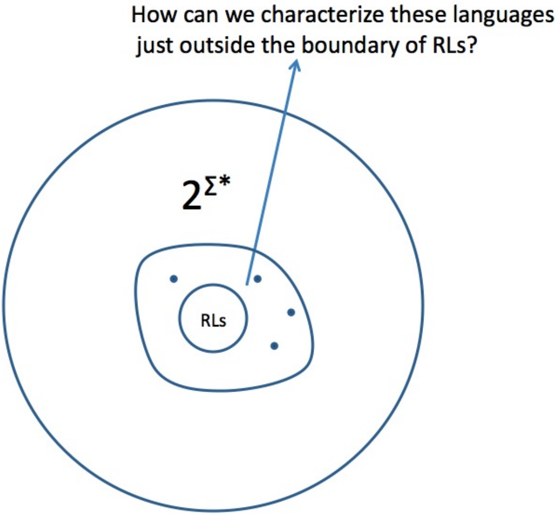
FORMAL LANGUAGES, AUTOMATA AND COMPUTATION

CONTEXT FREE LANGUAGES

Carnegie Mellon University in Qatar

Where are we?


How can we characterize these languages just outside the boundary of RLs?
2^{Σ^*}
RLs

A Nonregular Language

- We showed that $L = \{a^n b^n \mid n \geq 0\}$ was not regular.
  - No DFA
  - No Regular Expression
- How can we describe such languages?
- Remember: the description has to be finite!

A Nonregular Language

- Consider $L = \{a^n b^n \mid n \geq 0\}$ again.
- How can we generate such strings?
  - Remember DFAs did recognition, not generation.
- Consider the following inductive way to generate elements of L
  - Basis: ϵ is in the language
  - Recursion: If the string w is in the language, then so is the string awb.
- $\epsilon \rightarrow ab \rightarrow aabb \cdots \rightarrow a^{55}b^{55} \cdots$
- Looks like we have simple and finite length process to generate all the strings in L
- How can we generalize this kind of description?

Another Nonregular Language

- Consider $L = \{w \mid n_a(w) = n_b(w)\}$.
- Now consider the following inductive way to generate elements of L
  - Basis: ϵ is in the language
  - Recursion 1: If the string w is in the language, then so are awb and bwa
  - Recursion 2: If the strings w and v are in the language, so is wv.
- The first recursion rules makes sure that the a's and b's are generated in the same number (regardless of order)
- The second recursion takes any two strings each with equal number of a's and b's and generates a new such string by concatenating them.

GRAMMARS

- Grammars provide the generative mechanism to generate all strings in a language.
- A grammar is essentially a collection of substitution rules, called productions
- Each production rule has a left-hand-side and a right-hand-side.

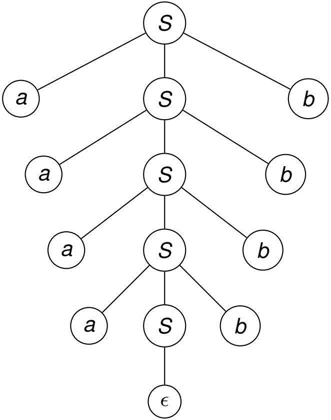
Grammars - An Example

- Consider once again $L = \{a^n b^n \mid n \geq 0\}$
- Basis: ϵ is in the language
  - Production: $S \rightarrow \epsilon$
- Recursion: If w is in the language, then so is the string awb.
  - Production: $S \rightarrow aSb$
- S is called a variable or a nonterminal symbol
- a, b etc., are called terminal symbols
- One variable is designated as the start variable or start symbol.

How does a grammar work?

- Consider the set of rules $R = \{S \rightarrow \epsilon, S \rightarrow aSb\}$
- Start with the start variable S
- Apply the following until all remaining symbols are terminal.
  - Choose a production in R whose left-hand sides matches one of the variables.
  - Replace the variable with the rule's right hand side.
- $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$
- The string $aaaabbbb$ is in the language L
- The sequence of rule applications above is called a derivation.

Parse Trees



The terminals concatenated from left to right give us the string.

- Derivations can also be represented with a parse tree.
- The leaves constitute the yield of the tree.
- Terminal symbols can occur only at the leaves.
- Variables can occur only at the internal nodes.

Language of a Grammar

- All strings generated this way starting with the start variable constitute the language of the grammar.
- We write $L(G)$ for the language of the grammar G.

A GRAMMAR FOR A FRAGMENT OF ENGLISH

$$
\begin{array}{rcl}
S & \rightarrow & NP\ VP \\
NP & \rightarrow & CN \mid CN\ PP \\
VP & \rightarrow & CV \mid CV\ PP \\
PP & \rightarrow & P\ NP \\
CN & \rightarrow & DT\ N \\
CV & \rightarrow & V \mid V\ NP \\
DT & \rightarrow & \text{a} \mid \text{the} \\
N & \rightarrow & \text{boy} \mid \text{girl} \mid \text{flower} \mid \text{telescope} \\
V & \rightarrow & \text{touches} \mid \text{likes} \mid \text{sees} \mid \text{gives} \\
P & \rightarrow & \text{with} \mid \text{to}
\end{array}
$$

Nomenclature:

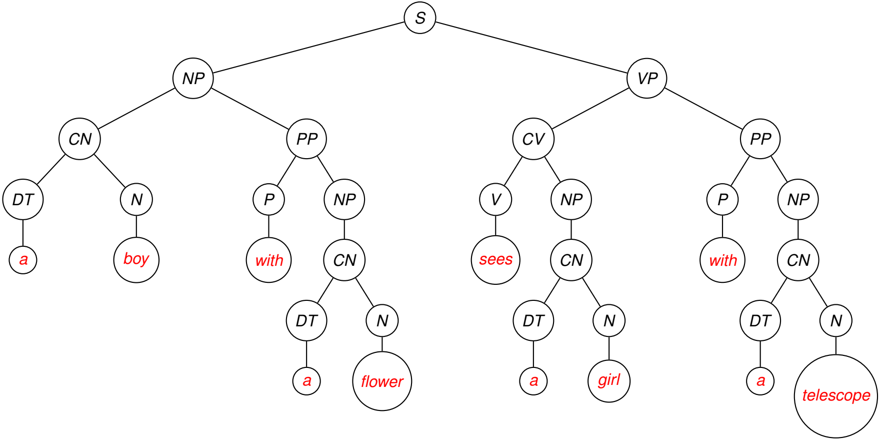
- *S*: Sentence
- *NP*: Noun Phrase
- *CN*: Complex Noun
- *PP*: Prepositional Phrase
- *VP*: Verb Phrase
- *CV*: Complex Verb
- *P*: Preposition
- *DT*: Determiner

A Grammar for a Fragment of English

$$
\begin{aligned}
S &\rightarrow NP\ VP \\
NP &\rightarrow CN \mid CN\ PP \\
VP &\rightarrow CV \mid CV\ PP \\
PP &\rightarrow P\ NP \\
CN &\rightarrow DT\ N \\
CV &\rightarrow V \mid V\ NP \\
DT &\rightarrow \text{a} \mid \text{the} \\
N &\rightarrow \text{boy} \mid \text{girl} \mid \text{flower} \mid \text{telescope} \\
V &\rightarrow \text{touches} \mid \text{likes} \mid \text{sees} \mid \text{gives} \\
P &\rightarrow \text{with} \mid \text{to}
\end{aligned}
$$

$$
\begin{aligned}
S &\Rightarrow NP\ VP \\
&\Rightarrow CN\ PP\ VP \\
&\Rightarrow DT\ N\ PP\ VP \\
&\Rightarrow \text{a}\ N\ PP\ VP \\
&\Rightarrow \ldots \\
&\Rightarrow \text{a boy with a flower}\ VP \\
&\Rightarrow \text{a boy with a flower}\ CV\ PP \\
&\Rightarrow \ldots \\
&\Rightarrow \text{a boy with a flower sees a girl with a telescope}
\end{aligned}
$$

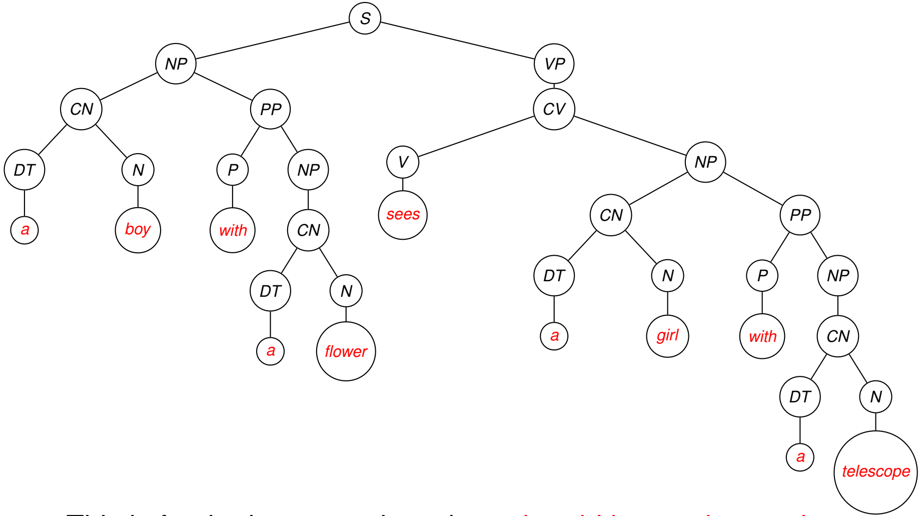
English Parse Tree



- This structure is for the interpretation where the boy is seeing with the telescope!

English Parse Tree

Alternate Structure



- This is for the interpretation where the girl is carrying a telescope.

STRUCTURAL AMBIGUITY

- A set of rules can assign multiple structures to the same string.
- Which rule one chooses determines the eventual structure.
  - $VP \rightarrow CV \mid CV\ PP$
  - $CV \rightarrow V \mid V\ NP$
  - $NP \rightarrow CN \mid CN\ PP$
  - $\cdots$ $[_{VP}$ $[_{CV}$ sees $[_{NP}$ a girl$]$ $[_{PP}$ with a telescope$]]$.
  - $\cdots$ $[_{VP}$ $[_{CV}$ sees$]$ $[_{NP}$ $[_{CN}$ a girl$]$ $[_{PP}$ with a telescope$]]$.
    - (Not all brackets are shown!)

Other Examples of Grammar Applications

- Programming Languages
  - Users need to how to "generate" correct programs.
  - Compilers need to know how to "check" and "translate" programs.
- XML Documents
  - Documents need to have a structure defined by a DTD grammar.
- Natural Language Processing, Machine Translation

Formal Definition of a Grammar

- A Grammar is a 4-tuple $G = (V, \Sigma, R, S)$ where
  - V is a finite set of variables
  - Σ is a finite set of terminals, disjoint from V.
  - R is a set of rules of the $X \to Y$
  - $S \in V$ is the start variable
- In general $X \in (V \cup \Sigma)^+$ and $Y \in (V \cup \Sigma)^*$
- A context-free grammar is a grammar where all rules have $X \in V$ (remember $V \subset (V \cup \Sigma)^+$)
  - The substitution is independent of the context V appears in.
- The right hand side of the rules can be any combination of variables and terminals, including ϵ (hence $Y \in (V \cup \Sigma)^*$).

FORMAL DEFINITION OF A GRAMMAR

- If u, v and w are strings of variables and terminals and $A \rightarrow w$ is a rule of the grammar, we say uAv yields uwv, notated as $uAv \Rightarrow uwv$
- We say u derives v, notated as, $u \stackrel{*}{\Rightarrow} v$, if either
  - $u = v$, or
  - a sequence $u_1, u_2, \ldots, u_k$, $k \geq 0$ exists such that $u \Rightarrow u_1 \Rightarrow u_2, \cdots, \Rightarrow u_k \Rightarrow v$.
  - We call u, v, and all u_i as sentential forms.
- The language of the grammar is $\{w \in \Sigma^* \mid S \stackrel{*}{\Rightarrow} w\}$

Designing Context Free Grammars

- Consider once again the language $L = \{w \mid n_a(w) = n_b(w)\}$.
- The grammar for this language is $G = (\{S\}, \{a, b\}, R, S)$ with R as follows:
    1. $S \rightarrow aSb$
    2. $S \rightarrow bSa$
    3. $S \rightarrow SS$
    4. $S \rightarrow \epsilon$
- From now we will only list the productions, the others will be implicit.
- We will also combine productions with the same left-hand side using $\mid$ symbol.
- $S \rightarrow aSb \mid bSa \mid SS \mid \epsilon$

Designing Context Free Grammars

- $L = \{w \mid n_a(w) = n_b(w)\}$.
- $S \rightarrow aSb \mid bSa \mid SS \mid \epsilon$

- Clearly the strings generated by G have equal number of a's and b's. (Obvious from the rules!)
- We also have to show that all strings in L can be generated with this grammar.

Designing Context Free Grammars

Assertion

Grammar G with $R = \{S \rightarrow aSb \mid bSa \mid SS \mid \epsilon\}$ generates $L = \{w \mid n_a(w) = n_b(w)\}$.

Proof (by Induction)

- The grammar generates the basis strings of ϵ, ab and ba.
- All other strings in L have even length and can be in one of the 4 possible forms:
  1. awb ($w \in \Sigma^*$)
  2. bwa
  3. awa
  4. bwb

Designing Context Free Grammars

Proof (continued)

- Assume that G generates all strings of equal number of a's and b's of (even) length n.
- Consider a string like awb of length $n+2$.
- awb will be generated from w by using the rule $S \rightarrow aSb$ provided $S \stackrel{*}{\Rightarrow} w$.
- But w is of length n, so $S \stackrel{*}{\Rightarrow} w$ by the induction hypothesis.
- There is a symmetric argument for strings like bwa.

Designing Context Free Grammars

Proof (continued)

- Consider a string like awa. Clearly $w \notin L$. Consider (symbols of) this string annotated as follows

$$_0a_1 \cdots _{-1}\, a_0$$

 where the subscripts after a prefix v of awa denotes $n_a(v) - n_b(v)$.

- Think of this as count starting as 0, each a adding one and each b subtracting 1. We should end with 0 at the end.
- Note that right after the first symbol we have 1 and right before the last a we must have -1.
- Somewhere along the string (in w) the counter crosses 0.

Designing Context Free Grammars

Proof (continued)

- Somewhere along the string (in w) the counter crosses 0.

$$_0\overbrace{a_1 \cdots x}^{u}{}_0\underbrace{y \cdots {}_{-1}\, a}_{v}{}_0 \quad x, y \in \{a, b\}$$

- So u and v have equal numbers of a's and b's and are shorter.
- u, $v \in L$ by the induction hypothesis and the rule $S \to SS$ generates $awa = uv$, given $S \stackrel{*}{\Rightarrow} u$ and $S \stackrel{*}{\Rightarrow} v$
- There is a symmetric argument for strings like bwb.