det
 [root: a]
 [number: sg ]]

• Combine to get result:

[NP [Det
 [root: a]
 [number: sg ]]
 [Nominal [number: sg]
 ...]
 [number: sg]]

Rule with constraints:

NP → Det Nominal

<NP number> = <Det number>
<NP number> = <Nominal number>

becomes:

[NP [Det [number: (1) ]]
 [Nominal [number: (1) ]
 ...]
 [number: (1) ]]
Another example

This (oversimplified) rule:

\[ S \rightarrow NP \ VP \]
\[ <S \ subject> = NP \]
\[ <S \ agreement> = <S \ subject \ agreement> \]

turns into this DAG:

\[ [S [subject (1) [agreement (2) ]] [agreement (2) ] [NP (1) ] [VP ] \]
Unification example without “EQ“

[agreement [number sg],
 subject [agreement [number sg]]]
⊔[subject [agreement [person 3rd, 
 number sg]]]
= [agreement [number sg],
 subject [agreement [person 3rd, 
 number sg]]]

• <agreement number> is equal to <subject agreement number>, but not EQ
Unification example with “EQ“

[agreement (1), subject [agreement (1)]]

⊔[subject [agreement [person 3rd, number sg]]
= [agreement (1),
    subject [agreement (1) [person 3rd, number sg]]]]

• <agreement number> is <subject agreement number> (EQ), so they are equal
Representing FSs as DAGs

- Taking feature paths seriously
- May be easier to think about than numbered cross-references in text
- [cat NP, agreement [number sg, person 3rd]]
Re-entrant FS as DAGs

• [cat S, head [agreement (1) [number sg, person 3rd], subject [agreement (1)]]]
Seems tricky. Why bother?

- Unification allows the systems that use it to handle many complex phenomena in “simple” elegant ways:
  - There **seems** to be a **dog** in the yard.
  - There **seem** to be **dogs** in the yard

- Unification makes this work smoothly.
  - Make the Subjects of the clauses EQ:
    - \( <\text{VP subj}> = <\text{VP COMP subj}> \)
    
        \[ \text{[VP [subj: (1)] [COMP [subj: (1)]]]} \]

  - (Ask Lori Levin for LFG details.)
Real Unification-Based Parsing

• $X_0 \rightarrow X_1 X_2$
  - $<X_0 \text{ cat}> = S$, $<X_1 \text{ cat}> = NP$, $<X_2 \text{ cat}> = VP$
  - $<X_1 \text{ head agree}> = <X_2 \text{ head agree}>$
  - $<X_0 \text{ head}> = <X_2 \text{ head}>$

• $X_0 \rightarrow X_1 \text{ and } X_2$
  - $<X_1 \text{ cat}> = <X_2 \text{ cat}>$, $<X_0 \text{ cat}> = <X_1 \text{ cat}>$

• $X_0 \rightarrow X_1 X_2$
  - $<X_1 \text{ orth}> = how$, $<X_2 \text{ sem}> = <\text{SCALAR}>$
Complexity

• Earley modification: “search the chart for states whose DAGs unify with the DAG of the completed state”. Plus a lot of copying.

• Unification parsing is “quite expensive”.
  – NP-Complete in some versions.
  – Early AWB paper on Turing Equivalence(!)

• So maybe too powerful?
  (like GoTo or Call-by-Name?)
  – Add restrictions to make it tractable:
    • Tomita’s Pseudo-unification (Tomabechi too)
    • Gerald Penn work on tractable HPSG: ALE
Formalities: subsumption

- Less specific FS1 **subsumes** more specific FS2
  \( \text{FS1} \sqsubseteq \text{FS2} \)  (Inverse is \( \text{FS2} \text{ extends} \text{ FS1} \))
- Subsumption relation forms a **semilattice**, at the top: \([\text{[]}\] )

\[
\begin{align*}
[\text{number sg}] & \quad [\text{person 3}] & \quad [\text{number pl}] \\
[\text{number sg, person 3}] \\
\end{align*}
\]

- Unification defined wrt semilattice:
  \( F \sqcup G = H \text{ s.t. } F \sqsubseteq H \text{ and } G \sqsubseteq H \)
  \( H \text{ is the Most General Unifier (MGU)} \)
Hierarchical Types

Hierarchical types allow *values* to unify too (or not):
Hierarchical subcat frames

Many verbs share *subcat* frames, some with more arguments specified than others:
Questions?