MA/CSSE 474
Theory of Computation

Answer Questions about Exam2 problems
Removing Ambiguity
Chomsky, Griebach Normal Forms

(who is he foolin', thinking that there will be time to get to all of this?)

Recap: Derivations of The Smart Cat

- A left-most derivation is:
  S ⇒ *NP VP* ⇒ the *Nominal VP* ⇒ the *Adjs N VP* ⇒
  the *Adj N VP* ⇒ the smart *N VP* ⇒ the smart cat *VP* ⇒
  the smart cat *V NP* ⇒ the smart cat smells *NP* ⇒
  the smart cat smells *Nominal* ⇒ the smart cat smells *N* ⇒
  the smart cat smells chocolate

- A right-most derivation is:
  S ⇒ *NP VP* ⇒ *NP V NP* ⇒ *NP V Nominal* ⇒ *NP V N* ⇒
  NP V chocolate ⇒ *NP* smells chocolate ⇒
  the *Nominal* smells chocolate ⇒
  the *Adjs N* smells chocolate ⇒
  the *Adjs* cat smells chocolate ⇒
  the *Adj* cat smells chocolate ⇒
  the smart cat smells chocolate

Recap: Ambiguity

A grammar is ***ambiguous*** iff there is at least one string in $L(G)$ for which G produces more than one parse tree*.

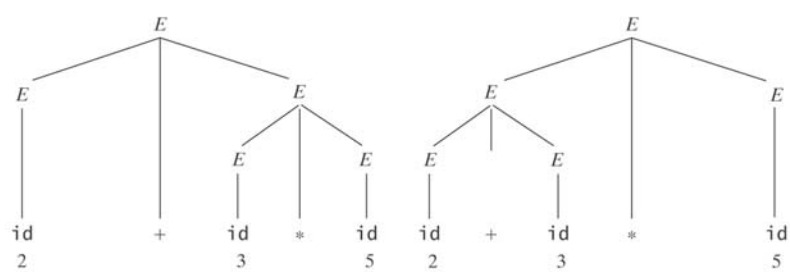
For many applications of context-free grammars, this is a problem.

Example: A programming language.
•If there can be two different structures for a string in the language, there can be two different meanings.
•Not good!

* Equivalently, more than one leftmost derivation, or more than one rightmost derivation.

Recap: Expression Grammar

$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow (E)$
$E \rightarrow \texttt{id}$



Recap:Inherent Ambiguity

Some CF languages have the property that **every** grammar for them is ambiguous. We call such languages ***inherently ambiguous***.

Example:

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

Recap: Inherent Ambiguity

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

One grammar for L has these rules:

$S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow S_1 c \mid A$ /* Generate all strings in $\{a^n b^n c^m\}$.
$A \rightarrow aAb \mid \varepsilon$

$S_2 \rightarrow aS_2 \mid B$ /* Generate all strings in $\{a^n b^m c^m\}$.
$B \rightarrow bBc \mid \varepsilon$

Consider any string of the form $a^n b^n c^n$.

It turns out that L is inherently ambiguous.

Recap: Ambiguity and undecidability

Both of the following problems are undecidable*:

- Given a context-free grammar *G*, is *G* ambiguous?
- Given a context-free language *L*, is *L* inherently ambiguous?

Informal definition of *undecidable* for the first problem:
There is no algorithm (procedure that is guaranteed to always halt) that, given a grammar G, determines whether G is ambiguous.

But We Can Often Reduce Ambiguity

We can get rid of:

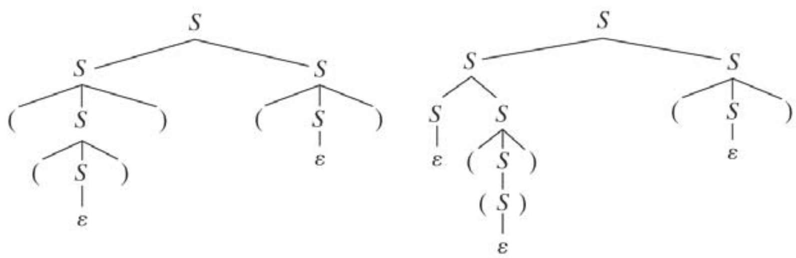
- some ε rules like $B \to \varepsilon$,
- rules with symmetric right-hand sides, e.g.,

  $S \to SS$
  $E \to E + E$

- rule sets that lead to ambiguous attachment of optional postfixes, such as `if … else …`.

A Highly Ambiguous Grammar

$S \rightarrow \varepsilon$
$S \rightarrow SS$
$S \rightarrow (S)$



Resolving the Ambiguity with a Different Grammar

The biggest problem is the ε rule.

A different grammar for the language of balanced parentheses:

$S^* \rightarrow \varepsilon$
$S^* \rightarrow S$
$S \rightarrow SS$
$S \rightarrow (S)$
$S \rightarrow ()$

We'd like to have an algorithm for removing *all* ε-productions except for the case where ε is actually in the language; then we introduce a new start symbol and add one ε-production whose left side is that new symbol.

Nullable Nonterminals

Examples:

$S \rightarrow \text{a}T\text{a}$
$T \rightarrow \varepsilon$

$S \rightarrow \text{a}T\text{a}$
$T \rightarrow A\ B$
$A \rightarrow \varepsilon$
$B \rightarrow \varepsilon$

A nonterminal X is ***nullable*** iff either:
(1) there is a rule $X \rightarrow \varepsilon$, or
(2) there is a rule $X \rightarrow PQR\ldots$ and P, Q, R, … are all nullable.

Nullable Nonterminals

A nonterminal X is ***nullable*** iff either:
(1) there is a rule $X \rightarrow \varepsilon$, or
(2) there is a rule $X \rightarrow PQ\ldots$ where P, Q, … are all nullable nonterminals.

So we compute *Nullable*, the set of nullable nonterminals, as follows:

1. Set *Nullable* to the set of nonterminals that satisfy (1).
2. Repeat until an entire pass is made without adding anything to *Nullable*
   Evaluate all other nonterminals with respect to (2).
   If any nonterminal satisfies (2) and is not in *Nullable*, add it.

A General Technique for Getting Rid of ε-Rules

Definition: a rule is ***modifiable*** iff it is of the form:

$P \to \alpha Q \beta$, for some nullable nonterminal Q.

removeEps(G: cfg) =
1. Let $G' = G$.
2. Find the set *Nullable* of nullable nonterminals in G'.
3. Repeat until G' contains no modifiable rules that haven't been processed:
   Given the rule $P \to \alpha Q \beta$, where $Q \in$ *Nullable*,
   add the rule $P \to \alpha\beta$ if it is not already present
   and if $\alpha\beta \neq \varepsilon$ and if $P \neq \alpha\beta$.
4. Delete from G' all rules of the form $X \to \varepsilon$.
5. Return G'.

Then $L(G') = L(G) - \{\varepsilon\}$

An Example

$G = (\{S, T, A, B, C, \text{a}, \text{b}, \text{c}\}, \{\text{a}, \text{b}, \text{c}\}, R, S)$,

$R = \{ S \to \text{a}T\text{a}$
$T \to ABC$
$A \to \text{a}A \mid C$
$B \to B\text{b} \mid C$
$C \to \text{c} \mid \varepsilon \}$

removeEps(G: cfg) =
1. Let $G' = G$.
2. Find the set N of nullable nonterminals in G'.
3. Repeat until G' contains no modifiable rules that haven't been processed:
   Given the rule $P \to \alpha Q \beta$, where $Q \in N$,
   add the rule $P \to \alpha\beta$
   if it is not already present and if $\alpha\beta \neq \varepsilon$
   and if $P \neq \alpha\beta$.
4. Delete from G' all rules of the form $X \to \varepsilon$.
5. Return G'.

What If $\varepsilon \in L$?

atmostoneEps(G: cfg) =

1. G'' = *removeEps*(G).
2. If S_G is nullable then /* i. e., $\varepsilon \in L(G)$
   - 2.1 Create in G'' a new start symbol S^*.
   - 2.2 Add to $R_{G''}$ the two rules:

     $S^* \rightarrow \varepsilon$

     $S^* \rightarrow S_G$.
3. Return G''.