Review: Inflectional Morphology and syntactic agreement

• Morphology is the study of the internal structure of words.

  – Derivational morphology. How new words are created from existing words.
    • [grace]
    • [(grace)ful]
    • [un(grace)ful]]

  – Inflectional morphology. How features relevant to the syntactic context of a word are marked on that word.
    • This example illustrates number (singular and plural) and tense (present and past).
    • Green indicates irregular. Blue indicates zero marking of inflection. Red indicates regular inflection.
    • This student walks.
    • These students walk.
    • These students walked.

  – Compounding. Creating new words by combining existing words
    • With or without spaces: surfboard, golf ball, blackboard
Review: Features, morphology, FSTs:

Figure 3.19: Generating or parsing with FST lexicon and rules

parse

generate
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 → Det-s N1s
- NP1p → Det-p N1p
  ...
- NP3s → Det-s N3s
- NP3p → Det-p N3p
  ...
- S1s → NP1s VP1s
- S1p → NP1p VP1p
- S3s → NP3s VP3s
- S3p → 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**:

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

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

```
[NP
  [agreement: [[number: sg]
    [person: 3rd]]]]
```

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

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

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

Lexicon entry:

\[
\begin{align*}
&D \quad \text{[root: } a \text{]} \\
&\text{[number: sg]} \\
\end{align*}
\]

Rule with constraints:

\[
\begin{align*}
&\text{NP} \rightarrow \text{Det Nominal} \\
&<\text{NP number}> = <\text{Det number}> \\
&<\text{NP number}> = <\text{Nominal number}>
\end{align*}
\]

• Combine to get result:

\[
\begin{align*}
&\text{NP [Det} \\
&\quad \text{[root: } a \text{]} \\
&\quad \text{[number: sg]} \\
&\quad \text{[Nominal [number: sg] ...]} \\
&\quad \text{[number: sg]}]
\end{align*}
\]
Similar issue with VP types

Another place where grammar rules could explode:

Jack laughed

\[ \text{VP} \rightarrow \text{Verb} \quad \text{for many } \text{specific} \text{ verbs} \]

Jack found a key

\[ \text{VP} \rightarrow \text{Verb NP} \quad \text{for many } \text{specific} \text{ verbs} \]

Jack gave Sue the paper

\[ \text{VP} \rightarrow \text{Verb NP NP} \quad \text{for many } \text{specific} \text{ verbs} \]
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
+s:that -- Jack believed that the world was flat
+s:for -- 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”)

Verb frames are not totally semantic

• It does seem to be partly lexical:
  John wants to fly
  John likes to fly
  John likes flying
  *John wants flying

• Can vary with dialect:
  ??The car needs washed  (only in Pittsburghese?)
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 ([_Quo] “What was it like?”)</td>
</tr>
<tr>
<td><em>NP</em></td>
<td>asking ([_NP] a question)</td>
</tr>
<tr>
<td><em>Swh</em></td>
<td>asked ([_Swh] 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>Vto</em></td>
<td>asked ([_Vto] to see a girl called Evelyn)</td>
</tr>
<tr>
<td><em>NP Sif</em></td>
<td>asked ([_NP] him) ([_Sif] whether he could make)</td>
</tr>
<tr>
<td><em>NP NP</em></td>
<td>asked ([_NP] myself) ([_NP] a question)</td>
</tr>
<tr>
<td><em>NP Swh</em></td>
<td>asked ([_NP] him) ([_Swh] why he took time off)</td>
</tr>
</tbody>
</table>
Adding transitivity constraint

• $S \rightarrow NP \; VP$
  \[
  <NP \; number> \; = \; <VP \; number>
  \]

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

• $VP \rightarrow \text{Verb} \; NP$
  \[
  <VP \; head> \; = \; <\text{Verb} \; head>
  \]
  \[
  <VP \; head \; subcat> \; = \; +np \quad \text{(which means transitive)}
  \]
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]]]
Relation to LFG constraint notation

• \( VP \rightarrow \text{Verb NP} \)
  \(<\text{VP head}> = <\text{Verb head}>\)
  \(<\text{VP head subcat}> = +\text{np}\)

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

• \( VP \rightarrow \text{Verb NP} \)
  \((\uparrow \text{head}) = (\downarrow \text{head})\)
  \((\uparrow \text{head subcat}) = +\text{np}\)
Unification

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

• Simple FS examples:

\[
\text{[number sg]} \cup \text{[number sg]} = \text{[number sg]}
\]

\[
\text{[number sg]} \cup \text{[number pl]} \quad \text{FAILS}
\]

\[
\text{[number sg]} \cup \text{[number [ ]]} = \text{[number sg]}
\]

\[
\text{[number sg]} \cup \text{[person 3rd]} = \text{[number sg, person 3rd]}
\]
New kind of “=” sign

• Already had two meanings in programming:
  – “:=” means “make the left be equal to the right”
  – “==” means “the left and right happen to be equal”

• Now, a third meaning:
  – ⊨ “=” means “make the left and the right be the same thing (from now on)”
Recap: applying constraints

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]]
Unifying constraints (1)

Lexicon entries:

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

[Nominal
  [root: dog]
  [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
    [root: dog]
    [number: sg] ...]
  [number: sg]]
Unifying constraints (2)

Lexicon entries:

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

[Nominal
  [root: dogs]
  [number: pl ]]

• Rule with constraints:
  NP → Det Nominal
  \[<\text{NP number}> = <\text{Det number}>\]
  \[<\text{NP number}> = <\text{Nominal number}>\]

• Combine to get result:
  FAIL
Unifying constraints (3)

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

[Nominal
  [root: deer]
  [number: {sg pl} ]]

• Rule with constraints:
  NP → Det Nominal
  \(<\text{NP number}> = <\text{Det number}>\)
  \(<\text{NP number}> = <\text{Nominal number}>\)

• Combine to get result:
  [NP [Det
    [root: a]
    [number: sg ]]
  [Nominal
    [root: deer]
    [number: sg] …]
  [number: sg]]
Turning constraint eqns. into FS

FS1: Lexicon entry:
[Det
 [root: a]
 [number: sg ]]

• Combine FS1 and FS2 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 FS2:
[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:

\[
[S \ [\text{subject} \ (1) \\
\quad [\text{agreement} \ (2) ] ]
\quad [\text{agreement} \ (2) ]
[NP \ (1) ]
[VP ]
\]
“Unification” example without “EQ”

\[
\text{[agreement [number sg], } \\
\text{ subject [agreement [number sg]]] } \\
\Box [\text{subject [agreement [person 3rd, } } \\
\text{ number sg]]}] \\
\]

\[
= [\text{agreement [number sg], } \\
\text{ subject [agreement [person 3rd, } } \\
\text{ number sg]]}] \\
\]

- \text{<agreement>} is (initially) equal to \text{<subject agreement>}, but \textbf{not} EQ
- So \textbf{not} equal anymore \textit{after} the operation: \textbf{<agreement person>} is still null
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> *is* <subject agreement> (EQ), so they are equal

• and *stay* equal, *always*, in the future:
  <agreement person> *is* 3rd afterwards!
Ordinary 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:
    \[<\text{VP subj}> = <\text{VP COMP subj}>\]
    \[[\text{VP } [\text{subj: (1)}] [\text{COMP [subj: (1)]}]]\]
  – (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

• J&M II: “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
  \( FS1 \sqsubseteq FS2 \) (Inverse is FS2 *extends* FS1)
- Subsumption relation forms a *semilattice*,
  at the top: []

[ [number sg] [person 3] [number pl] ]

[ [number sg, person 3] ]

- Unification defined wrt semilattice:
  \( F \sqcup G = H \) s.t. \( F \sqsubseteq H \) and \( G \sqsubseteq H \)
  \( H \) 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?
# Subcategorization

## Noun Phrase Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>There</td>
<td>nonreferential there</td>
</tr>
<tr>
<td>It</td>
<td>nonreferential it</td>
</tr>
<tr>
<td>NP</td>
<td>noun phrase</td>
</tr>
</tbody>
</table>

## Preposition Phrase Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP</td>
<td>preposition phrase</td>
</tr>
<tr>
<td>PPing</td>
<td>gerundive PP</td>
</tr>
<tr>
<td>PPart</td>
<td>particle</td>
</tr>
</tbody>
</table>

## Verb Phrase Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPbrst</td>
<td>bare stem VP</td>
</tr>
<tr>
<td>VPto</td>
<td>to-marked infin. VP</td>
</tr>
<tr>
<td>VPwh</td>
<td>wh-VP</td>
</tr>
<tr>
<td>VPIng</td>
<td>gerundive VP</td>
</tr>
</tbody>
</table>

## Complement Clause types

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sfin</td>
<td>finite clause</td>
</tr>
<tr>
<td>Swh</td>
<td>wh-clause</td>
</tr>
<tr>
<td>Sif</td>
<td>whether/if clause</td>
</tr>
<tr>
<td>Sing</td>
<td>gerundive clause</td>
</tr>
<tr>
<td>Sto</td>
<td>to-marked clause</td>
</tr>
<tr>
<td>Sforto</td>
<td>for-to clause</td>
</tr>
<tr>
<td>Sbrst</td>
<td>bare stem clause</td>
</tr>
</tbody>
</table>

## Other Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>AjP</td>
<td>adjective phrase</td>
</tr>
<tr>
<td>Quo</td>
<td>quotes</td>
</tr>
</tbody>
</table>