Feature Structures and Unification Grammars

11-711 Algorithms for NLP
15 November 2016 – 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 swim**s**
  - 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 → NP VP

  <NP number> = <VP number>

- NP → 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 → Verb *for many specific verbs*

Jack found a key

VP → Verb NP *for many specific verbs*

Jack gave Sue the paper

VP → 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
Quo	asked [$_{Quo}$ "What was it like?"]
NP	asking [$_{NP}$ a question]
Swh	asked [$_{Swh}$ what trades you're interested in]
Sto	ask [$_{Sto}$ him to tell you]
PP	that means asking [$_{PP}$ at home]
Vto	asked [$_{Vto}$ to see a girl called Evelyn]
NP Sif	asked [$_{NP}$ him] [$_{Sif}$ whether he could make]
NP NP	asked [$_{NP}$ myself] [$_{NP}$ a question]
NP Swh	asked [$_{NP}$ him] [$_{Swh}$ why he took time off]

Adding transitivity constraint

- S → NP VP

  <NP number> = <VP number>

- NP → Det Nominal

  <NP head> = <Nominal head>

  <Det head agree> = <Nominal head agree>

- VP → 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]]

- 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

- 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] ⊔ [number sg] = [number sg]

[number sg] ⊔ [number pl] FAILS

[number sg] ⊔ [number []] = [number sg]

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

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 → 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"

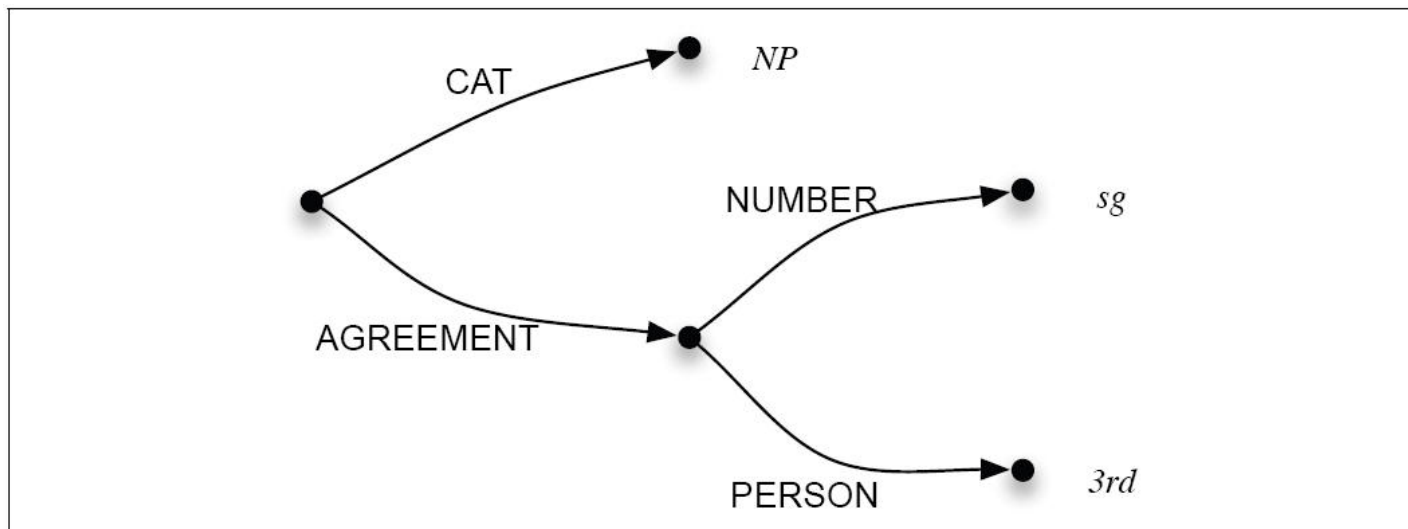[agreement (1), subject [agreement (1)]]

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

= [agreement (1),

   subject [agreement (1) [person 3rd,

                              number sg]]]

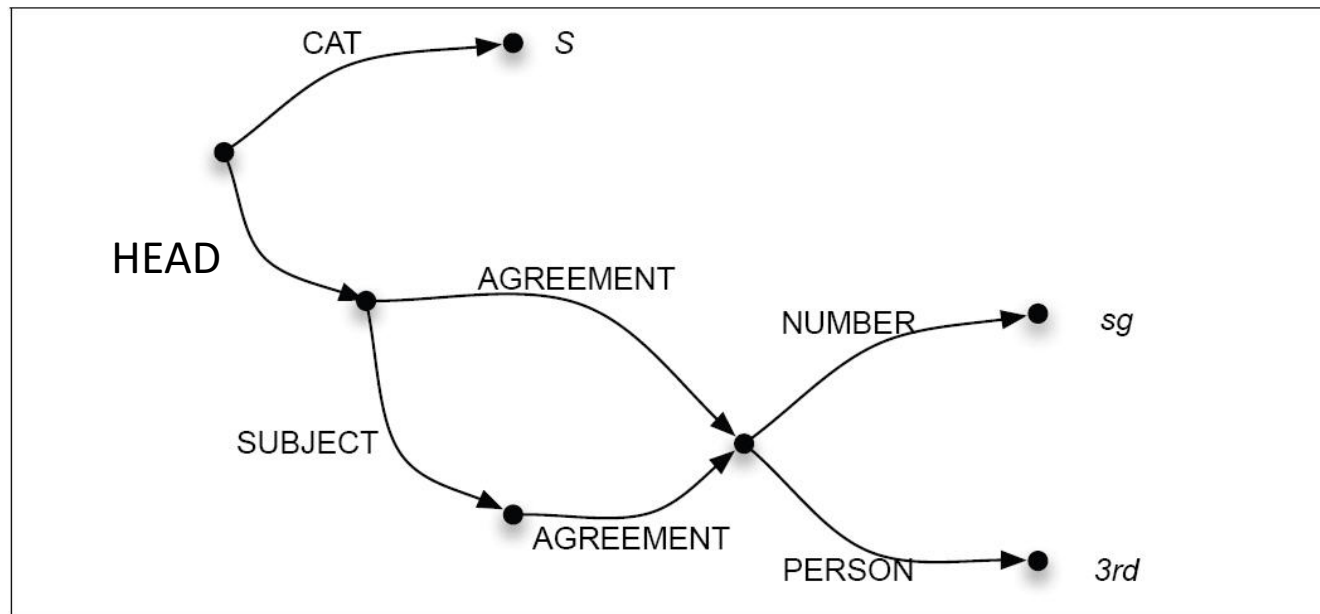- <agreement number> *is* <subject agreement number> (EQ), so they are equal

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 <u>seems</u> to be <u>a dog</u> in the yard.
  - There <u>seem</u> to be <u>dogs</u> 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

- X0 → X1 X2

  <X0 cat> = S, <X1 cat> = NP, <X2 cat> = VP

  <X1 head agree> = <X2 head agree>

  <X0 head> = <X2 head>

- X0 → X1 *and* X2

  <X1 cat> = <X2 cat>, <X0 cat> = <X1 cat>

- X0 → X1 X2

  <X1 orth> = *how*, <X2 sem> = <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
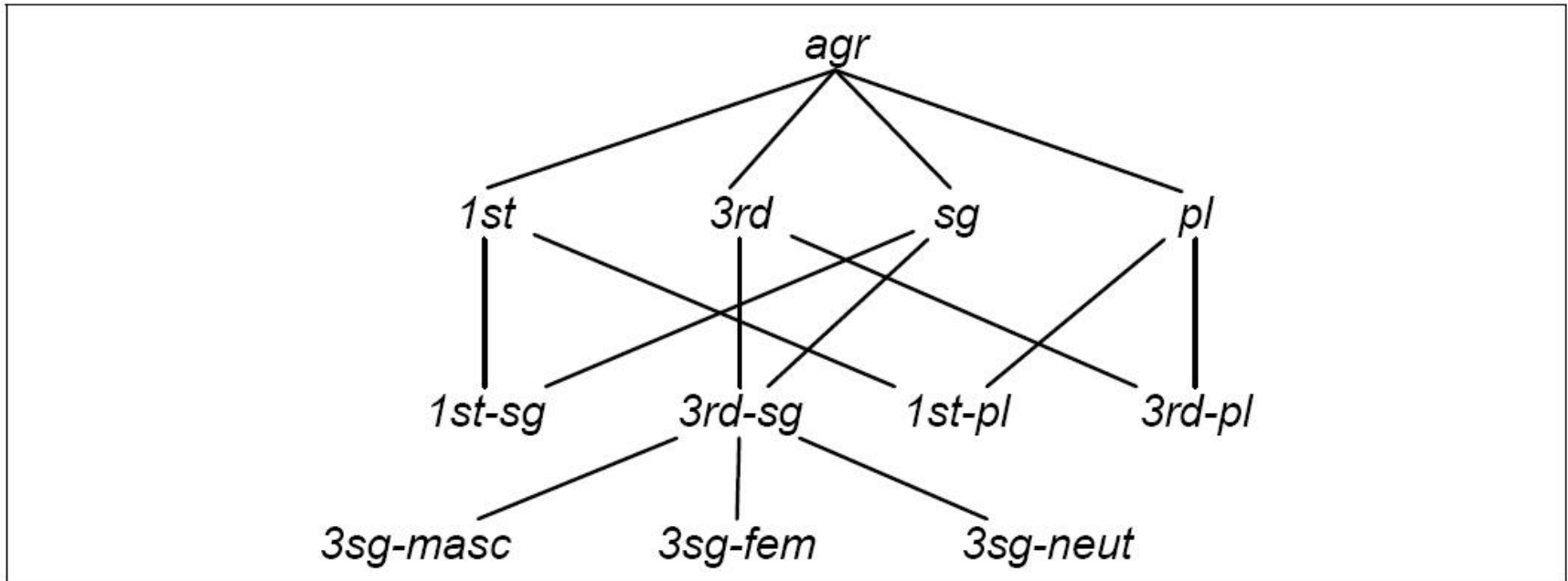
  FS1 ⊑ FS2   (Inverse is FS2 ***extends*** FS1)
- Subsumption relation forms a ***semilattice***,

  at the top: []

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

[number sg, person 3]
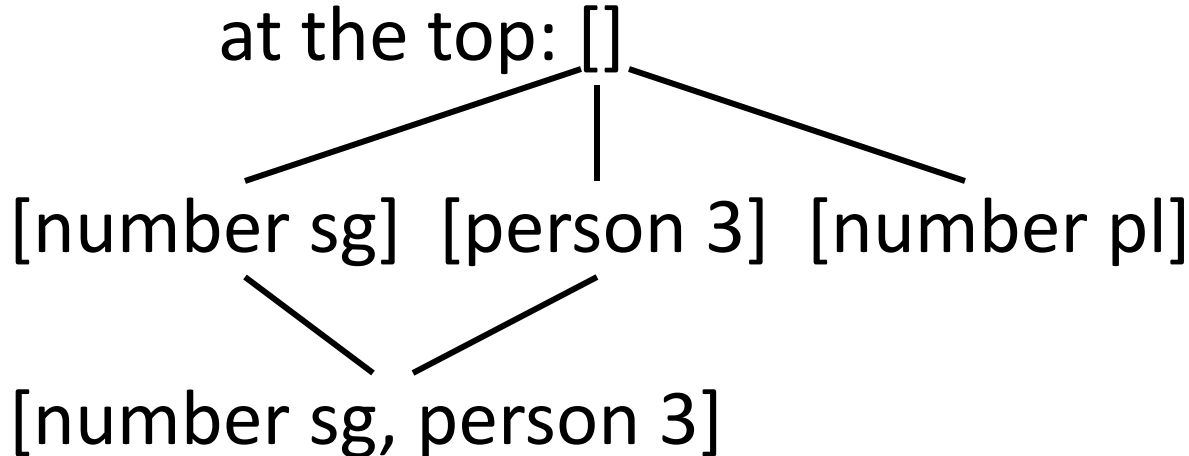
- Unification defined wrt semilattice:

  F ⊔ G = H s.t. F ⊑ H and G ⊑ H

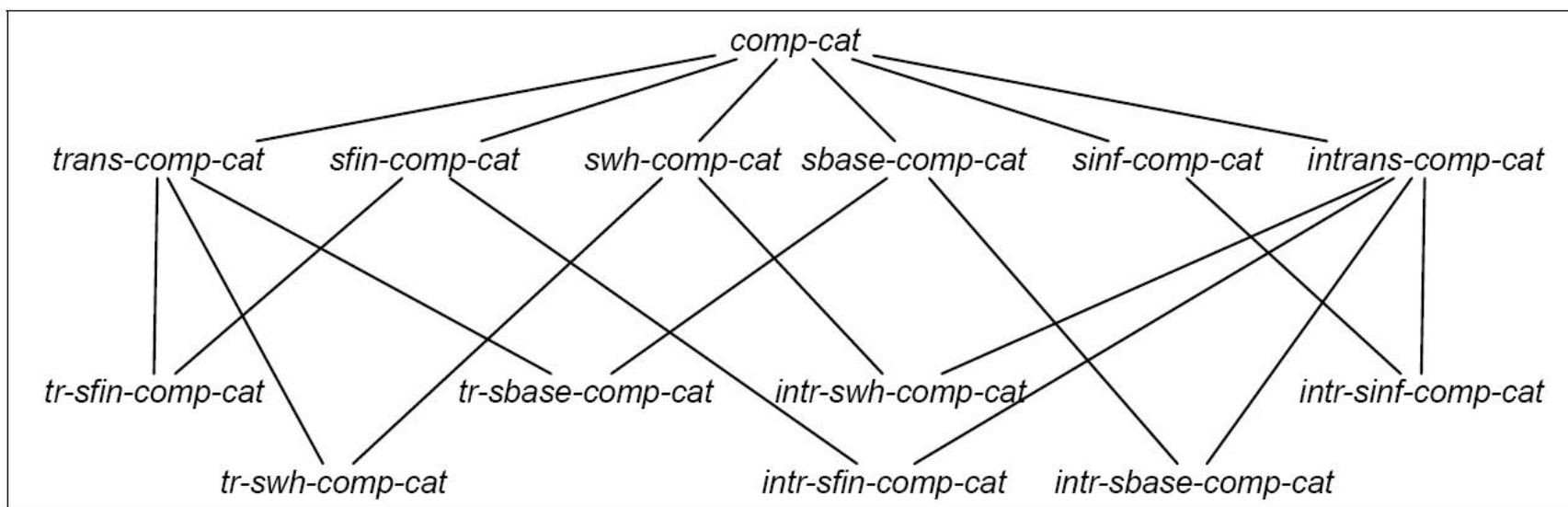 H is the Most General Unifier (MGU)

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** *is still much to learn*
It	nonreferential it	**It** *was evident that my ideas*
NP	noun phrase	*As he was relating* **his story**

Preposition Phrase Types		
PP	preposition phrase	*couch their message* **in terms**
PPing	gerundive PP	*censured him* **for not having intervened**
PPpart	particle	*turn it* **off**

Verb Phrase Types		
VPbrst	bare stem VP	*she could* **discuss it**
VPto	to-marked infin. VP	*Why do you want* **to know**?
VPwh	wh-VP	*it is worth considering* **how to write**
VPing	gerundive VP	*I would consider* **using it**

Complement Clause types		
Sfin	finite clause	*maintain* **that the situation was unsatisfactory**
Swh	wh-clause	*it tells us* **where we are**
Sif	whether/if clause	*ask* **whether Aristophanes is depicting a**
Sing	gerundive clause	*see* **some attention being given**
Sto	to-marked clause	*know* **themselves to be relatively unhealthy**
Sforto	for-to clause	*She was waiting* **for him to make some reply**
Sbrst	bare stem clause	*commanded* **that his sermons be published**

Other Types		
AjP	adjective phrase	*thought it* **possible**
Quo	quotes	*asked* **"What was it like?"**

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