MA/CSSE 474
Theory of Computation

Context-Free Grammars

Context-Free Grammars

A **context-free grammar** (a.k.a. CFG) G is a quadruple, (V, Σ, R, S), where:

- V is the **rule alphabet** (**vocabulary**), which contains nonterminals and terminals.
- Σ (the set of **terminals**) is a subset of V,
- R (the set of **rules**) is a finite subset of $(V - \Sigma) \times V^*$,
- S (the **start symbol**) is an element of $V - \Sigma$.

Rules are also called **productions**.

Example:

$(\{S, \mathrm{a}, \mathrm{b}\}, \{\mathrm{a}, \mathrm{b}\}, \{S \rightarrow \mathrm{a}S\mathrm{b}, S \rightarrow \varepsilon\}, S)$

Note: Some authors say that a CFG is (N, Σ, R, S), where N is the set of nonterminal symbols. In that case $V = N \cup \Sigma$. I may sometimes use N in this way. $N = V - \Sigma$.

Q1-2

Derivations

$x \Rightarrow_G y$ iff $x = \alpha A \beta$

and $A \to \gamma$ is in R

$y = \alpha \gamma \beta$

$w_0 \Rightarrow_G w_1 \Rightarrow_G w_2 \Rightarrow_G \ldots \Rightarrow_G w_n$ is a **derivation** in G.

Let $\Rightarrow_G{}^*$ be the reflexive, transitive closure of $\Rightarrow_G$.

Then the **language generated by *G***, denoted $L(G)$, is:

$\{w \in \Sigma^* : S \Rightarrow_G{}^* w\}$.

A language L is ***context-free*** if there is some context-free grammar G such that $L = L(G)$.

Q3

An Example Derivation

Example:

Let $G = (\{S, \mathrm{a}, \mathrm{b}\}, \{\mathrm{a}, \mathrm{b}\}, \{S \to \mathrm{a}\, S\, \mathrm{b}, S \to \varepsilon\}, S)$

$S \Rightarrow \mathrm{a}\, S\, \mathrm{b} \Rightarrow \mathrm{aa}\, S\, \mathrm{bb} \Rightarrow \mathrm{aaa}\, S\, \mathrm{bbb} \Rightarrow \mathrm{aaabbb}$

So we can write $S \Rightarrow^* \mathrm{aaabbb}$

Regular Grammars

A brief side-trip into Chapter 7

Regular Grammars

In a regular grammar, every rule in R must have a right-hand side that is:

- ε, or
- a single terminal, or
- a single terminal followed by a single nonterminal.

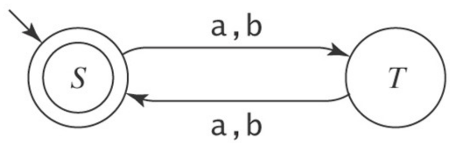
Regular: $S \rightarrow \mathtt{a}$, $S \rightarrow \varepsilon$, and $T \rightarrow \mathtt{a}S$

Not regularl: $S \rightarrow \mathtt{a}S\mathtt{a}$ and $\mathtt{a}S\mathtt{a} \rightarrow T$

Q4

Regular Grammar Example

$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : |w| \text{ is even}\}$ $\quad ((\texttt{aa}) \cup (\texttt{ab}) \cup (\texttt{ba}) \cup (\texttt{bb}))^*$



$S \rightarrow \varepsilon$
$S \rightarrow \texttt{a}T$
$S \rightarrow \texttt{b}T$
$T \rightarrow \texttt{a}$
$T \rightarrow \texttt{b}$
$T \rightarrow \texttt{a}S$
$T \rightarrow \texttt{b}S$

Regular Languages and Regular Grammars

Theorem: A language is regular iff it can be defined by a regular grammar.

Proof: By two constructions (first one on next slide)

Regular Languages and Regular Grammars

Regular grammar* → *FSM:

grammartofsm($G = (V, \Sigma, R, S)$) =

1. Create in M a separate state for each nonterminal in V.
2. Start state is the state corresponding to S .
3. If there are any rules in R of the form $X \rightarrow w$, for some $w \in \Sigma$, create a new state labeled #.
4. For each rule of the form $X \rightarrow w\ Y$, add a transition from X to Y labeled w.
5. For each rule of the form $X \rightarrow w$, add a transition from X to # labeled w.
6. For each rule of the form $X \rightarrow \varepsilon$, mark state X as accepting.
7. Mark state # as accepting.

FSM* → *Regular grammar: Similar.

Example - Even Length Strings

$S \rightarrow \varepsilon$	$T \rightarrow \mathtt{a}$
$S \rightarrow \mathtt{a}T$	$T \rightarrow \mathtt{b}$
$S \rightarrow \mathtt{b}T$	$T \rightarrow \mathtt{a}S$
	$T \rightarrow \mathtt{b}S$

Construct the FSM

End of regular grammars side-trip

Q5

Recursive Grammar Rules

- A rule is ***recursive*** iff it is $X \rightarrow w_1 Y w_2$, where:
  $Y \Rightarrow^* w_3 X w_4$ for some w_1, w_2, w_3, and w_4 in V^*.
- A grammar G is recursive iff G contains at least one recursive rule.
- **Examples:** $S \rightarrow (S)$ $\qquad$ $S \rightarrow (T)$
  $T \rightarrow (S)$

In general, non-recursive grammars are boring!

Self-Embedding Grammar Rules

- A rule in a grammar G is ***self-embedding*** iff it is :
  $X \rightarrow w_1 Y w_2$, where $Y \Rightarrow^* w_3 X w_4$ and both $w_1 w_3$ and $w_4 w_2$ are in Σ^+.

What is the difference between *self-embedding* and *recursive*?

- A grammar is self-embedding iff it contains at least one self-embedding rule.
- Examples: $S \rightarrow \mathrm{a}S\mathrm{a}$ self-embedding

  $S \rightarrow \mathrm{a}S$ recursive but not self-embedding

  $S \rightarrow \mathrm{a}T$
  $T \rightarrow S\mathrm{a}$ self-embedding

Q6

Where Context-Free Grammars Get Their Power

- If a grammar G is not self-embedding then $L(G)$ is regular.
- If a language L has the property that every grammar that defines it is self-embedding, then L is not regular.

Equal Numbers of `a`'s and `b`'s

Let $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^*: \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w)\}$.

Find a CFG G such that L = L(G)

Q7

Arithmetic Expressions

$G = (V, \Sigma, R, E)$, where

$V = \{+, *, (,), \texttt{id}, E\}$,

$\Sigma = \{+, *, (,), \texttt{id}\}$,

$R = \{$

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow (E)$

$E \rightarrow \texttt{id}$

$\}$

BNF

A notation for writing practical context-free grammars

- The symbol | should be read as "or".

  Example: $S \rightarrow \texttt{a}S\texttt{b} \mid \texttt{b}S\texttt{a} \mid SS \mid \varepsilon$

- Allow a nonterminal symbol to be any sequence of characters surrounded by angle brackets.

  Examples of nonterminals:

  <program>
  <variable>

BNF for a Java Fragment

```
<block>      ::= {<stmt-list>}   |
                 {}

<stmt-list> ::= <stmt>   |
                 <stmt-list> <stmt>

<stmt>       ::= <block>   |
                 while (<cond>) <stmt>   |
                 if (<cond>) <stmt>   |
                 do <stmt> while (<cond>);   |
                 <assignment-stmt>;   |
                 return   |
                 return <expression>   |
                 <method-invocation>;
```

Spam Generation

```
<spc>  → space | . | - | _ | = | : | * | / | ¦ | empty

<Word> → <V> <spc> <I> <spc> <A> <spc> <G> <spc> <R> <spc> <A>
  <V>  → V | v | \/
  <I>  → I | i | ! | ¡ | : | ì | í | ï | î | ì | Í | Ï | Î | ¦ | 1 | 1
  <A>  → A | a | /\ | @ | ^ | Á | Â | À | Å | Ã | á | â | ä | à | å | ã
  <G>  → G | g | & | 6 | 9
  <R>  → R | r | ®

Example production:
  <spc> → -
    <V> → v    <I> ::= !    <A> ::= ä    <G> ::= G    <R> ::= ®    <A> ::= ^
 <Word> → v-!-ä-G-®-^
```

These production rules yield 1,843,200 possible spellings.
How Many Ways Can You Spell V1@gra? By Brian Hayes
American Scientist, July-August 2007
http://www.americanscientist.org/template/AssetDetail/assetid/55592

HTML

```
<ul>
    <li>Item 1, which will include a sublist</li>
        <ul>
            <li>First item in sublist</li>
            <li>Second item in sublist</li>
        </ul>
    <li>Item 2</li>
</ul>
```

A grammar:

/* Text is a sequence of elements.
HTMLtext → *Element HTMLtext* | ε

Element → *UL* | *LI* | … (and other kinds of elements that are allowed in the body of an HTML document)

/* The `<ul>` and `</ul>` tags must match.
UL → `<ul>` *HTMLtext* `</ul>`

/* The `<li>` and `</li>` tags must match.
LI → `<li>` *HTMLtext* `</li>`

English

S → *NP VP*
NP → the *Nominal* | a *Nominal* | *Nominal* | *ProperNoun* | *NP PP*
Nominal → *N* | *Adjs N*
N → cat | dogs | bear | girl | chocolate | rifle
ProperNoun → Chris | Fluffy
Adjs → *Adj Adjs* | *Adj*
Adj → young | older | smart
VP → *V* | *V NP* | *VP PP*
V → like | likes | thinks | shoots | smells
PP → *Prep NP*
Prep → with

Designing Context-Free Grammars

- Generate related regions together.

  A^nB^n

- Generate concatenated regions:

  $A \rightarrow BC$

- Generate outside in:

  $A \rightarrow \mathtt{a}A\mathtt{b}$

Concatenating Independent Languages

Let $L = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^m : n, m \geq 0\}$.

The $\mathtt{c}^m$ portion of any string in L is completely independent of the $\mathtt{a}^n\mathtt{b}^n$ portion, so we should generate the two portions separately and concatenate them together.

$G = (\{S, N, C, \mathtt{a}, \mathtt{b}, \mathtt{c}\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, R, S\}$ where:

$$
\begin{aligned}
R = \{\ & S \rightarrow NC \\
& N \rightarrow \mathtt{a}N\mathtt{b} \\
& N \rightarrow \varepsilon \\
& C \rightarrow \mathtt{c}C \\
& C \rightarrow \varepsilon \\
\}.&
\end{aligned}
$$

$L = \{ a^{n_1}b^{n_1}a^{n_2}b^{n_2}\ldots a^{n_k}b^{n_k}$: $k \geq 0$ and $\forall i\ (n_i \geq 0)\}$

Examples of strings in L: ε, abab, aabbaaabbbabab

Note that $L = \{a^n b^n : n \geq 0\}^*$.

$G = (\{S, M, a, b\}, \{a, b\}, R, S\}$ where:

$R = \{ S \rightarrow MS$
$S \rightarrow \varepsilon$
$M \rightarrow aMb$
$M \rightarrow \varepsilon$
$\}$.

Another Example: Unequal a's and b's

$L = \{a^n b^m : n \neq m\}$

$G = (V, \Sigma, R, S)$, where
$V = \{a, b, S,$ $\}$,
$\Sigma = \{a, b\}$,
$R =$

Simplifying Context-Free Grammars

Remove non-productive and unreachable non-terminals.

Unproductive Nonterminals

removeunproductive(*G*: CFG) =

1. *G′* = *G*.
2. Mark every nonterminal symbol in *G′* as unproductive.
3. Mark every terminal symbol in *G′* as productive.
4. Until one entire pass has been made without any new symbol being marked do:
   For each rule $X \rightarrow \alpha$ in *R* do:
      If every symbol in α has been marked as productive and *X* has not yet been marked as productive then:
         Mark *X* as productive.
5. Remove from *G′* every unproductive symbol.
6. Remove from *G′* every rule that contains an unproductive symbol.
7. Return *G′*.

Unreachable Nonterminals

removeunreachable(G: CFG) =
1. $G' = G$.
2. Mark S as reachable.
3. Mark every other nonterminal symbol as unreachable.
4. Until one entire pass has been made without any new symbol being marked do:
   For each rule $X \rightarrow \alpha A \beta$ (where $A \in V - \Sigma$) in R do:
     If X has been marked as reachable and A has not then:
       Mark A as reachable.
5. Remove from G' every unreachable symbol.
6. Remove from G' every rule with an unreachable symbol on the left-hand side.
7. Return G'.

Proving the Correctness of a Grammar

$A^nB^n = \{a^n b^n : n \geq 0\}$

$G = (\{S, a, b\}, \{a, b\}, R, S)$,

$$R = \{\ S \rightarrow a\,S\,b \\ S \rightarrow \varepsilon \\ \}$$

- Prove that G generates only strings in L.
- Prove that G generates all the strings in L.

Structure

Context free languages:

We care about structure.