

Feature Structures and Unification Grammars

11-711 Algorithms for NLP

15 October 2019 – 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 → 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

- $S \rightarrow NP VP$
 <NP number> = <VP number>
- $NP \rightarrow Det Nominal$
 <NP head> = <Nominal head>
 <Det head agree> = <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

VP \rightarrow Verb *for many specific verbs*

Jack found a key

VP \rightarrow Verb NP *for many specific verbs*

Jack gave Sue the paper

VP \rightarrow Verb NP NP *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
+sthat -- 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)

Subcat	Example
<i>Quo</i>	asked [<i>Quo</i> “What was it like?”]
<i>NP</i>	asking [<i>NP</i> a question]
<i>Swh</i>	asked [<i>Swh</i> what trades you’re interested in]
<i>Sto</i>	ask [<i>Sto</i> him to tell you]
<i>PP</i>	that means asking [<i>PP</i> at home]
<i>Vto</i>	asked [<i>Vto</i> to see a girl called Evelyn]
<i>NP Sif</i>	asked [<i>NP</i> him] [<i>Sif</i> whether he could make]
<i>NP NP</i>	asked [<i>NP</i> myself] [<i>NP</i> a question]
<i>NP Swh</i>	asked [<i>NP</i> him] [<i>Swh</i> why he took time off]

Adding transitivity constraint

- $S \rightarrow NP VP$
<NP number> = <VP number>
- $NP \rightarrow Det Nominal$
<NP head> = <Nominal head>
<Det head agree> = <Nominal head agree>
- $VP \rightarrow \text{Verb} NP$
<VP head> = <Verb head>
<VP head subcat> = +np (*which means transitive*)

Applying a verb subcat feature

Lexicon entry:

[Verb
[root: *found*]
[head: find]
[subcat: +np]]

Rule with constraints:

VP → Verb NP
<VP head> = <Verb head>
<VP head 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 → Verb NP
 <VP head> = <Verb head>
 <VP head subcat> = +np

from JM book is the same as the LFG expression

- VP → Verb NP
 (↑ head) = (↓ head)
 (↑ head subcat) = +np

Unification

- Merging FSs (and failing if not possible) is called ***Unification***
- Simple FS examples:

[number sg] \sqcup [number sg] = [number sg]

[number sg] \sqcup [number pl] **FAILS**

[number sg] \sqcup [number []] = [number sg]

[number sg] \sqcup [person 3rd] = [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:
 - \sqcup “=” 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]]

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> is (initially) equal to <subject agreement>, but **not** EQ
- So not equal anymore *after* the operation:
<agreement person> is still null

Unification example with “EQ”

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

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

= [agreement (1),

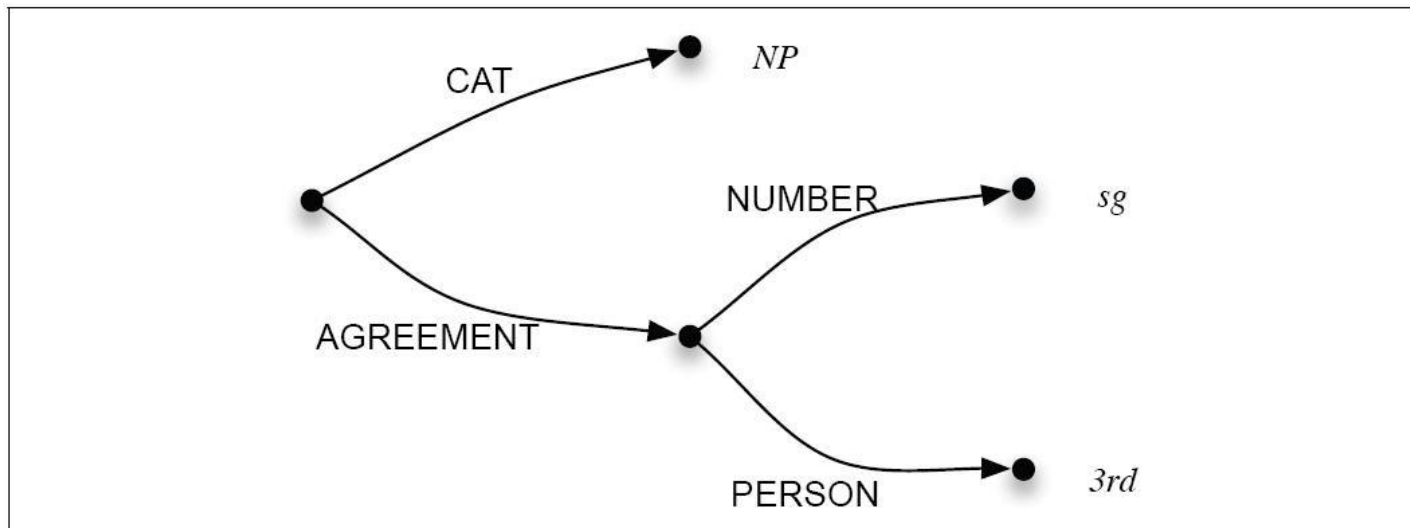
subject [agreement (1) [person 3rd,

number sg]]]

- <agreement> *is* <subject agreement> (EQ), so they are equal
- and *stay* equal, always, in the future:
<agreement person> *is* 3rd afterwards!

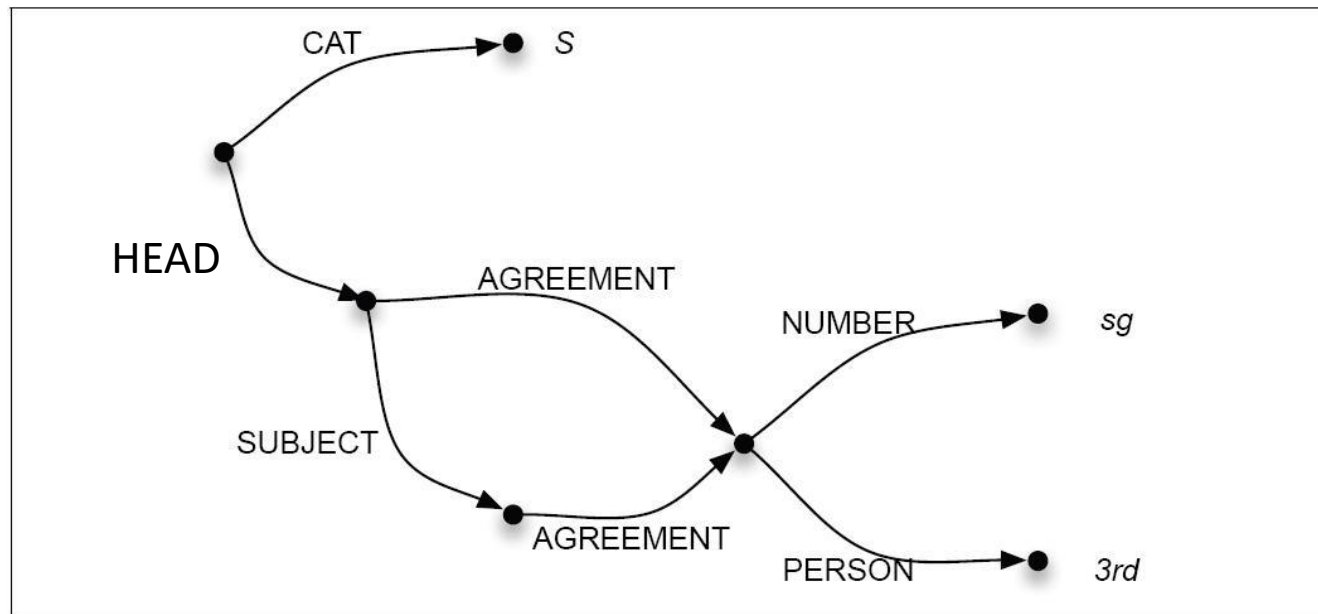
Ordinary FSs as DAGs

- 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:
 <VP subj> = <VP COMP subj>
 [VP [subj: (1)] [COMP [subj: (1)]]]
 - (Ask Lori Levin for LFG details.)

Real Unification-Based Parsing

- $X_0 \rightarrow X_1 X_2$
 $\langle X_0 \text{ cat} \rangle = S, \langle X_1 \text{ cat} \rangle = NP, \langle X_2 \text{ cat} \rangle = VP$
 $\langle X_1 \text{ head agree} \rangle = \langle X_2 \text{ head agree} \rangle$
 $\langle X_0 \text{ head} \rangle = \langle X_2 \text{ head} \rangle$
- $X_0 \rightarrow X_1 \text{ and } X_2$
 $\langle X_1 \text{ cat} \rangle = \langle X_2 \text{ cat} \rangle, \langle X_0 \text{ cat} \rangle = \langle X_1 \text{ cat} \rangle$
- $X_0 \rightarrow X_1 X_2$
 $\langle X_1 \text{ orth} \rangle = \textit{how}, \langle X_2 \text{ sem} \rangle = \langle \text{SCALAR} \rangle$

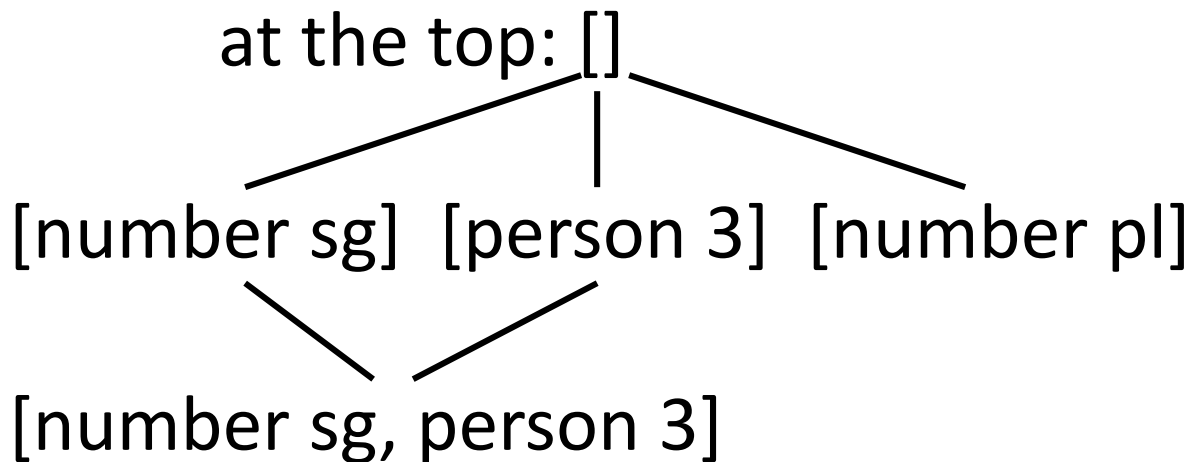
Complexity

- 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
 $FS1 \sqsupseteq FS2$ (Inverse is FS2 **extends** FS1)

- Subsumption relation forms a **semilattice**,



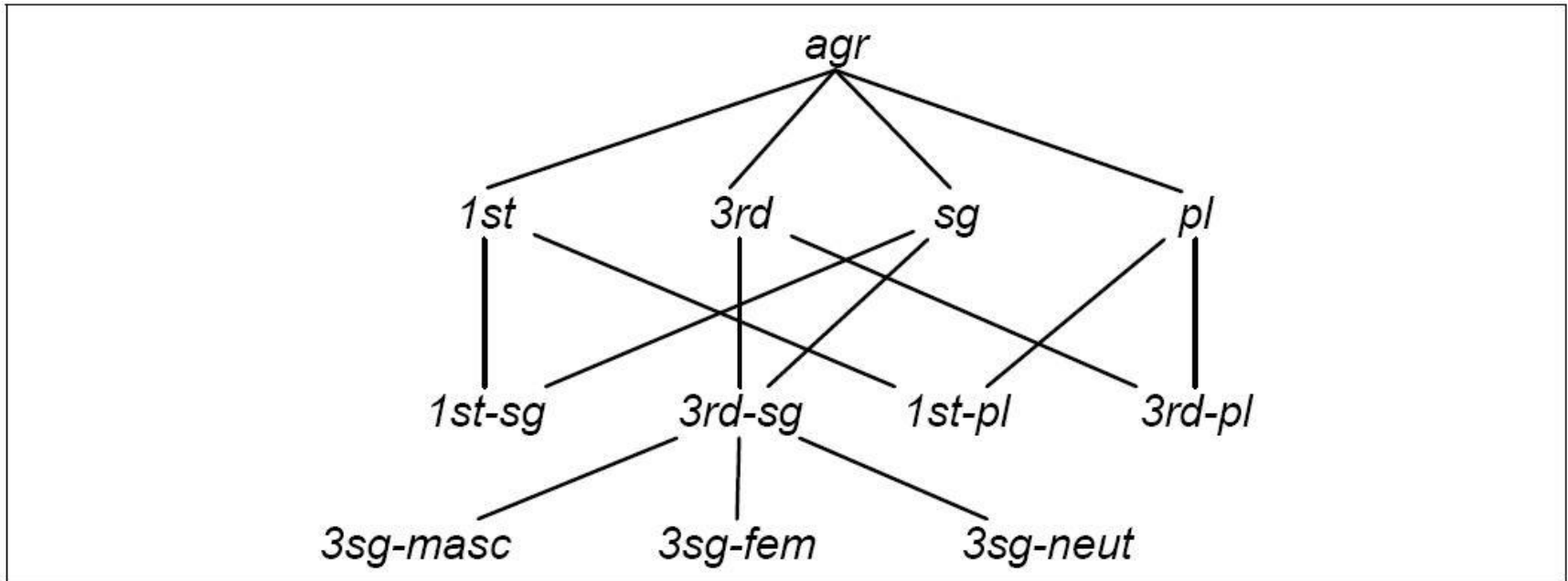
- Unification defined wrt semilattice:

$F \sqcup G = H$ s.t. $F \sqsupseteq H$ and $G \sqsupseteq H$

H is the Most General Unifier (MGU)

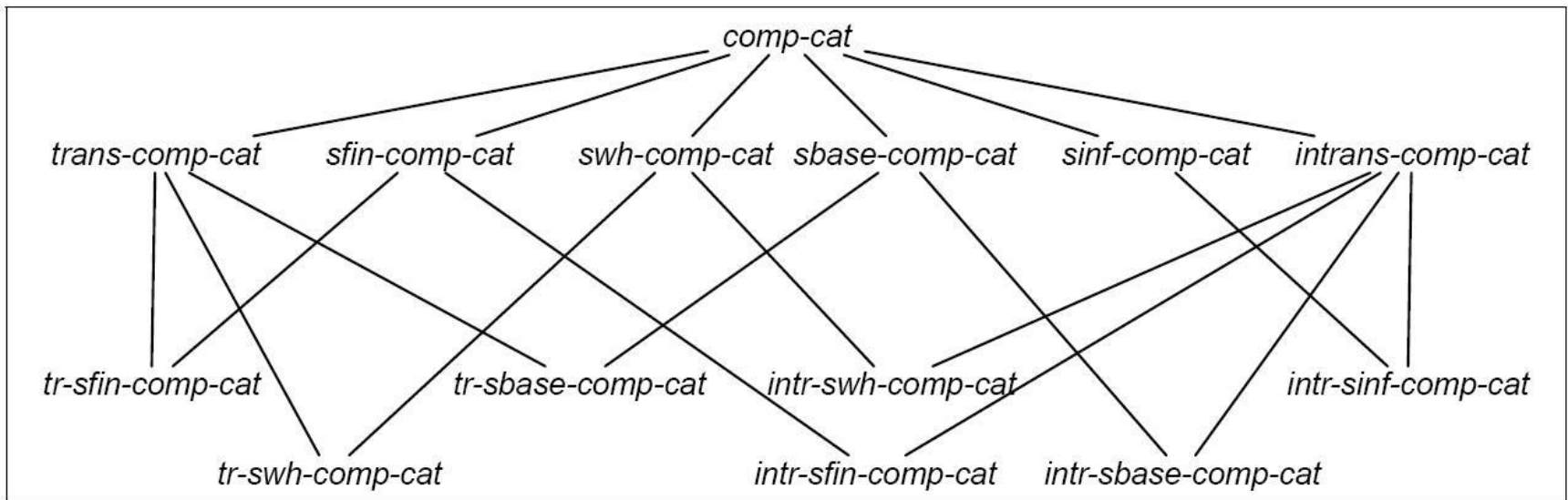
Hierarchical Types

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		
There	nonreferential there	There <i>is still much to learn</i>
It	nonreferential it	It <i>was evident that my ideas</i>
NP	noun phrase	<i>As he was relating</i> his story
Preposition Phrase Types		
PP	preposition phrase	<i>couch their message</i> in terms
PPing	gerundive PP	<i>censured him</i> for not having intervened
PPpart	particle	<i>turn it</i> off
Verb Phrase Types		
VPbrst	bare stem VP	<i>she could</i> discuss it
VPto	to-marked infin. VP	<i>Why do you want</i> to know?
VPwh	wh-VP	<i>it is worth considering</i> how to write
VPing	gerundive VP	<i>I would consider</i> using it
Complement Clause types		
Sfin	finite clause	<i>maintain</i> that the situation was unsatisfactory
Swh	wh-clause	<i>it tells us</i> where we are
Sif	whether/if clause	<i>ask</i> whether Aristophanes is depicting a
Sing	gerundive clause	<i>see</i> some attention being given
Sto	to-marked clause	<i>know</i> themselves to be relatively unhealthy
Sforto	for-to clause	<i>She was waiting</i> for him to make some reply
Sbrst	bare stem clause	<i>commanded</i> that his sermons be published
Other Types		
AjP	adjective phrase	<i>thought it</i> possible
Quo	quotes	<i>asked</i> “What was it like?”

- (Add an example full parse “he runs”)
 - After “another example” slide?
- Get from F15(?) Recitation notes??