6.863J Natural Language Processing
Lecture 9: Writing grammars; feature-based grammars

Robert C. Berwick
berwick@ai.mit.edu

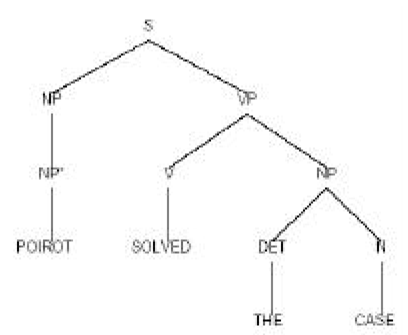
The Menu Bar

- Administrivia:
  - Schedule alert: Lab 3 out; due next Weds.
  - Lab time today, tomorrow
  - Please read notes3.pdf!! englishgrammar.pdf (on web)
- *Agenda:*
- Building grammars – basics to complex
- Limits of context-free grammars: the trouble with tribbles
- Foundation for the laboratory

Grammars for natural languages

- Where do the rules come from?
- Roughly: read them off of parse trees...
- A "rule-based", construction-based point of view
- Take 'surface' phrase patterns (mostly)
- But we still want to map to an *underlying* 'logical' form
- How do we start out?

Reading rules from parse trees...



S→NP VP
VP→V NP
NP→Det N
NP→ N*

Can't we get a computer to do this?

Key elements – part 1

- Establish basic phrase types: S, VP, NP, PP, ...
- Where do these come from???

What *kinds* of phrases are there?

- Noun phrases, verb phrases, adjectival phrases (“green with envy”), adverbial phrases (“quickly up the hill”), prepositional phrases (“off the wall”), etc.
- In general: *grounded* on lexical items
- Shows us the *constraints* on context-free rules for *natural grammars*
- Example:

Phrase types are constrained by lexical projection

Verb Phrase → Verb Noun Phrase

"is-a" ("kick the ball")

Prepositional Phrase → Preposition Noun Phrase

("on the table")

Adjective Phrase → Adjective Prep. Phrase

("green with envy")

Etc. … what is the pattern?

Function-argument relation

XP → X arguments, where X= Noun, Verb, Preposition, Adjective (all lexical categories in the language)

Like function-argument structure

(so-called "Xbar theory")

Constrains what grammar rules *cannot* be:

Verb Phrase → Noun Noun Phrase

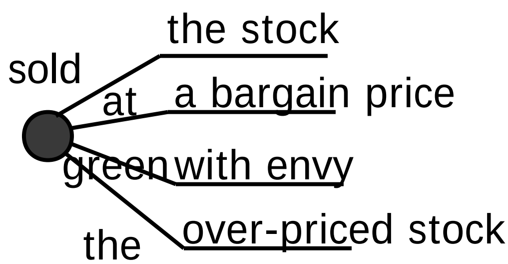
or even

Verb Phrase → Noun Phrase Verb Noun Phrase

English is function-argument form

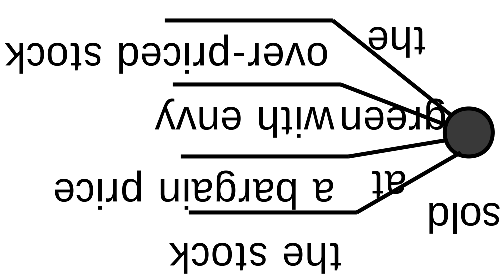
function

args



Other languages are the mirror-inverse: arg-function

This is like Japanese



Key elements – part 2

- Establish *verb subcategories*
- What are these?
  - Different verbs take different # arguments
  - 0, 1, 2 arguments ('complements')
  - Poirot thought; Poirot thought the gun; Poirot thought the gun was the cause.
  - Some verbs take certain sentence complements:
  - *I know who John saw/? I think who John saw* propositional types:
  - Embedded questions: *I wonder whether...*
  - Embedded proposition: *I think that John saw Mary*

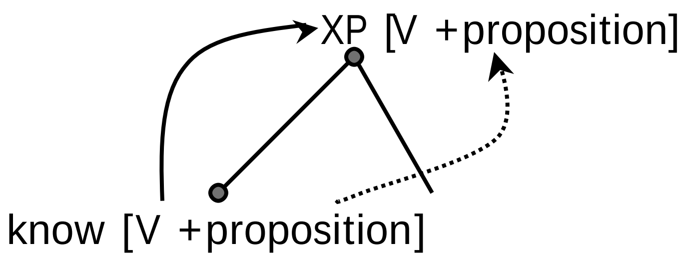
Key elements

- Subtlety to this
- Believe, know, think, wonder,...
  - ? I believe why John likes ice-cream
  - I know why John likes ice-cream
  - I believe that John likes ice-cream
  - I believe (that) John likes ice-cream

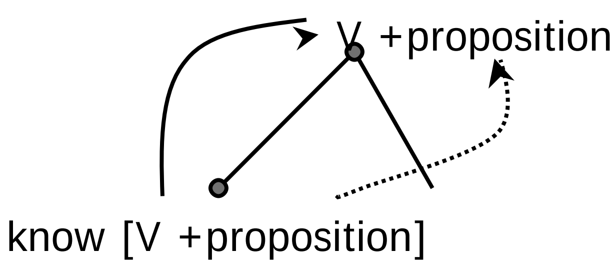
- # args, type: <u>Verb subcategories</u>
- How many subcategories are there?
- What is the structure?

Idea for phrases

- They are based on ‘projections’ of words (lexical items) – imagine features ‘percolating’ up

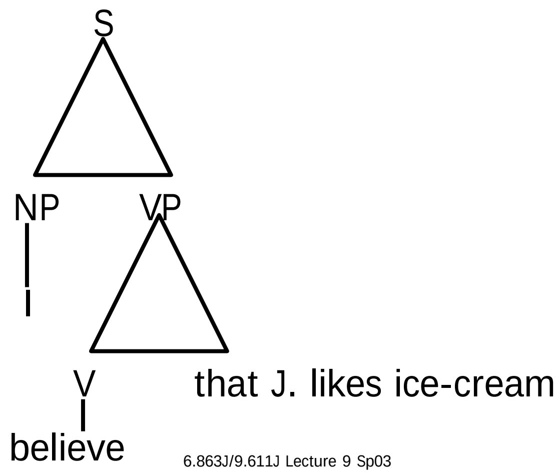


Heads of phrases

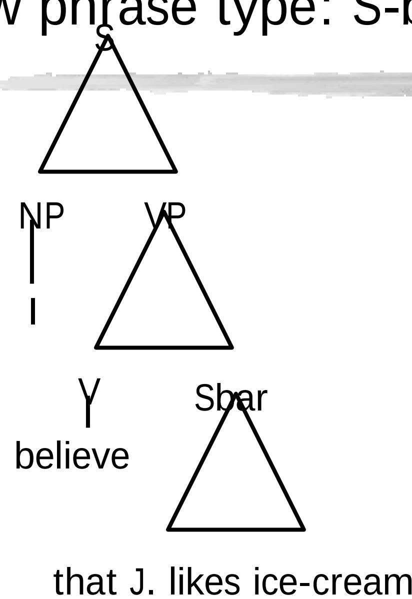


The parse structure for ‘embedded’ sentences

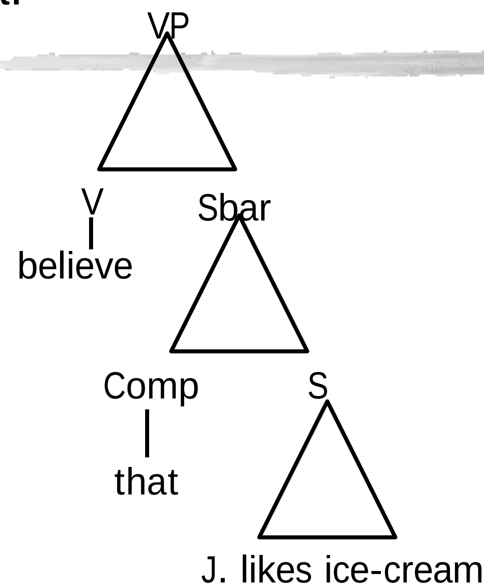
I believe (that) John likes ice-cream



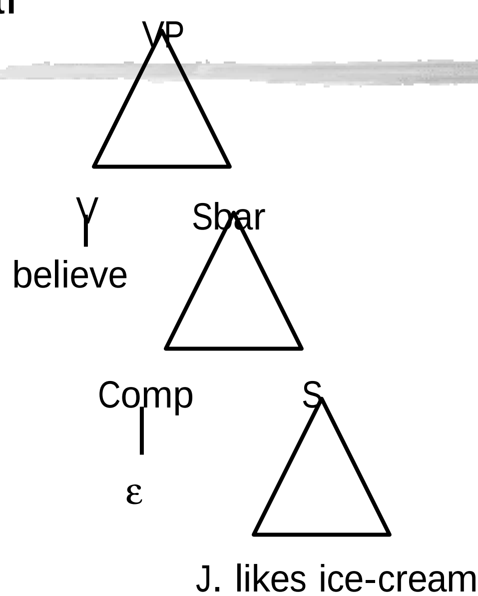
New phrase type: S-bar



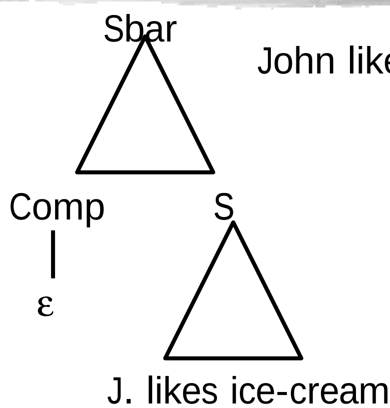
Sbar


VP
V
believe
Sbar
Comp
that
S
J. likes ice-cream

Sbar


VP
V
believe
Sbar
Comp
ε
S
J. likes ice-cream

In fact, true for all sentences...



John likes ice-cream

Why?

What rules will we need?

- (U do it..)

Verb types - continued

- What about:

  Clinton admires honesty/Honesty admires Clinton

How do we encode these in a CFG?

Should we encode them?

- Colorless green ideas sleep furiously
- Revolutionary new ideas appear infrequently

Features

The trouble with tribbles

morphology of a single word:

Verb[head=thrill, tense=present, num=sing, person=3,...] $\rightarrow$ thrills

projection of features up to a bigger phrase

VP[head=α, tense=β, num=γ...] $\rightarrow$ V[head=α, tense=β, num=γ...] NP

provided α is in the set TRANSITIVE-VERBS

agreement between sister phrases:

S[head=α, tense=β] $\rightarrow$ NP[num=γ,...] VP[head=α, tense=β, num=γ...]

provided α is in the set TRANSITIVE-VERBS

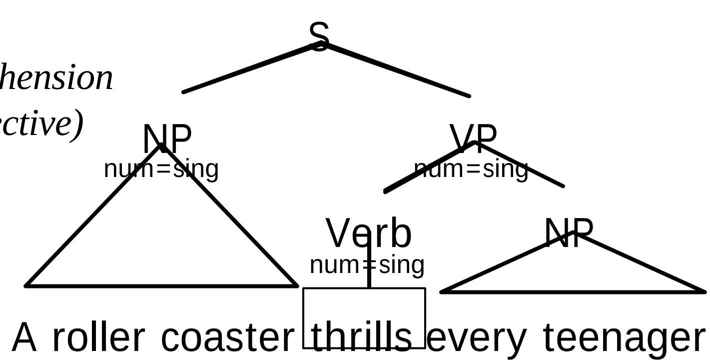
3 Common Ways to Use Features

Verb[head=thrill, tense=present, num=sing, person=3,...] $\rightarrow$ thrills

VP[head=α, tense=β, num=γ...] $\rightarrow$ V[head=α, tense=β, num=γ...] NP

S[head=α, tense=β] $\rightarrow$ NP[num=γ,...] VP[head=α, tense=β, num=γ...]

(comprehension perspective)



CFG Solution

- Encode constraints into the non-terminals
  - Noun/verb agreement

    S→ SgS

    S → PlS

    SgS → SgNP SgVP

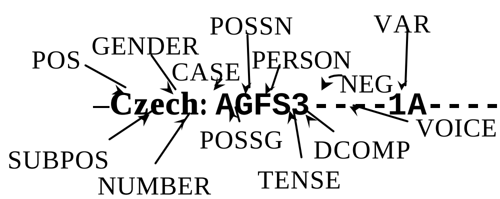
    SgNP → SgDet SgNom
  - Verb subcategories:

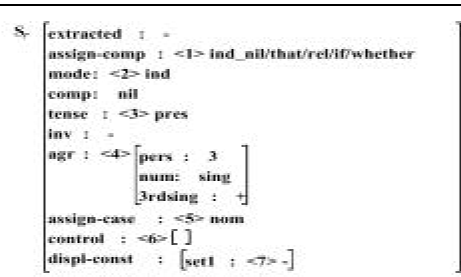
    IntransVP → IntransV

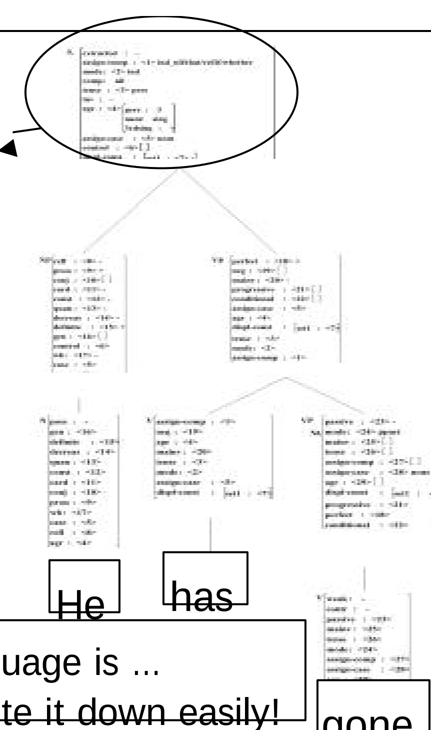
    TransVP → TransV NP

Problems with this – how much info?

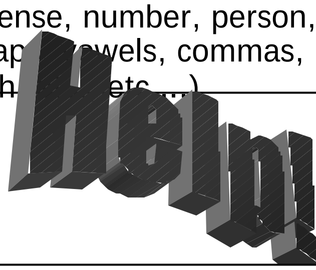
Agreement gets complex...







- Lots of features (tense, number, person, gaps, vowels, commas, wh, etc., ...)



- Sorry, that's just how language is ...
- You know too much to write it down easily!

Other sentence types

- Questions:
  - Will John eat ice-cream?
  - Did John eat ice-cream?
- How do we encode this?

`Empty’ elements or categories

- Where surface phrase is displaced from its canonical syntactic position
- Examples:
  - The ice-cream was eaten vs.
  - John ate the ice-cream
  - What did John eat?
  - What did Bill say that that John thought the cat ate?
  - For What x, did Bill say... the cat ate x
  - Bush is too stubborn to talk to
  - Bush is too stubborn [x to talk to Bush]
  - Bush is too stubborn to talk to the Pope
  - Bush is too stubborn [Bush to talk to the Pope]

More interesting clause types

- Apparently “long distance” effects: ‘displacement’ of phrases from their ‘base’ positions

1. So-called ‘wh-movement’:
   What did John eat ?
2. Topicalization (actually the same)
   On this day, it snowed two feet.
3. Other cases: so-called ‘passive’:
   The eggplant was eaten by John

- How to handle this?

We can think of this as ‘fillers’ and ‘gaps’

- Filler= the displaced item
- Gap = the place where it belongs, as argument
- Fillers can be NPs, PPs, S’s
- Gaps are *invisible*- so hard to parse! (we have to guess)
- Can be complex:
  Which book did you file__ without__ reading__ ?
  Which violins are these sonatas difficult to play__ on

Gaps ("deep" grammar!)

- Pretend "kiss" is a pure transitive verb.
- Is "the president kissed" grammatical?
  - If so, what type of phrase is it?

- the sandwich that
- I wonder what
- What else has

the president kissed e
Sally said the president kissed e
Sally consumed the pickle with e
Sally consumed e with the pickle

Gaps

- Object gaps:
- ~~the sandwich~~ that
- I wonder ~~what~~
- ~~What else~~ has

the president kissed e
Sally said the president kissed e
Sally consumed the pickle with e
Sally consumed e with the pickle

[how could you tell the difference?]

- Subject gaps:
- ~~the sandwich~~ that
- I wonder ~~what~~
- ~~What else~~ has

e kissed the president
Sally said e kissed the president

Gaps

- All gaps are really the same – a missing XP:
- the sandwich that
- I wonder what
- What else has

the president kissed e
Sally said the president kissed e
Sally consumed the pickle with e
Sally consumed e with the pickle
e kissed the president
Sally said e kissed the president

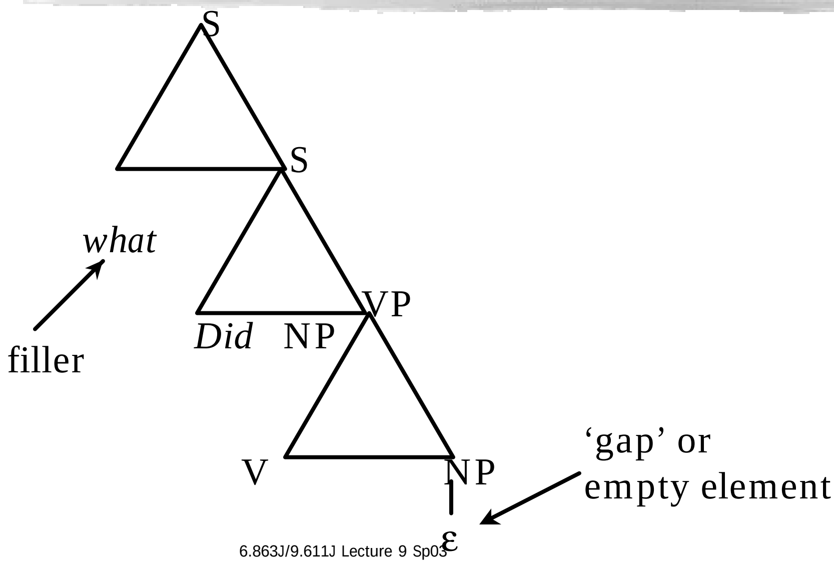
Phrases with missing NP:
X[missing=NP]
or just X/NP for short

Representation & computation questions again

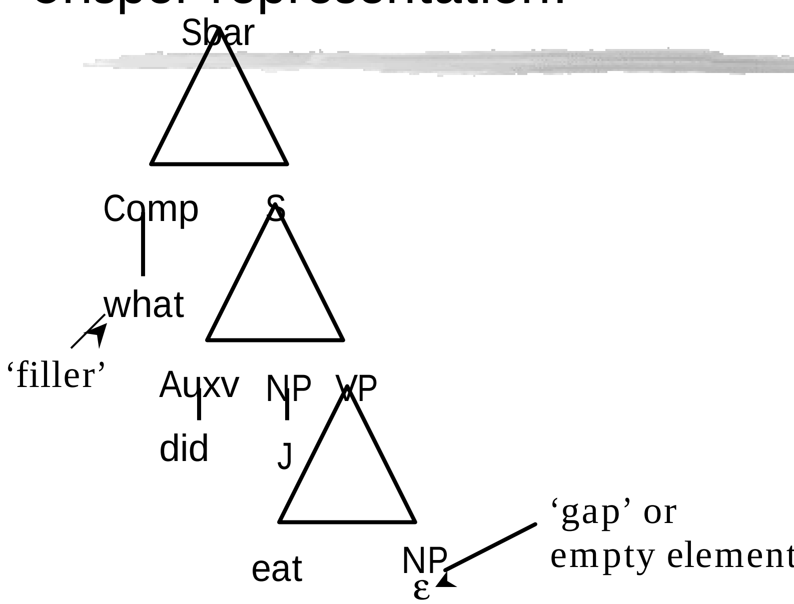
- How do we *represent* this displacement? (difference between underlying & surface forms)
- How do we *compute* it? (I.e., parse sentences that exhibit it)
- We want to recover the *underlying* structural relationship because this tells us what the predicate-argument relations are – *Who did what to whom*
- Example: *What did John eat* → For which x, x a thing, did John eat x?
- Note how the eat-x predicate-argument is established

Representations with gaps

- Let's first look at a tree with gaps:



Crisper representation:



Fillers can be arbitrarily far from gaps they match with...

- What did John say that Mary thought that the cat ate___?

Fillers and gaps

- Since ‘gap’ is NP going to empty string, we could just add rule, $NP \rightarrow \varepsilon$
- But this will *overgenerate* why?
- We need a way to distinguish between
  - What did John eat
  - Did John eat
- How did this work in the FSA case?

So, what do we need

- A rule to expand NP as the empty symbol; that's easy enough: NP$\rightarrow\varepsilon$
- A way to make sure that NP is expanded as empty symbol iff there is a gap (in the right place) before/after it
- A way to link the filler and the gap
- We can do all this by futzing with the nonterminal names: *Generalized Phrase Structure Grammar (GPSG)*

Still other 'missing' elements

- John promised Mary ___ to leave
- John promised Mary [John to leave]
- Known as 'control'

- John persuaded Mary [___ to leave]
- John persuaded Mary [Mary to leave]

Limits of CFGs

- Agreement (A cat sleeps. Cats sleep.)

S → NP VP

NP → Det Nom

But these rules overgenerate, allowing, e.g., *A cat sleep...

- Subcategorization (Cats dream. Cats eat cantaloupe.)

VP → V

VP → V NP

But these also allow *Cats dream cantaloupe.

- We need to constrain the grammar rules to enforce e.g. number agreement and subcategorization differences
- We'll do this with feature structures and the constraint-based unification formalism

CFG Solution

- Encode constraints into the non-terminals
  - Noun/verb agreement

    S→ SgS

    S → PlS

    SgS → SgNP SgVP

    SgNP → SgDet SgNom
  - Verb subcat:

    IntransVP → IntransV

    TransVP → TransV NP

- But this means huge proliferation of rules...
- An alternative:
  - View terminals and non-terminals as complex objects with associated features, which take on different values
  - Write grammar rules whose application is constrained by tests on these features, e.g.

    S → NP VP (only if the NP and VP agree in number)

Design advantage

- Decouple skeleton syntactic structure from lexicon
- We'll explore later, for now...

Feature Structures

- Sets of feature-value pairs where:
  - Features are atomic symbols
  - Values are atomic symbols or feature structures
  - Illustrated by attribute-value matrix

$$\begin{bmatrix} Feature_1 & Value_1 \\ Feature_2 & Value_2 \\ \ldots & \ldots. \\ Feature_n & Value_n \end{bmatrix}$$

- Number feature

$$\begin{bmatrix} Num & SG \end{bmatrix}$$

- Number-person features

$$\begin{bmatrix} Num & SG \\ Pers & 3 \end{bmatrix}$$

- Number-person-category features (3sgNP)

$$\begin{bmatrix} Cat & NP \\ Num & SG \\ Pers & 3 \end{bmatrix}$$

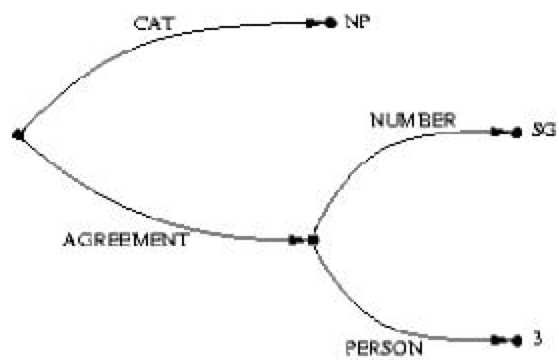
  - How do we define 3plNP?
  - How does this improve over the CFG solution?
- Feature values can be feature structures themselves
  - Useful when certain features commonly co-occur, e.g. number and person

$$\begin{bmatrix} Cat & NP \\ Agr & \begin{bmatrix} Num & SG \\ Pers & 3 \end{bmatrix} \end{bmatrix}$$

  - Feature path: path through structures to value (e.g.

    Agr → Num → SG

Graphical Notation for Feature Structures



Reentrant Structures

- Feature structures may also contain features that share some feature structure as a value

$$\begin{bmatrix} Cat & S \\ Head & \begin{bmatrix} Agr & 1 \begin{bmatrix} Num & SG \\ Pers & 3 \end{bmatrix} \\ Subj & \begin{bmatrix} Agr & 1 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- Numerical indices indicate the shared values

Operations on Feature Structures

- What will we need to do to these structures?
  - Check the compatibility of two structures
  - Merge the information in two structures
- We can do both using unification
- We say that two feature structures can be unified if the component features that make them up are compatible
  - [Num SG] U [Num SG] = [Num SG]
  - [Num SG] U [Num PL] fails!
  - [Num SG] U [Num []] = [Num SG]

- [Num SG] U [Pers 3] = $\begin{bmatrix} Num & SG \\ Pers & 3 \end{bmatrix}$
- Structure are compatible if they contain no features that are **in**compatible
- Unification of two feature structures:
  - Are the structures compatible?
  - If so, return the union of all feature/value pairs
- A failed unification attempt

$$\begin{bmatrix} Agr & 1\begin{bmatrix} Num & SG \\ Pers & 3 \end{bmatrix} \\ Subj & \begin{bmatrix} Agr & 1 \end{bmatrix} \end{bmatrix} \;U\; \begin{bmatrix} Agr & \begin{bmatrix} Num & Pl \\ Pers & 3 \end{bmatrix} \\ Subj & \begin{bmatrix} Agr & \begin{bmatrix} Num & PL \\ Pers & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Features, Unification and Grammars

- How do we incorporate feature structures into our grammars?
  - Assume that constituents are objects which have feature-structures associated with them
  - Associate sets of unification constraints with grammar rules
  - Constraints must be satisfied for rule to be satisfied
- For a grammar rule $\beta_0 \rightarrow \beta_1 \ldots \beta_n$
  - <β_i feature path> = Atomic value
  - <β_i feature path> = <β_j feature path>

- To enforce subject/verb number agreement

  S → NP VP

  <NP NUM> = <VP NUM>

Agreement in English

- We need to add PERS to our subj/verb agreement constraint

This cat likes kibble.

S → NP Vp

<NP AGR> = <VP AGR>

Do these cats like kibble?

S → Aux NP VP

<Aux AGR> = <NP AGR>

- Det/Nom agreement can be handled similarly

These cats

This cat

NP → Det Nom

<Det AGR> = <Nom AGR>

<NP AGR> = <Nom AGR>

- And so on for other constituents and rules

Head Features

- Features of most grammatical categories are copied from head child to parent (e.g. from V to VP, Nom to NP, N to Nom, ...)
- These normally written as 'head' features, e.g.

  VP → V NP

  <VP HEAD> = <V HEAD>

  NP → Det Nom

  <NP→ HEAD> = <Nom HEAD>

  <Det HEAD AGR> = <Nom HEAD AGR>

  Nom → N

  <Nom HEAD> = <N HEAD>

Subcategorization

- Recall: Different verbs take different types of argument
  - Solution: SUBCAT feature, or subcategorization frames

e.g.

$$\begin{bmatrix} ORTH & want \\ CAT & V \\ HEAD & \begin{bmatrix} SUBCAT & \left\langle \begin{bmatrix} CAT & NP \end{bmatrix}, \begin{bmatrix} CAT & VP \\ HEAD & \begin{bmatrix} VFORM & INF \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

- But there are many phrasal types and so many types of subcategorization frames, e.g.
  - believe
  - believe [VPrep in] [NP ghosts]
  - believe [NP my mother]
  - believe [Sfin that I will pass this test]
  - believe [Swh what I see] ...
- Verbs also subcategorize for subject as well as object types ([$_{Swh}$ What she wanted] seemed clear.)
- And other p.o.s. can be seen as subcategorizing for various arguments, such as prepositions, nouns and adjectives (It was clear [Sfin that she was exhausted])

- NB: p.o.s. that subcategorize similarly define rough classes e.g. verb categories like transfer verbs and subcat frame relationships within verb classes are called alternations
  - George gave Martha a letter [NP NP]
  - George gave a letter to Martha [NP PP]

Long-Distance Dependencies

- What happens when a verb's arguments are not in the VP?
  - What meals does the restaurant serve?

  Wh-NP fills a slot in serve

  S --> wh-NP Aux NP VP
- How to solve?
  - Gap list: GAP feature (filler: what meals) passed up from phrase to phrase in parse tree -- complicated mechanism
  - Even bigger problem for representations such as FSAs and Ngrams

How can we parse with feature structures?

- Unification operator: takes 2 features structures and returns ***either*** a merged feature structure or *fail*
- Input structures represented as DAGs
  - Features are labels on edges
  - Values are atomic symbols or DAGs
- Unification algorithm goes through features in one input DAG_1 trying to find corresponding features in DAT_2 – if all match, success, else fail

Unification and Chart Parsing

- Goal:
  - Use feature structures to provide richer representation
  - Block entry into chart of ill-formed constituents
- Changes needed to Earley
  - Add feature structures to grammar rules, e.g.

    S → NP VP

    <NP HEAD AGR> = <VP HEAD AGR>

    <S HEAD> = <VP HEAD>
  - Add field to states containing DAG representing feature structure corresponding to state of parse, e.g.

    S → • NP VP, [0,0], [], DAG

- Add new test to Completer operation
  - Recall: Completer adds new states to chart by finding states whose • can be advanced (i.e., category of next constituent matches that of completed constituent)
  - Now: Completer will only advance those states if their feature structures unify
- New test for whether to enter a state in the chart
  - Now DAGs may differ, so check must be more complex
  - Don’t add states that have DAGs that are more specific than states in chart: is new state subsumed by existing states?

Summing Up

- Feature structures encoded rich information about components of grammar rules
- Unification provides a mechanism for merging structures and for comparing them
- Feature structures can be quite complex:
  - Subcategorization constraints
  - Long-distance dependencies
- Unification parsing:
  - Merge or fail
  - Modifying Earley to do unification parsing