The Menu Bar

- Administrivia:
  - Lab 3b out later today - Weds; due after vacation – April 5

Agenda:
  Fillers & Gaps; I shrank the grammar!
  Features & feature grammars
Job 1: writing grammar rules

Three sorts of examples to handle:

1. Simple declarative sentences
   - Poirot solved the case
   - Poirot thought
   - Poirot sent the solution to the police
   - Poirot believed the detectives were incompetent

2. Auxiliary verb sentences
   - P. may have been solving the case

3. Unbounded dependencies: Questions and relative clauses
   - Which case did Poirot solve
   - The solution that P. sent to the police solved the case

Want to block:

- do not overgenerate
  *Poirot solved; *P. may solved the case;
  *Which solution did P. send which solution to the police?
Preliminaries: phrase names

- I said that ice-cream was on the table
  *I* said *ice-cream* was on the table
- What is the structure here?
- Existence of *Complementizer* (*COMP*) before _Sentence phrase, forming an “Sbar phrase”

(S):

\[
\begin{array}{c}
\text{Sbar} \\
\text{Sbar} \\
\rightarrow (\text{Comp}) \text{ S}
\end{array}
\]

- The *Comp* item can be *that*, *which*, ... or a displaced phrase
- Also present in ‘root’ (top level) sentences (*I like ice-cream*) but usually we don’t ‘hear’ it (unless it’s filled by question or focus phrase)
- Serves as ‘landing site’ for fillers
- In embedded sentences, in English, the *Comp* is optional
- If *Comp* is filled – then that blocks things:
  Who (F) do I know that John likes (G)

Sbar = Comp S
Filler-gap examples

<table>
<thead>
<tr>
<th>Example</th>
<th>Ordinary</th>
<th>Filler-gap analog</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wh-question</td>
<td>Mary saw Bill</td>
<td>Who (F) did Mary see (G)?</td>
</tr>
<tr>
<td>Topicalization</td>
<td>John hates beans</td>
<td>Beans (F) John hates (G)</td>
</tr>
<tr>
<td>Tough-movement</td>
<td>It is hard to please John</td>
<td>John (F) is hard to please (G)</td>
</tr>
<tr>
<td>Relative clauses</td>
<td>John likes the guy</td>
<td>The guy (F) that John likes (G)</td>
</tr>
</tbody>
</table>

Fillers and gaps, redux

- Fillers and Gaps summary: F, G
- Filler is the *displaced phrase*
- Gap is a *phonological null* (unpronounced) *empty category* (though it can have secondary phonological consequences:
  - *This student (F) you want (G) to solve the problem* blocks contraction between *want* and *to* into *wanna*
  - *? This student you wanna solve the problem*
- F-G relation represents displacement from canonical semantic argument position
- Many examples of this in natural language
Fillers and gaps

- Since ‘gap’ is NP going to empty string, we could just add rule, \( NP \rightarrow \varepsilon \)
- But this will *overgenerate* how?
- 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 rules: *Generalized Phrase Structure Grammar (GPSG)*
“State-splitting” to remember what seen (but not heard) – need new states (cf. vowel harmony, etc.)

- We could encode the two possible routes by distinct chains of states, as follows:

```
ε → NP → V → NP
          what
```

- Names not very enlightening, so will use this instead:

```
ε → NP → V → NP
```

So we have to add rules with new nonterminals to ‘name’ states...

- $S/NP \rightarrow NP\ VP/NP$
- $VP/NP \rightarrow V\ NP/NP$
- $NP/NP \rightarrow \epsilon$
- We haven’t put the auxiliary verb stuff in...
- Note the ‘chain’ of slashed rules in the final structure
- What happens computationally?
Actual ‘marks’ in the literature

- Called a ‘slash category’
- Ordinary category: Sbar, VP, NP
- Slash category: Sbar/NP, VP/NP, NP/NP
- “X/Y” is ONE atomic nonterminal
- Interpret as: Subtree X is missing a Y (expanded as e) underneath
- Example: Sbar/NP = Sbar missing NP underneath (see our example)

As for slash rules...

- We need slash category introduction rule, e.g., Sbar → Comp S/NP
- We need ‘elimination’ rule NP/NP → e
- These are paired (why?)
- We’ll need other slash categories, e.g.,
How do we do 'slashed rules' systematically & formally

Step 1: form slashed categories
- Basic categories: 'original' nonterminals $N = \{ S, VP, NP, PP, ... \}$
- Slashed (derived) categories: $\alpha/\beta$, $\alpha$, $\beta$ range over $N$
- E.g., $S/NP$, $S/VP$, $S/PP$, $NP/NP$, $NP/VP$, $NP/PP$
- Interpretation: tree rooted at $\alpha$, with 'gap' (subtree) of type $\beta$ somewhere beneath

Step 2: form slashed rules from basic rules
- Basic rule: $S \to NP \ VP$
- Slashed rule: $S/NP \to NP \ VP/NP$
- (Why not $S/NP \to NP/NP \ VP$)
Also have ‘subject’ gaps

Filler-gap configuration

- Equivalent to notion of ‘scope’ for natural languages (scope of variables) \( \approx \) Environment frame in Scheme/binding environment for ‘variables’ that are empty categories
- Formally: Fillers c-command gaps (constituent command)
- Definition of c-command:
Constraints on filler-gap relations

- Can be “unbounded” on the surface, but underlyingly is *successive cyclic* (AKA – *forms a chain* linking filler to gap)
  - [what (F) did John think (F) that Bill said (F) that Mary liked (G)]
- Note that this F-G distance cannot exceed 1 adjacent S or NP boundary (in English)
  - What (F) [ do you wonder [who likes (G)]
  - (Note: *What (F) do you wonder (F) who likes (G)* is blocked)

Constraints on filler-gaps

- Obeys structural relation called *c-command*
- True in other languages also; also true there are multiple gaps
- So, how does *generalized phrase structure grammar (GPSG)* handle all this?
- We covered: basic rules; *derived rules*
- Still to cover: *metarules; constraints*
Filler-gap configuration

A phrase $\alpha$ c-commands a phrase $\beta$ iff the first branching node that dominates $\alpha$ also dominates $\beta$ (blue = filler, green = gap).
Natural for $\lambda$ abstraction

Constraints on gpsg rules

• “Across the board” constraints in conjunction

The person who Mary likes (S/NP) and Sally hates George(S) computed my tax. Compare:
The person who Mary likes (S/NP) and Sally hates (S/NP) computed my tax

Can’t join S/NP and S – different categories, akin to
John likes pizza and beer (NP and NP)

• Extracted wh-phrase must be of same type

Which book and which pencil did John buy?
? John asked who and where Bill had seen (G)
More constraints

- English specific
  *Who (F) do you believe (G) that came*
  *Who (F) did you wonder whether (G) came*
  *Who (F) did you wonder if (G) came*
  OK: *Who is it that Mary likes*

- What is going on here?

Rightward displacement

- *Harry caught, and Mary killed, the rabid dog*
  - The man (G) was ill who was here (F)
  - John hummed (G) and Mary sang (G), **at equal volumes (F)**
  - Again can’t be dissimilar
    - *John offered and Harry gave Bill a Volvo (Bill a Volvo” isn’t a phrase)*
  - Again can’t be “too far”:
    - *Harry fished in the ocean and I don’t think Mary in the sea.*
Some examples to help with lab – corresponding tree structures (I won’t leave you at bay in a sea of nonterminals)

- If you start w/ tree structures, the CF rules write themselves (almost)
- *Fido chased Mary*

Structure for this sentence

```
  S
 / \  
NP VP+tns
     
Name V__+tns NP
   
Fido chased Mary
```
Why do we need

- V2?
- V2+tns?

Verb subcategories

- You will need V1, V2, V3, V4,...
Now easy to read off rules from trees

- $S \rightarrow NP \ VP+TNS$
- $VP+TNS \rightarrow V2+TNS \ NP$

Etc...

Idea 1: Wysiwyg

Root

Question

NP+wh

Who

saw

Pronp+wh

Name

Mary

V2+tns

V2+tns

NP
Idea 2: conform to wh-pattern of others, e.g., “What did John see”

More complex syntax – simpler semantics (canonical)
First alternative

- Syntactic structures ‘closer to the surface’
  - Then we have to figure out semantic differences from hacking the semantic part
  - In fact, this is what GPSG does for so-called ‘passive’ also – it doesn’t ‘encode’ this in a change from active sentence to passive sentence, e.g., John ate the ice-cream → The ice-cream was eaten (by John)
  - Instead, it just has two forms. Is this right?
  - Which form is ‘primary’? (more fundamental)
  - Evidence: doesn’t seem to be cases where you have a passive form without the corresponding active form, but does seem to be cases the other way around (active but no passive)

Now, what if we move the object?

Mary caught e John killed e the rabid dog
Another example

\[ S \rightarrow S \text{bar} \quad \text{Conj} \quad S \text{bar} \]

\[ S \text{bar} \quad S \text{bar} \]

\[ \text{Mary caught the rabid dog} \quad \text{and} \quad \text{John killed the rabid dog} \]

Besides reading off the rules...

- Why can’t we just build a machine to do this?
- We could induce rules from the structures
- But we have to know the right representations (structures) to begin with
- Penn treebank has structures – so could use learning program for that
- This is, as noted, a construction based approach
- We have to account for various constraints, as noted
Constraints – and language variation

Examples:

‘Distance’ effects: What do you wonder who likes
(English): must have a subject (unlike Spanish) I came; vs. came

- How do we want to account for these?
- 2 possible ways
  1. More engineering: make a list
  2. More scientific: look for deeper theory that has primitives that only lets you ‘write the correct’ rules – like automatic program construction

What if we move the object?

Mary caught e and John killed e

the rabid dog
Why not read off the rules?

- Why can’t we just build a machine to do this?
- We could induce rules from the structures
- But we have to know the right representations (structures) to begin with
- Penn treebank has structures – so could use learning program for that
- This is, as noted, a construction based approach
- We have to account for various constraints, as noted

So what?

- What about multiple fillers and gaps?
- Which violins are these sonatas difficult to play _____ on which ______?
How many context-free rules?

- For every displaced phrase, what do we do to the ‘regular’ context-free rules?
- How many kinds of displaced rules are there?
  - Which book and Which pencil did Mary buy?
    - *Mary asked who and what bought
- Well, how many???

And then..

- John saw more horses than bill saw cows or Mary talked to
- John saw more horses than bill saw cows or mary talked to cats
- The kennel which Mary made and Fido sleeps in has been stolen
- The kennel which Mary made and Fido sleeps has been stolen
How big can the grammar get???

- John sleeps
- They sleep
- I know her
- ?I know she
- Agreement features
- Quite systematic

Other languages; formalizing features

- Two kinds:
  1. Syntactic features, purely grammatical function
     Example: Case in German (NOMinative, ACCusative, DATive case) – relative pronoun must agree w/ Case of verb with which it is construed
     \[ Wer \text{ nicht} \text{ stark} \text{ is}, \text{ muss} \text{ klug} \text{ sein} \]
     \[ Who \text{ not} \text{ strong} \text{ is}, \text{ must} \text{ clever} \text{ be} \]
     NOM       NOM
     Who isn't strong must be clever
Continuing this example

\[ \text{Ich nehme, wen } du mir empfehlst } \]
\[ I \text{ take whomever you me recommend } \]
\[ \text{ACC } \text{ACC } \text{ACC} \]
\[ I \text{ take whomever you recommend to me} \]

\[ *\text{Ich nehme, wen } du vertraust \]
\[ I \text{ take whomever you trust} \]
\[ \text{ACC } \text{ACC } \text{DAT} \]

Other class of features

2. Syntactic features w/ meaning – example, number, def/indef., adjective degree

Hungarian

\[ \text{Akart egy könyvet } \]
\[ \text{He-wanted a} \text{ book} \]
\[ \text{-DEF -DEF} \]
\[ \text{egy könyv amit akart} \]
\[ \text{A book which he-wanted} \]
\[ \text{-DEF -DEF} \]
The trouble with tribbles

morphology of a single word:
Verb[head=thrill, tense=present, num=sing, person=3,...] → thrills

projection of features up to a bigger phrase
VP[head=α, tense=β, num=γ,...] → 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] → NP[num=γ,...] VP[head=α, tense=β, num=γ,...]
provided α is in the set TRANSITIVE-VERBS

3 common ways to use features

morphology of a single word:
Verb[head=thrill, tense=present, num=sing, person=3,...] → thrills

projection of features up to a bigger phrase
VP[head=α, tense=β, num=γ,...] → 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] → NP[num=γ,...] VP[head=α, tense=β, num=γ,...]
provided α is in the set TRANSITIVE-VERBS
A roller coaster thrills every teenager

3 Common Ways to Use Features

Verb[head=thrill, tense=present, num=sing, person=3,...] → thrills
VP[head=α, tense=β, num=γ,...] → V[head=α, tense=β, num=γ,...] NP
S[head=α, tense=β] → NP[head=γ,...] VP[head=α, tense=β, num=γ,...]

(generation perspective)

A roller coaster thrills every teenager

3 Common Ways to Use Features

Verb[head=thrill, tense=present, num=sing, person=3,...] → thrills
VP[head=α, tense=β, num=γ,...] → V[head=α, tense=β, num=γ,...] NP
S[head=α, tense=β] → NP[head=γ,...] VP[head=α, tense=β, num=γ,...]

(comprehension perspective)
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.
  \[ S \rightarrow NP \ VP \text{ (only if the NP and VP agree in number)} \]

Design advantage

- Decouple skeleton syntactic structure from lexicon
- In fact, the syntactic structure really is a skeleton:
From this...

To this
Features are everywhere

morphology of a single word:
Verb[head=thrill, tense=present, num=sing, person=3,...] \rightarrow \text{thrills}

projection of features up to a bigger phrase
VP[head=\alpha, tense=\beta, num=\gamma,...] \rightarrow V[head=\alpha, tense=\beta, num=\gamma,...] \text{ NP}
provided \alpha is in the set TRANSITIVE-VERBS

agreement between sister phrases:
S[head=\alpha, tense=\beta] \rightarrow \text{NP}[num=\gamma,...] \text{ VP}[head=\alpha, tense=\beta, num=\gamma,...]
provided \alpha is in the set TRANSITIVE-VERBS

Better approach to factoring linguistic knowledge

• Use the superposition idea: we superimpose one set of constraints on top of another:
  1. Basic skeleton tree
  2. Plus the added feature constraints
• \begin{align*}
S & \rightarrow \text{NP} \quad \text{VP} \\
[\text{num x}] & \quad [\text{num x}] \\
the \text{ guy} & \quad \text{eats} \\
[\text{num singular}] & \quad [\text{num singular}]
\end{align*}
Or in tree form:

Values trickle up
Checking features


the [number singular] guy [number singular] eats [number singular]

What sort of power do we need here?

- We have [feature value] combinations so far
- This seems fairly widespread in language
  - We call these atomic feature-value combinations
  - Other examples:
    1. In English:
       person feature (1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup>);
       Case feature (degenerate in English: nominative, object/accusative, possessive/genitive): I know her vs.
       I know she;
       Number feature: plural/sing; definite/indefinite
       Degree: comparative/superlative
Feature Structures

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

How to formalize?

- Let $F$ be a finite set of feature names, let $A$ be a set of feature values
- Let $p$ be a function from feature names to permissible feature values, that is, $p: F \rightarrow 2^A$
- Now we can define a word category as a triple $<F, A, p>$
- This is a partial function from feature names to feature values
Example

\[ F = \{ \text{CAT, PLU, PER} \} \]

- \( p \):
  - \( p(\text{CAT}) = \{ V, N, \text{ADJ} \} \)
  - \( p(\text{PER}) = \{ 1, 2, 3 \} \)
  - \( p(\text{PLU}) = \{ +, - \} \)

\[
\text{sleep} = \{ [\text{CAT } V], [\text{PLU } -], [\text{PER } 1] \} \\
\text{sleep} = \{ [\text{CAT } V], [\text{PLU } +], [\text{PER } 1] \} \\
\text{sleeps}= \{ [\text{CAT } V], [\text{PLU } -], [\text{PER } 3] \}
\]

Checking whether features are compatible is relatively simple here...how bad can it get?

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
  - \([\text{Num SG}] \cup [\text{Num SG}] = [\text{Num SG}]\)
  - \([\text{Num SG}] \cup [\text{Num PL}] \) fails!
  - \([\text{Num SG}] \cup [\text{Num []}] = [\text{Num SG}]\)
• [Num SG] U [Pers 3] =

• Structures are **compatible** if they contain no features that are incompatible.

• Unification of two feature structures:
  • Are the structures compatible?
  • If so, return the union of all feature/value pairs
  • A failed unification attempt

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 $\beta_0 \rightarrow \beta_1 ... \beta_n$
  • $<\beta_i \text{ feature path}> = \text{ Atomic value}$
  • $<\beta_i \text{ feature path}> = <\beta_j \text{ feature path}>$
• NB: if simple feat-val pairs, no arbitrary nesting, then no need for paths
Feature unification examples

1) [ agreement: [ number: singular
      person: first ] ]

2) [ agreement: [ number: singular]
      case: nominative ]

• (1) and (2) can unify, producing (3):

3) [ agreement: [ number: singular
      person: first ]
      case: nominative ]

(try overlapping the graph structures corresponding to these two)

• (2) & (4) can unify, yielding (5):

4) [ agreement: [ number: singular
      person: third ] ]

5) [ agreement: [ number: singular
      person: third ]
      case: nominative ]

• BUT (1) and (4) cannot unify because their values conflict on <agreement person>
To enforce subject/verb number agreement

\[ S \rightarrow NP \ VP \]
\[ <NP \ NUM> = <VP \ NUM> \]

**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.
  \[ VP \rightarrow V \ NP \]
  \[ <VP \ HEAD> = <V \ HEAD> \]
  \[ NP \rightarrow Det \ Nom \]
  \[ <NP \rightarrow HEAD> = <Nom \ HEAD> \]
  \[ <Det \ HEAD \ AGR> = <Nom \ HEAD \ AGR> \]
  \[ Nom \rightarrow N \]
  \[ <Nom \ HEAD> = <N \ HEAD> \]
The plan to swallow Wanda has been thrilling for Otto.
The plan to swallow Wanda has been thrilling Otto.

**Why use heads?**

- Morphology (e.g., word endings)
- $N[h=plan,n=1] \rightarrow plan$
- $N[h=plan,n=2+] \rightarrow plans$
- $V[h=thrill,tense=prog] \rightarrow thrilling$
- $V[h=thrill,tense=past] \rightarrow thrilled$
- $V[h=go,tense=past] \rightarrow went$
- Why use heads?

- Subcategorization (i.e., transitive vs. intransitive)
- When is VP $\rightarrow$ V NP ok?
  $VP[h=\alpha] \rightarrow V[h=\alpha] NP$
  restrict to $\alpha \in$ TRANSITIVE VERBS
- When is N $\rightarrow$ N VP ok?
  $N[h=\alpha] \rightarrow N[h=\alpha] VP$
  restrict to $\alpha \in \{\text{plan, plot, hope,\ldots}\}$

- Selectional restrictions
- I.e., $VP[h=\alpha] \rightarrow V[h=\alpha] NP[h=\beta]$
- Don't fill template in all ways:
  $VP[h=\text{thrill}] \rightarrow V[h=\text{thrill}] NP[h=\text{plan}]$
  * $VP[h=\text{thrill}] \rightarrow V[h=\text{thrill}] NP[h=\text{Otto}]$

Equivalently: keep the template but make prob depend on $\alpha, \beta$.

- Why use heads?

- Subcategorization (i.e., transitive vs. intransitive)
- When is VP $\rightarrow$ V NP ok?
  $VP[h=\alpha] \rightarrow V[h=\alpha] NP$
  restrict to $\alpha \in$ TRANSITIVE VERBS
- When is N $\rightarrow$ N VP ok?
  $N[h=\alpha] \rightarrow N[h=\alpha] VP$
  restrict to $\alpha \in \{\text{plan, plot, hope,\ldots}\}$

- Selectional restrictions
- I.e., $VP[h=\alpha] \rightarrow V[h=\alpha] NP[h=\beta]$
- Don't fill template in all ways:
  $VP[h=\text{thrill}] \rightarrow V[h=\text{thrill}] NP[h=\text{Otto}]$
  * $VP[h=\text{thrill}] \rightarrow V[h=\text{thrill}] NP[h=\text{plan}]$

Equivalently: keep the template but make prob depend on $\alpha, \beta$. 
- 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

- Feature path: path through structures to value (e.g.
  Agr $\rightarrow$ Num $\rightarrow$ SG

---

**Features and grammars**

- **category**: N
- **agreement**:
  - **person**: third
  - **number**: singular

---

6•863J/9•611J SP04 Lecture 13
Feature checking by unification

John sleep

*John sleep

Our feature structures

- Maria NAME[agr [person 3, plural -]]

Kimmo entry for Verb (e.g., 'coge' after analysis):
- +e Suffix "[fin +, agr [tense pres, mode ind, person 3, plural -]]"
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 DAG \(_2\) – if all match, success, else fail
- WE WILL USE MUCH SIMPLER kind of feature structure

Features and Earley 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, & lexical entries
  - Add field to states containing set representing feature structure corresponding to state of parse, e.g.

\[
S \rightarrow • NP\ VP, [0,0], [], \text{Set}=[\text{Agr} \{\text{plural -}\}]
\]
• 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 feature structures may differ, so check must be more complex
    • Suppose feature structure is more specific than existing one tied to this state? Do we add it?

Evidence that you don’t need this much power

• Linguistic evidence: looks like you just check whether features are nondistinct, rather than equal or not – variable matching, not variable substitution
• Full unification lets you generate unnatural languages: \( a^i \), s.t. \( i \) a power of 2 – e.g., \( a, aa, aaaa, aaaaaaaa, \ldots \)
  why is this ‘unnatural’ – another (seeming) property of natural languages:
  Natural languages seem to obey a constant growth property
Parsing with features – hook from kimmo to earley

- Features written in this form (in Kimmo)

  - As Suffix "[fin +, agr [tense pres, mode ind, person 2, plural -]]"

  - In general: 
    [feature value, feature [feature val, ..., feature val]]

Where wolf

```
Start
  \[\text{S\text{\text{\text{-}}}f\text{\text{-}}}n \text{\text{-}}}\]
\[\text{N\text{\text{-}}}P\text{\text{-}}} \text{\text{-}}}\]
\[\text{\text{-}}}N\text{\text{-}}}\]
\[\text{\text{-}}}A\text{\text{-}}}U\text{\text{-}}}X\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{\text{-}}}\text{
<table>
<thead>
<tr>
<th>Maria</th>
<th>'cAge'</th>
<th>'to'</th>
<th>'wh'</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Sfb^f All</td>
<td>*Vf</td>
<td>*VP COM</td>
<td>*FB BAR</td>
</tr>
<tr>
<td>*Grdf VP</td>
<td>*Vf</td>
<td>*NP</td>
<td>*FB BAR</td>
</tr>
<tr>
<td>Sf^f TV</td>
<td>Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>Sf^f TVA</td>
<td>Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>Sf^f TVA</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maria</th>
<th>'cAge'</th>
<th>'to'</th>
<th>'wh'</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Sfb^f All</td>
<td>*Vf</td>
<td>*VP COM</td>
<td>*FB BAR</td>
</tr>
<tr>
<td>*Grdf VP</td>
<td>*Vf</td>
<td>*NP</td>
<td>*FB BAR</td>
</tr>
<tr>
<td>Sf^f TV</td>
<td>Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>Sf^f TVA</td>
<td>Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>Sf^f TVA</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
<tr>
<td>*Vf NP</td>
<td>*Vf NP</td>
<td>*NP BAR</td>
<td>*FACT SBAR</td>
</tr>
</tbody>
</table>
Constant growth property

Claim: \exists \text{ Bound } k \text{ on the `distance gap' between any two consecutive sentences in this list, which can be specified in advance (fixed)}

- ‘Intervals’ between valid sentences cannot get too big – cannot grow w/o bounds
- We can do this a bit more formally
Constant growth

- **Dfn.** A language \( L \) is *semilinear* if the number of occurrences of each symbol in any string of \( L \) is a linear combination of the occurrences of these symbols in some fixed, finite set of strings of \( L \).

- **Dfn.** A language \( L \) is *constant growth* if there is a constant \( c_0 \) and a finite set of constants \( C \) s.t. for all \( w \in L \), where \(|w| > c_0 \), there exists \( w' \in L \) s.t. \(|w| = |w'| + c\), some \( c \in C \).

- **Fact.** (Parikh, 1971). Context-free languages are semilinear, and constant-growth.

- **Fact.** (Berwick, 1983). The power of 2 language is non constant-growth.

General feature grammars – how violate these properties

- Take example from so-called “lexical-functional grammar” but this applies as well to any general unification grammar.

- **Lexical functional grammar (LFG):** add checking rules to CF rules (also variant HPSG).
Example LFG

- Basic CF rule:
  $S \rightarrow NP \ VP$
- Add corresponding ‘feature checking’
  $S \rightarrow NP \ VP$
  $(↑ \ subj \ num) = ↓ \ ↑ = ↓$
- What is the interpretation of this?

Applying feature checking in LFG

[subj [num singular]]

Copy up above

$S \rightarrow NP \ VP$

$(↑ \ subj \ num) = ↓ \ ↑ = ↓$

N
  [num plural]
  $↑$ guys

V
  [num singular]
  $↑ = ↓$
  sleeps

Whatever features from below
Alas, this allows non-constant growth, unnatural languages

- Can use LFG to generate power of 2 language
- Very simple to do
- \[ A \rightarrow A \]
  \[ (\uparrow f) = \downarrow \ (\uparrow f) = \downarrow \]
- \[ A \rightarrow a \]
  \[ (\uparrow f) = 1 \]

Let us `count' the number of embeddings on the right & the left – make sure a power of 2

Example

\[ f = 1 \]
\[ f[f = 1] \]
\[ f[f[f = 1]] \]
\[ f[f[f[f = 1]]] \]
\[ f[f[f[f[f = 1]]]] \]
\[ f[f[f[f[f[f = 1]]]]] \]
\[ f[f[f[f[f[f[f = 1]]]]]] \]
\[ f[f[f[f[f[f[f[f = 1]]]]]]] \]
\[ f[f[f[f[f[f[f[f[f = 1]]]]]]]] \]
\[ f[f[f[f[f[f[f[f[f[f = 1]]]]]]]]] \]
\[ f[f[f[f[f[f[f[f[f[f[f = 1]]]]]]]]]] \]

Checks ok
If mismatch anywhere, get a feature clash...

Conclusion then

- If we use too powerful a formalism, it lets us write ‘unnatural’ grammars
- This puts burden on the person writing the grammar – which may be ok.
- However, child doesn’t presumably do this (they don’t get ‘late days’)
- We want to strive for automatic programming – ambitious goal