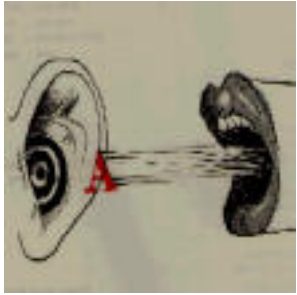


Lecture 6: Going nonlinear



Professor Robert C. Berwick
berwick@csail.mit.edu

The Menu Bar

- Administrivia
 1. Finish up word parsing
 2. Lab 3, out today
- Solution to Martha Stewart's *Revenge or The Recipe Problem*: a dynamic programming algorithm
- The most important Principle of ALL: AWP
Or: The IBM Power Lunch
- Three Loops to Rule them All, Three Loops to Bind them
- Removing redundancy redundancy

OK?

Now, on to an assessment of this
approach...

So, are we done????

- What can 2-level system do?
- What can't it do?
- What is its computational complexity?
- Is it too strong or too weak or both?

Two areas:

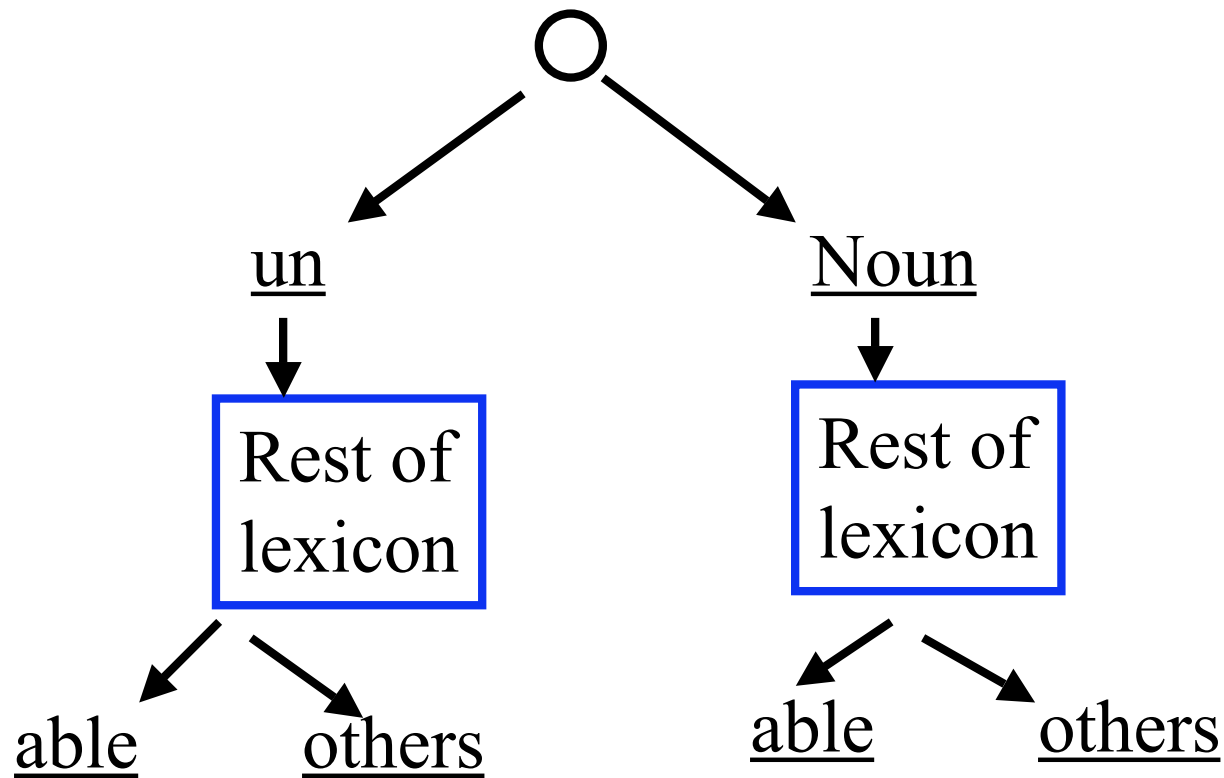
Is the two-level system sufficient?

Is the two-level system efficient?

Is 2-level morphology sufficient?

- So, this lets us think what the system *might not* be good for... let's look at English first....
- Is morphology really purely linear???
- There seem to be some kinds of 'long distance' constraints...
- Prefix/suffix links: only some prefixes tied to some suffixes
 - Un_____able
 - Undoable, uncanny, ?uncannyable, unthinkable, *unthink, thinkable, readable, unreadable, unkind, *unkindable
- So, we have to 'keep track' that the *un* is first or *not* – what does lexicon look like?

Lexicon must be (grotesquely) duplicated



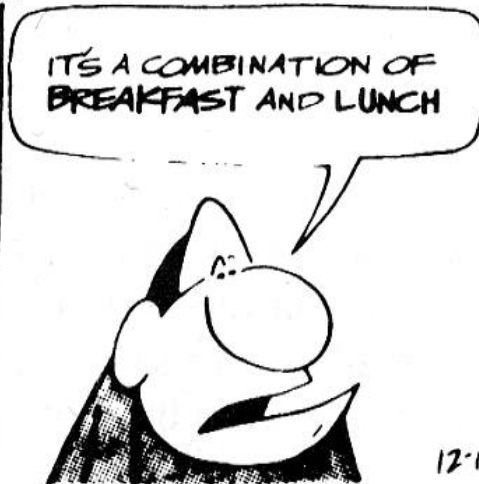
This kind of duplication is a litmus test of something wrong

- Duplication: no relation between the two lexicons, but we know they're identical
- Principle *AWP*
- We will see this again and again
- Usually means we haven't carved (factored) the knowledge at the right 'joints'
- Solution? Usually more powerful machinery 'overlay' representations

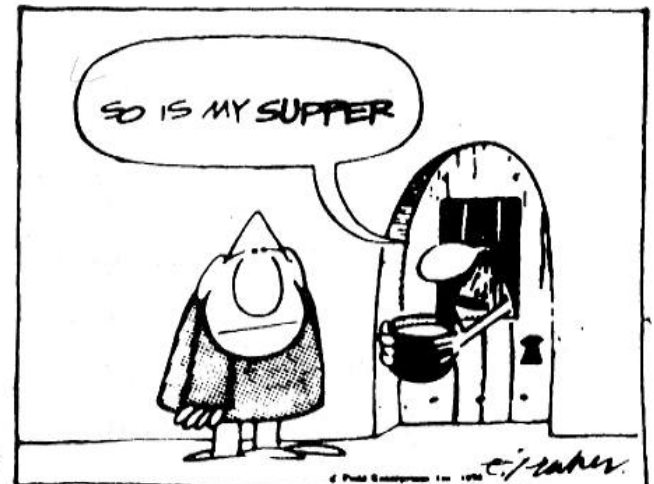
And finally...

- Is morphology really linear? (strictly associative) or does the order of composition matter, as in ‘dark blue sky’?
- Example: decentralization is really found this way:
([de ([center-al] ize)] ation)

Somehow, we haven’t captured possibly hierarchical structure – instead, shoehorned in



12-14



Similar example of ‘long distance’ constraint

- French elision: le, la: l’arbe; l’homme
- Always put in front, elided if noun/adj begins w/ a vowel
 - However, blocked if noun is plural: *l’arbés,
les arbés

Not *all* long distance effects are a barrier...

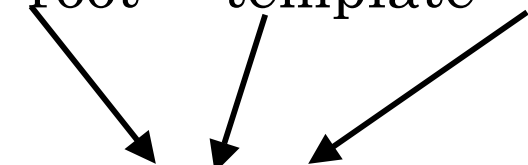
- Phenomena: *Vowel harmony*
 - *yourgun + sInIz → yorgunsunuz*
 - Round vowels assimilate to round vowels; back vowels to back, etc. - all the way from left to right
- Can a 2-level system do it? What would be your model? Suppose harmony is right to left?

Interdigitation in Arabic

Concatenative:

kuutib + a
“stem” “suffix”

Non-concatenative stem:

ktb **CVVCVC** **ui**
“root” “template” “vocalization”

kuutib

•

Informally speaking, the root, template and vocalization morphemes “interdigitate” into a stem.

Another example: Tagalog

- | Root | CV+root | Gloss |
|------|---------|----------|
| pili | pipili | ‘choose’ |
| tahi | tatahi | ‘sew’ |
| kuha | kukuha | ‘take’ |

What’s going on? How to do in 2-level system?

What would you propose?

Need extensions

- Add multiple intersections to interdigitate:
 $CCC^{\wedge}VV \rightarrow CVCVC$ then go on from there...
- In general – more powerful machine
- Not yet completely explored

Sufficiency: The coup de...Bambara

wulu o wulu = ‘whichever dog’ Noun o Noun

wulu+nyini+la = ‘dog searcher’

wulunyinina+nyini+la = ‘one who searches for dog searchers’

wulunyininanyinila+nyini+la = ‘one who searches for one who searches for dog searchers’

Combine w/ Noun-o-Noun pattern:

wulunyininanyinila+o+wulunyininanyinila= ‘which one who searches for dog searchers’

wulunyininanyinilanyinila o wulunyininanyinilanyinila

Is the two-level system efficient?

- To decide whether two level automata are appropriate models for human phonology or morphology, we can consider:
 - Do these models appropriately capture the properties of (generalizations about) human phonology and morphology?
- Do these models appropriately constrain the space of possibilities, allowing the possibility of explaining why many non-human systems never occur?

Is the power of 2-level systems necessary?

- Does it explain why many non-human systems never occur (ruling them out)
- Or does it overshoot?
- Is the backtracking we see inherent in the machinery or eliminable?

- Ans: it seems to overshoot, in at least 2 ways
- Overshoots detected by computational analysis
- The backtracking cannot be eliminated (in the worst case)

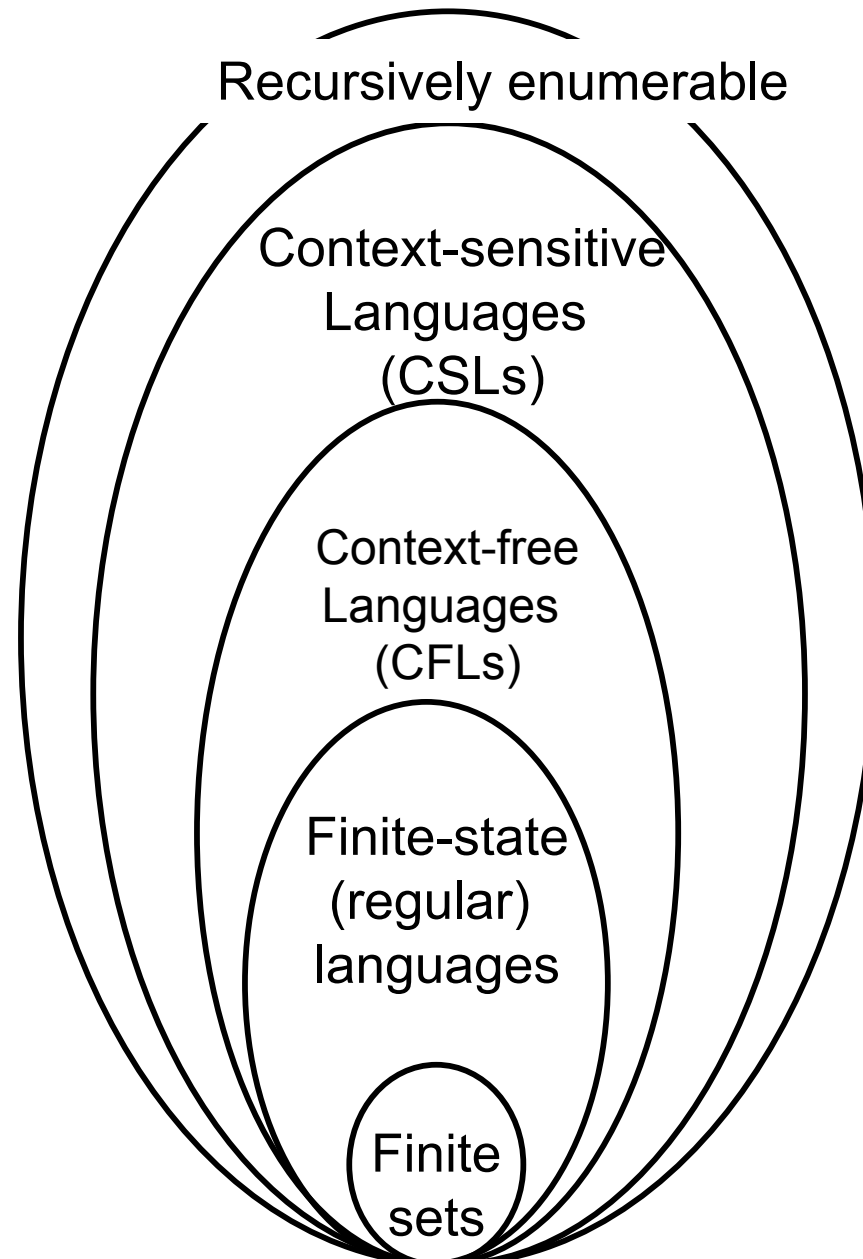
Oh, it's always easy to parse words...

buffalo buffalo buffalo

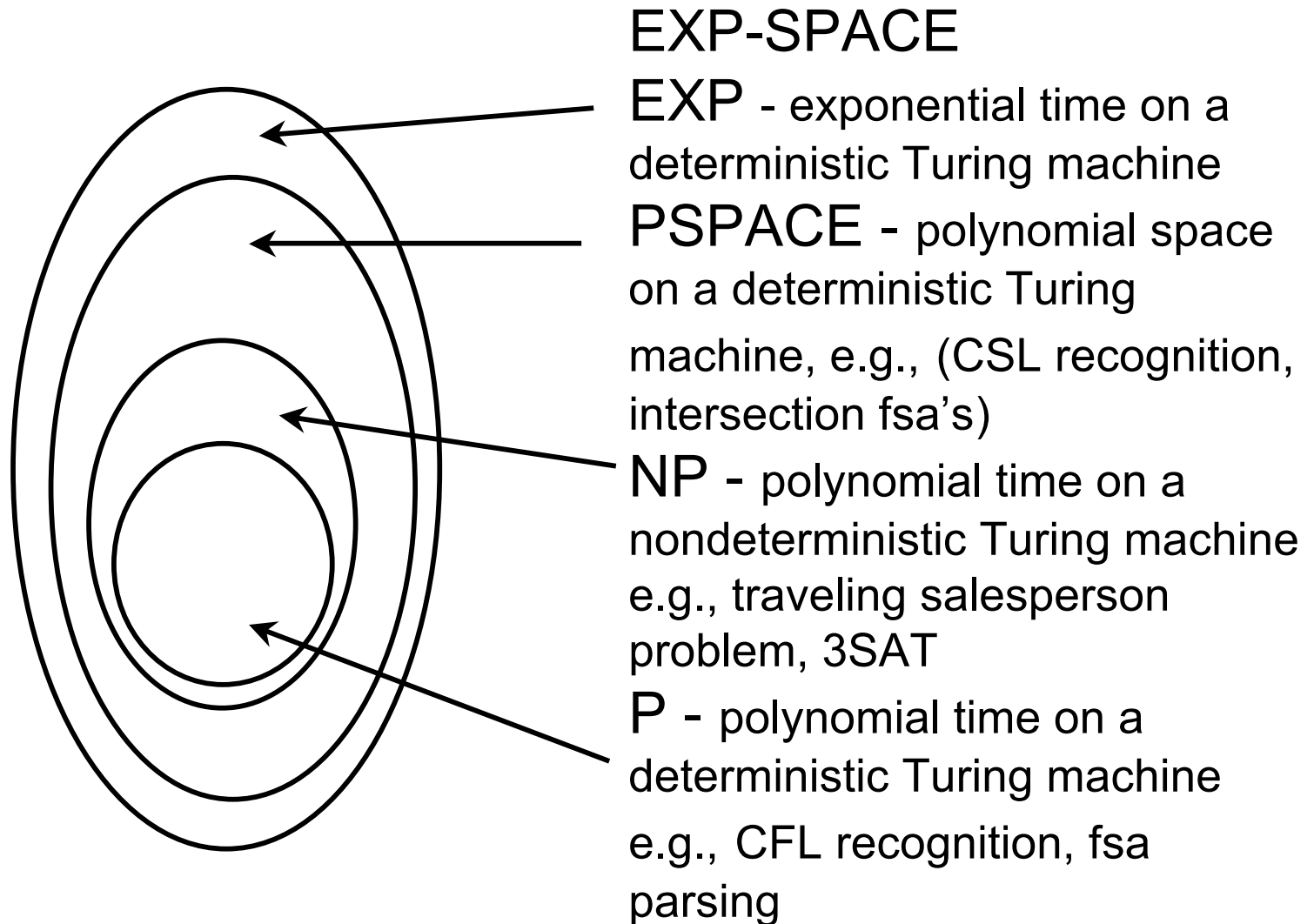
buffalo buffalo buffalo buffalo buffalo

buffalo buffalo buffalo buffalo buffalo buffalo buffalo

Remembrance of containments past...



A resource complexity hierarchy (not known whether inclusions are proper)



Two-level automata recognition is NP-hard!

- As hard as the hardest problems in NP
- No known polynomial time algorithm
- The argument goes like this:
 - the problem of deciding whether a 3-CNF formula is satisfiable is NP-complete
 - this 3-SAT problem can be represented as a recognition problem in a two level automaton; therefore,
 - the recognition problem for two level automata can be at least as hard as 3-SAT.

Reduction

Any 3-Sat problem

← Efficient (polynomial
time) transformation

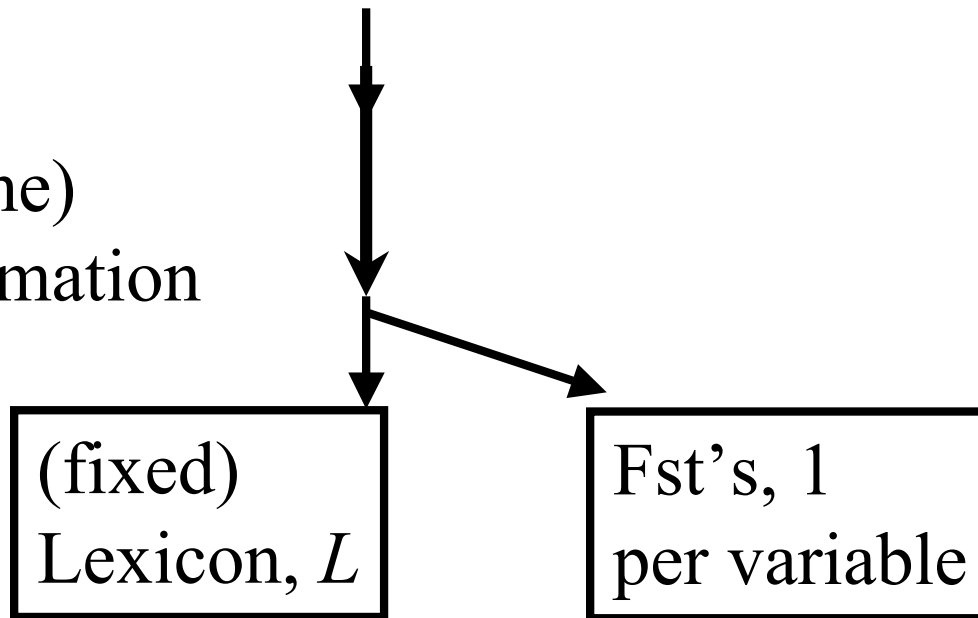
Equivalent
Two-level recognition problem

Answer to original SAT
problem

The reduction:

Given: arbitrary 3-SAT problem instance

Fast
(polytime)
transformation



$\text{word} \in L$ if Sat instance satisfiable

If we could solve 2-level recognition easily,
Then we could solve 3-Sat easily

Two components to 3-Sat

- The fact that an x that has a truth assignment in one place, must have the same truth assignment everywhere - what morphological process is that like?
- The fact that every triple must have at least 1 ‘T’ (‘true’) underlyingly (so that the triple is true) - what morphological process is that like?

How the reduction works

Given arbitrary 3-sat formula Φ , e.g.,

$$(x \vee \neg y \vee z) (\neg x \vee \neg z \vee w) (x \vee y \vee w)$$

Represent in the form, a long ‘word’ with morpheme triples:

$$x\bar{y}z, \bar{x}\bar{z}w, xyw, .$$

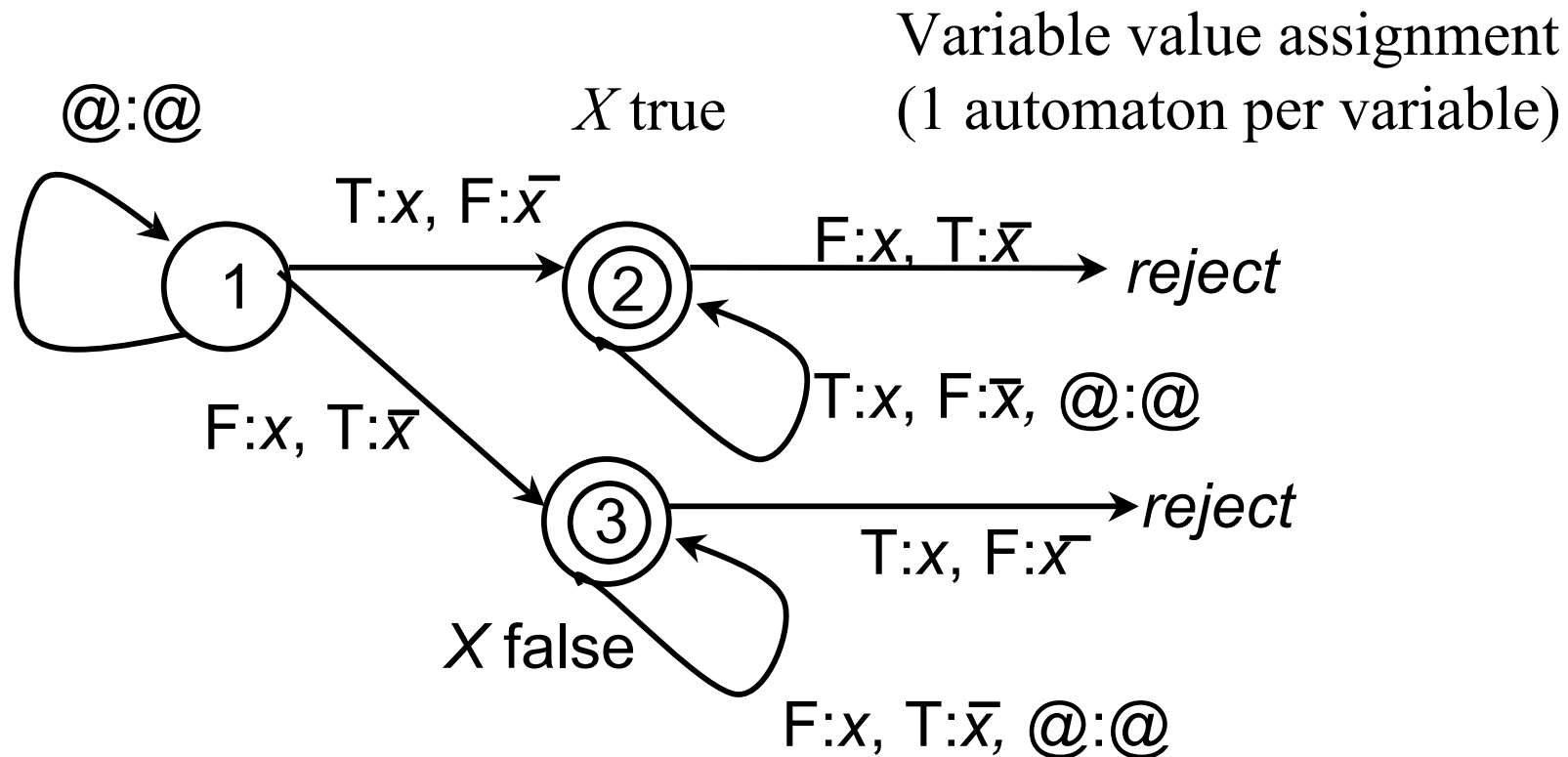
For each variable x , we have an ‘assignment machine’ that ensures that x is mapped to T or F throughout the whole formula

We have one machine (and a fixed dictionary) to checks each disjunction to make sure that at least one disjunct is true in every conjunct

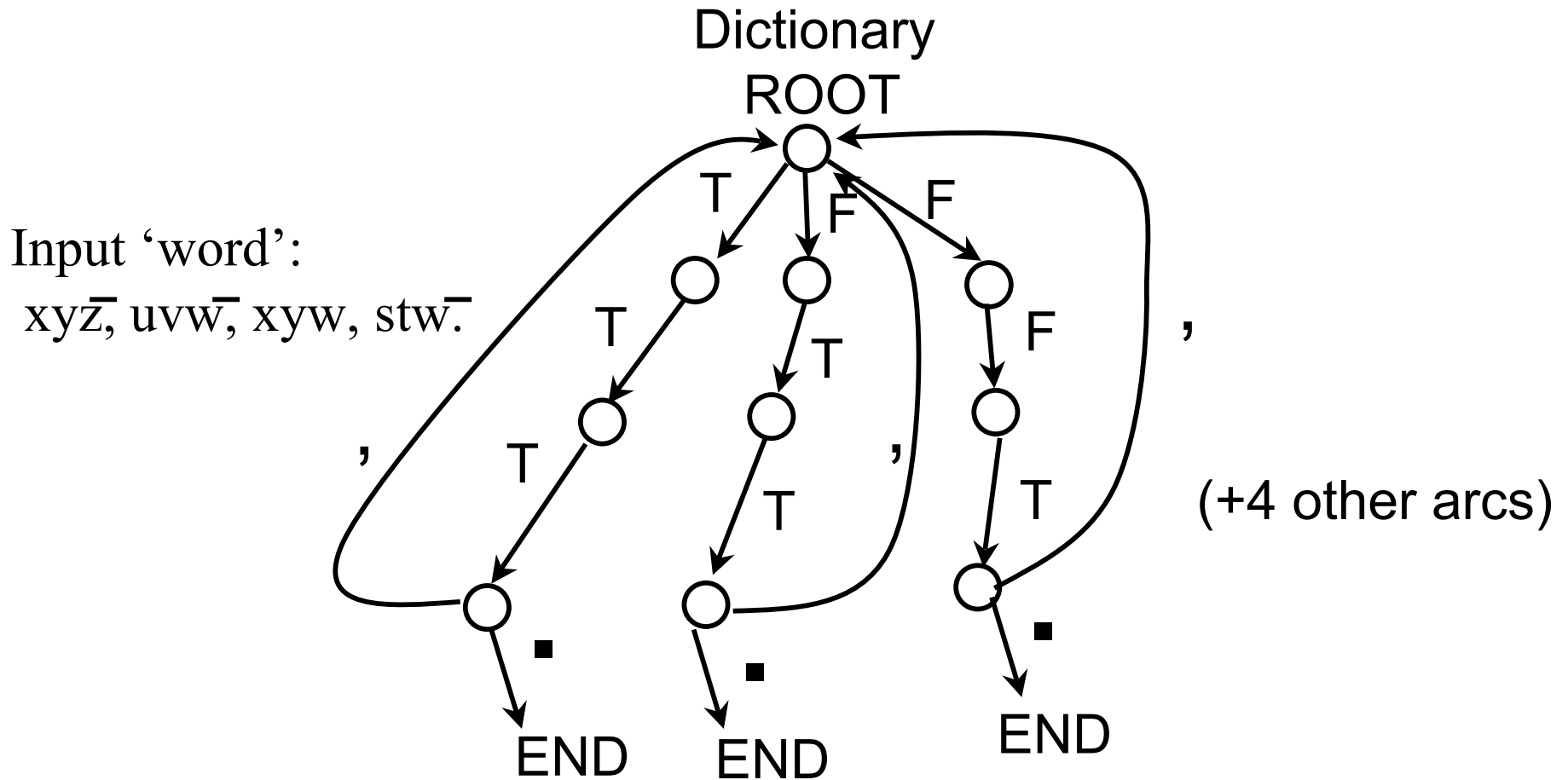
Two components

- Agreement: vowel harmony (if round at some point, round everywhere)
- Ambiguity: we can't tell what the underlying value of x is from the surface, but if there's at least one "t" per 'part of word', then we can spell out this constraint in dictionary
- Note that words (like Sat formulas) must be arbitrarily long... (pas de probleme)
- Dictionary is fixed...
- # of Vowel harmony processes corresponds to # of distinct literals (variables or their negations)

Reduce until done: variable assignment consistency



Reduce until done – 3-SAT formula must eval
to *true*, so at least 1 literal in every triple
must be assigned the value true

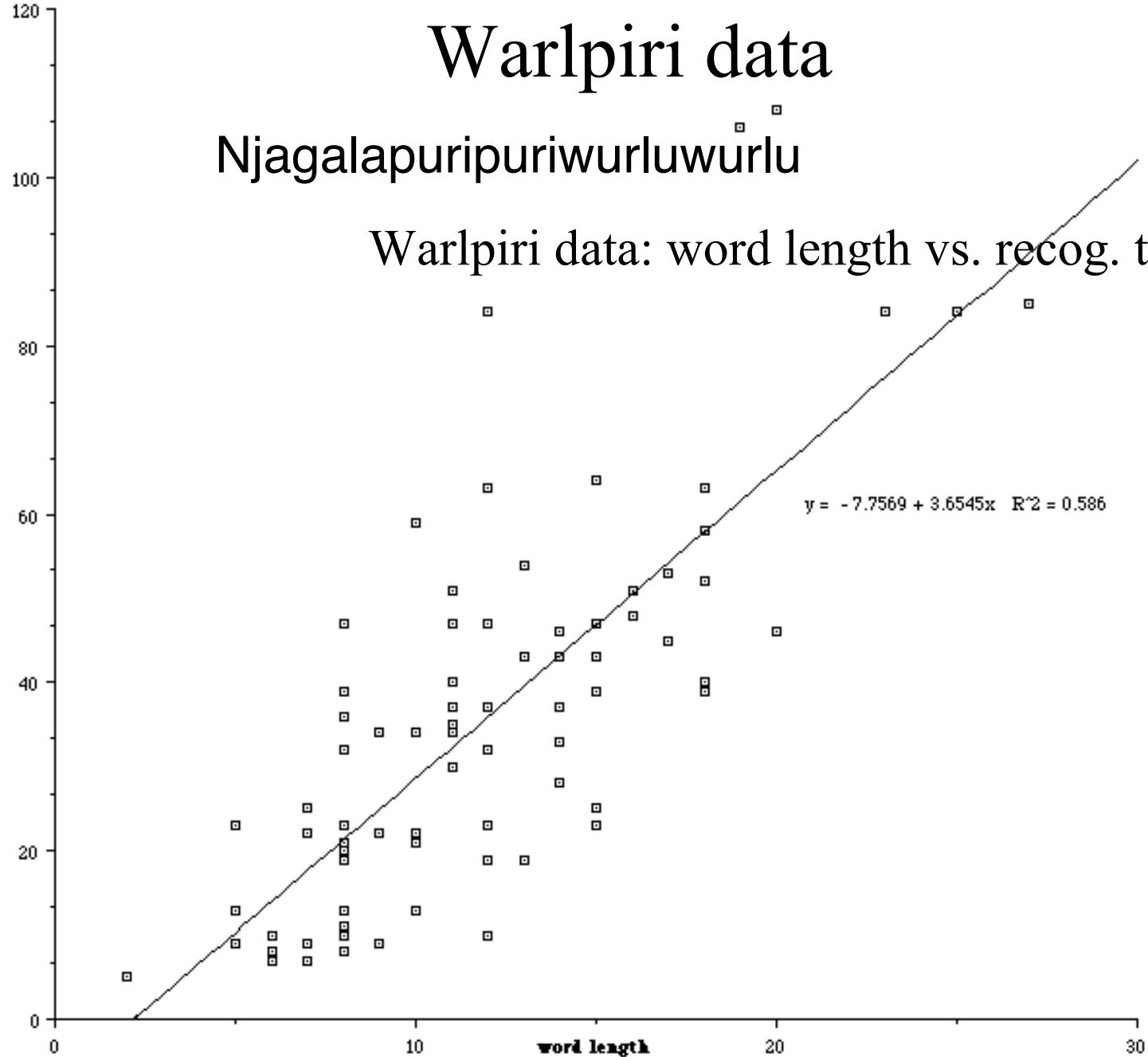


Does this hurt in 'real life'?

Warlpiri data

Njagalapuripuriwurluwurlu

Warlpiri data: word length vs. recog. time



Relevance

- This raises the question: do we need a recognition system that is powerful enough to represent intractable problems?
- While reduplication phenomena need to be accounted for, there is no evidence that we need mechanisms that can make 100 copies, or enforce 100 counting dependencies
- This could be due just to ‘performance’ restrictions, but we should consider whether there are weaker systems that can do the job needed without being able to do so many other things too

From word parsing to phrase parsing: some motivation

- Motivation: just as words are broken into parts, so are sentences
- We need to know these to figure out meaning since, e.g., meaning of ‘brown cow’ is composed from its parts
- Word substitution classes → finite automata states
- What about word sequences? Like ‘the dog’, ‘the cat’?
- They lead to phrase classes, or subroutine calls, and make grammars smaller by extracting regularities
- Parsing is inextricably bound to semantic interpretation— we’ll give a simple example, with sentences, noun phrases, verb phrase.

From finite-state transition networks
(fsa's) to recursive transition networks,
and so context-free grammars

- *The guy saw the dog*
- Write this out as a finite-transition network
- *The dog saw the person I met last Wednesday*
- Write this out as a finite-transition network
- What 'compaction' are we missing?
- Write this as a recursive transition network, or a push-down stack automaton (pda)

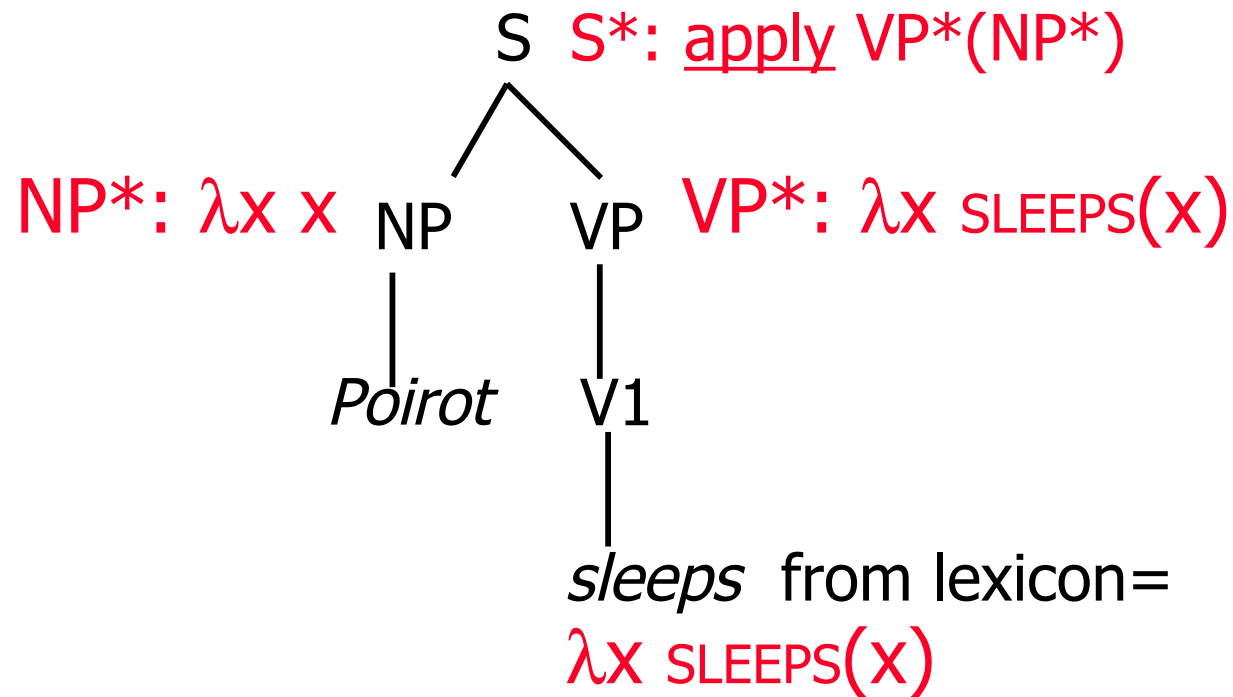


Hierarchical Parsing: *Why* are we doing this at all? Context-free semantics: associate interpretation rule with each context-free rule

<u>Item or rule</u>	<u>Semantic translation</u>
$S \rightarrow NP VP$	$S^*: \text{apply } VP^*(NP^*)$
$VP \rightarrow \textit{sleeps}$	$VP^*: \lambda x \text{ SLEEPS}(x)$
$NP \rightarrow \text{Name}$	$NP^*: \lambda x x$
$\text{Name} \rightarrow \textit{Poirot}$	$\text{Name}^*: \text{'Poirot' (ie, a constant)}$

- What do we have to know to recover information to assign thematic roles (argument structure)

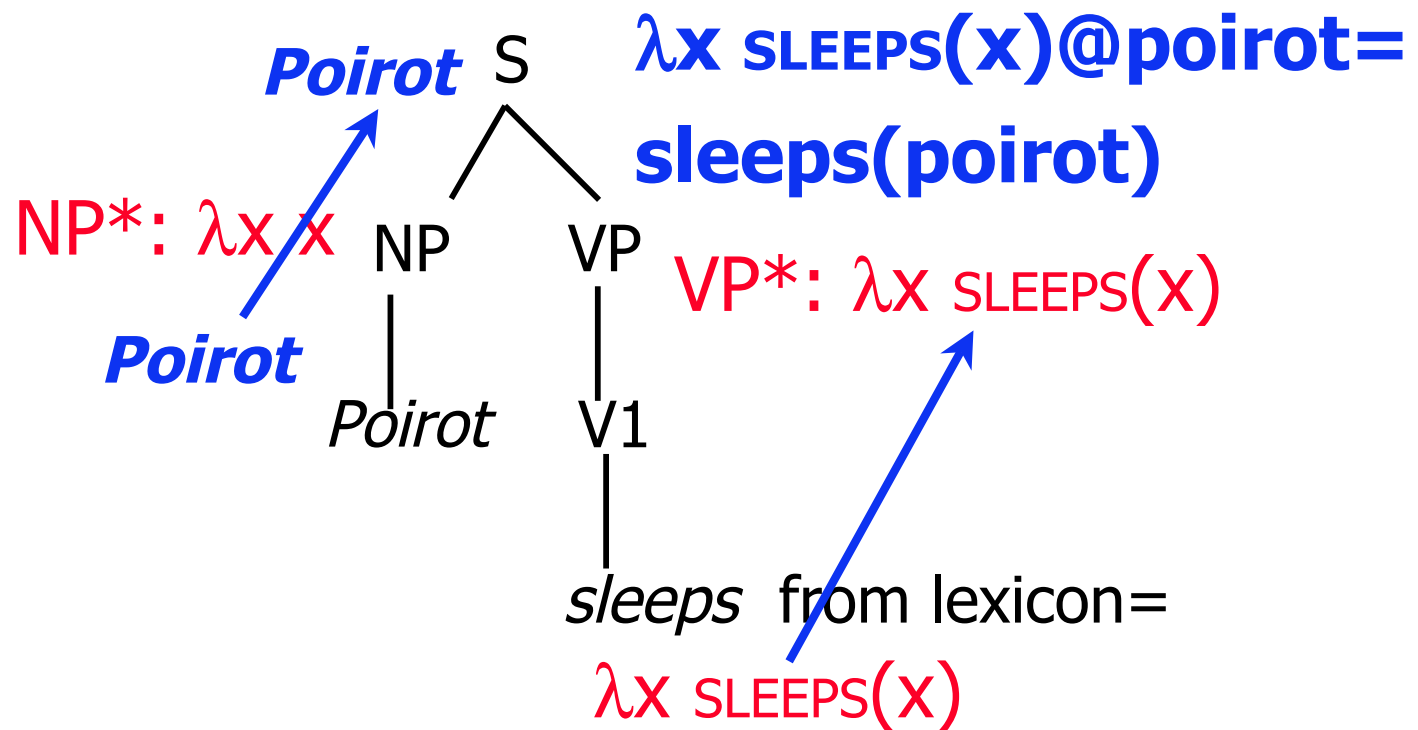
Rule-to-rule principle: decorate syntax tree with lambda calculus



Now it's all just eval-apply! (“beta reduction”)

Doing the beta reductions...

S^* : apply $VP^*(NP^*) \rightarrow$



Review: Context-free grammar (CFG) More precisely

A *context-free grammar* (CFG) is a 4-tuple (N, Σ, P, S) where:

N is a finite set of nonterminal symbols (phrase names, categories);

Σ is a finite set of terminal symbols ('lexical items', 'parts of speech', eg, noun, verb, auxiliary verb, ...);

P is a set of production rules $\langle A \in N, \alpha \rangle$, where α is a sequence of terminal or nonterminals; sometimes also written in the form

$X \rightarrow Y_1 Y_2 \dots Y_n$ for $n \geq 0$, $X \in N$, $Y_i \in (N \cup \Sigma)$

$S \in N$ is a designated start symbol.

We write the productions as $A \rightarrow \alpha$ ('is-a')

Note: this is already off the mark from human language in at least two ways! (What ways?)

Languages, grammars, and all that

- A (terminal) string $s \in \Sigma^*$ is in the language generated by the context-free grammar G iff there is at least one derivation that yields s
- The set of strings derivable (in the language of) G is denoted $L(G)$, the grammar's *weak generative capacity*
- A string may have more than one derivation (as in the fsa case!) = *ambiguity*
- *Parsing* = given an input sentence w , CFG G , recover the (set of derivations) of w wrt G
- And that is what makes parsing hard! Why?

Canonical derivations in CFGs

- **Left-most derivation** in a CFG:

Sequence of derivation lines s_1, s_2, \dots, s_n

where $s_1 =$ Start symbol

$s_n \in \Sigma^*$, that is, all terminal symbols

For $i=2, \dots, n$, each s_i is derived from s_{i-1} by selecting the leftmost nonterminal X in s_{i-1} and replacing it by some β , the right-hand side of a rule $X \rightarrow \beta$ in P

(Can you tell me how to map derivations to ‘hierarchical structures’– no, no, do not use the ‘T’ word....)

CFGs, languages, and derivations

- The derives relation \Rightarrow
- Define wrt grammar $G = (N, \Sigma, P, S)$ as follows
 $\alpha \Rightarrow \beta$ iff $\exists \alpha_1, \alpha_2$ s.t. $\alpha = \alpha_1 A \alpha_2$; $\beta = \alpha_1 \gamma \alpha_2$; and
 $A \rightarrow \gamma \in P$. (Some rule rewrites α as β)
- Reflexive, transitive closure of \Rightarrow is \Rightarrow^*

If α, β is in \Rightarrow^* then we say that α derives β (by 0 or more steps)

The *language generated* by a cfg G is the set of terminal strings $w \in \Sigma^*$ s.t. $S \Rightarrow^* w$ (also called the yield of S)

Derives relation & added info in context-free grammar

- Relates all elts by *either* dominance or precedence
- Induces a (derivation) tree (Q: do we lose any information in this tree?)
- FSA represents pure *linear* relation: what can *precede* or (*follow*) what
- CFG adds a single new predicate: *dominate*
- Claim: The dominance and precedence relations amongst the words exhaustively describe its *syntactic* structure
- When we parse, we are recovering these predicates

Derivation induces 2 relations, dominance & precedence

- Binary Relation D , dominance:

$A D v$ iff $\exists \alpha_1, \alpha_2$ s.t. $\alpha \Rightarrow \beta$ via $A \rightarrow \alpha_1 v \alpha_2$

- Binary relation $<$ precedence:

$v < w$ iff $\exists \alpha_1, \alpha_2$ s.t. $\alpha = \alpha_1 v w \alpha_2$ or $\beta = \alpha_1 v w \alpha_2$ & $\alpha \Rightarrow \beta$

Confirm that our derivation steps previously induce such relations... note that all elts are related by $<$ or D .
(Suppose not...?)

A context-free grammar

$S \rightarrow NP V$

$NP \rightarrow Det N$

$NP \rightarrow NP PP$

$VP \rightarrow VP PP$

$VP \rightarrow V NP$

$PP \rightarrow P NP$

$Det \rightarrow a$

$S \rightarrow NP VP$

$N \rightarrow caviar$

$N \rightarrow spoon$

$V \rightarrow spoon$

$V \rightarrow ate$

$P \rightarrow with$

$Det \rightarrow the \mid a$

$N \rightarrow guy$

A derivation of:
Madoff ate caviar with a spoon

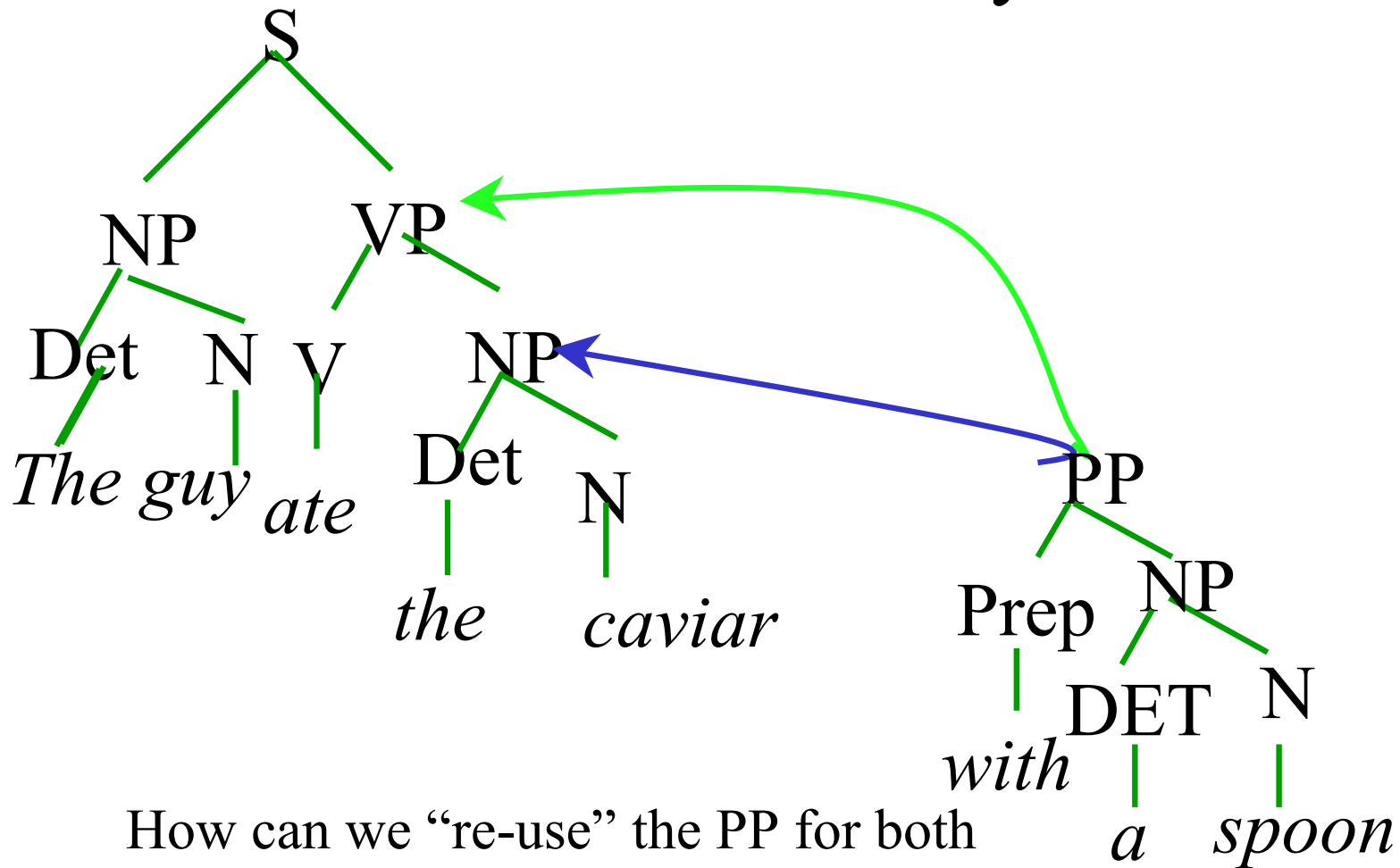
1. Root
2. S
3. NP VP
4. NP V NP
5. NP V NP PP
6. NP V NP P NP
7. NP V NP P Det N
8. NP V NP P Det *with*
9. NP V NP P *a spoon*
10. NP V NP *with a spoon*
11. NP V Det N *with a spoon*
12. NP V Det *caviar with a spoon*
13. NP V *the caviar with a spoon*
14. NP *ate the caviar with a spoon*
15. Det N *ate the caviar with a spoon*
16. Det *guy ate the caviar with a spoon*
17. The guy ate the caviar with a spoon

Now parse this...

Some simple methods

- Shift-reduce parsing: stack + two operations....*shift* item onto stack, or *reduce* (via grammar rule); bottom up
- What's the problem with this?
- Recursive descent parsing: top-down

Marxist analysis



How can we “re-use” the PP for both analyses? Otherwise, we will do exponential work!

Local & Global ambiguity makes parsing hard

- Local ambiguity

- Plural/singular: *the sheep are/is...*
- *The chair is too wobbly for the woman to sit on/on it*

- Global ambiguity

NP → NP PP | PP → P NP, *the guy on the hill with the telescope*

<i>n</i>	1	2	3	4	5	6	7	8
#	2	5	14	132	469	1430	4862	16796

Noun compounds: NP → NN NN, *water meter cover screw*

Conjunctions: NP → NP *and* NP

NP → NP *and* NP *and* NP

<i>n</i>	1	2	3	4	5	6	7	8	9
#	1	1	3	11	45	197	903	4279	20793

Why is parsing hard?

Martha Stewart's revenge

- If you write cookbooks...this from an actual example in a 30M word corpus...

Combine grapefruit with bananas, strawberries and bananas, bananas and melon balls, raspberries or strawberries and melon balls, seedless white grapes and melon balls, or pineapple cubes with orange slices...

of parses with 10 conjuncts is 103,049
(grows approx as $6^{\#}$ conjuncts)