Continuation of Context-Free Grammars (CFGs)

Example: CFG $G=\left( V, \Sigma, P, S \right)$

$$= \left( \{S\}, \{a,b\}, \{S \rightarrow e | aSb\}, S \right)$$

What is $L(G) = \{x \in \Sigma^* | S \Rightarrow^* x \}$?

Can formally prove what $L(G)$ is.

Informally (done in this course),
"guess" typical form(s) of $x \in \Sigma^*$
with $S \Rightarrow^* x$

Observe:

- $S \Rightarrow e$ (from $S \rightarrow e \in P$)
  $3 \in L(G)$
- $S \Rightarrow aSb$ (from $S \rightarrow aSb \in P$)
  $ab \in L(G)$
  $a^3b = aaabbb \in L(G)$

Similarly, $S \Rightarrow aSb$
- $\Rightarrow aaaSbb$
- $\Rightarrow aaasbbbbb$
- $\Rightarrow aaaaSb bbb$
- $\Rightarrow aaaaSb bbb$

In general, we can see that:
\[
\forall n \geq 0 \quad S \Rightarrow a^n b^n, \text{ i.e., } a^n b^n \in L(G)
\]

\[
\begin{align*}
S & \Rightarrow a S b \\
    & \Rightarrow a a S b b \\
    & \vdots \\
    & \Rightarrow a^n S b^n \\
    & \Rightarrow a^n b^n
\end{align*}
\]

This can convince us that

\[
\{a^i b^i \mid i \geq 0\} \subseteq L(G).
\]

Can you see that

\[
L(G) \subseteq \{a^i b^i \mid i \geq 0\}
\]

informally?

(Informally, argue that for all \(x \in \Sigma^*\),

if \(S \Rightarrow^* x\) then \(x\) must be of the form \(a^i b^i\)

for some \(i \geq 0\).)

- Exercise.

Recall that a language \(L\) is CF (a CFL)

if \(L = L(G)\) for some CFG \(G\).

For examples of languages \(L\) (context-free),

how do we find CFG \(G\) such that

\(L(G) = L\)?
Heuristics:

<table>
<thead>
<tr>
<th>Programs</th>
<th>Machines/FTNs</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables, identifiers</td>
<td>states</td>
</tr>
<tr>
<td>semantics</td>
<td>semantics</td>
</tr>
</tbody>
</table>

Regular expressions:
- Explore "recursive patterns" within typical strings for the language.
- Set up recursion relating all the patterns.

Grammars/LFGs:
- Explore "recursive patterns" within typical strings for the language.
- Set up production rules relating the variables.
Earlier, we studied two (known) non-regular languages:
\( \{a^i b^i | i \geq 0\} \)

**Example 1:**
This is CF

\[ \{a^2 | i \geq 0\} \text{ non-regular integer squares} \]

but it is **NOT** CF
(studied later)

**Example 2:**
\[ L = \{a^i b^i c^j d^j | i \geq 0, j = 23\} \text{ is context-free.} \]

Strings in \( L \):
- \( c^2 a^2 \)
- \( a b c a \)
- \( a^2 b^2 c^2 a^2 \)
- \( a^7 c^{102} a^{102} \)

Not in \( L \):
- \( e \)
- \( a^2 b^2 c a \)
- \( a^2 b^2 \)
- \( c a^2 a^2 b^2 \)
typical strings in L = \( S \) (the start variable)

\[
\begin{array}{ccc}
a_1 & b_2 & c_3 & a_4 \\
R & R & R & R
\end{array}
\]

same length (\( \geq 3 \))

lengths are "independent"

X \implies w \iff \( a_1 \) is of the form \( a_i^i b_i^i \) where \( i \geq 0 \)

Y \implies w \iff \( a_1 \) is of the form \( b_i^i c_i^i \) where \( i \geq 2 \)

So \( \text{L} \rightarrow XY \) a production rule.

What about production rules for \( X \) and \( Y \)?

Learn from Example 1: "Outside-In"

\[
a^i b^i : \quad \begin{array}{c}
\text{a} \\
\text{X} \\
\text{b}
\end{array}
\]

how to explore recursive pattern(s)?

\[
\begin{array}{c}
a \\
\text{X} \\
b
\end{array}
\]

What is left?

\[
a^{i-1} b^{i-1}
\]

also of the same form!

So \( X \rightarrow aXb \) is a production rule.
How do terminate a recursion?

Think of applying the recursions until we arrive at a "base" — no more recursion possible.

\[ a \Delta a \quad b \Delta b \]

\[ X \to axb | e \]

Similarly, \[ Y \to cYa | ecaq \]

Notation: \[ a^i b^j c^k d^l e^f \]

\[ f \geq 2 \]

So, a desired CFG \( G = (V, \Sigma, P, S) \):

\[ \Sigma = \{a, b, c, d\} \]

\[ V = \{S, X, Y\} \]

\[ P = \{ S \to XY, X \to axb | e \}

\[ Y \to cYa | c^2a^2 \} \]

Semantics/interpretation of \( S, X, Y \):

\[ X \Rightarrow^* w \text{ iff } w \text{ is a string from } a^i b^j \text{ when } i \geq 0 \]

\[ Y \Rightarrow^* w \text{ iff } w \text{ is a string from } c^i a^j \text{ when } i \geq 2 \]

\[ S \Rightarrow^* \text{ only if } w \in L \]

\[ w \text{ is a string from } a^i b^j c^k d^l \text{ when } i \geq 0, j \geq 2 \]
Why/what is "context-free" in the definition?

1. In 1-step derivation:
   \[ \alpha \stackrel{A \beta}{\longrightarrow} \alpha \delta \beta, \text{ if } A \rightarrow \delta \in P \]
   \[ \alpha, \beta: \text{Context of } A \]
   "context-free": free to apply the production rule or rewriting rule \( A \rightarrow \delta \) regardless of the context of \( A \)

2. In the previous example:
   \[ a^i b^i c^i d^i \]
   \[ \text{context of } a \quad \text{context of } d \]

   Informally, CFGs generate CFLs that obey block-structured controls, disjoint or containment.

   \[ \text{Languages} \]

Next example, "context":

\[ \text{draw} \]
Example 3: \[ L = \{ a^i b^j c^i d^j \mid i \geq 0, j \geq 1 \} \]

Strings in \( L \):
- \( bc \)
- \( b^3 c^3 \)
- \( a^2 b c d^2 \)
- \( a^3 b^3 c^3 d^3 \)

Not in \( L \):
- \( \varepsilon \)
- \( a b^2 c d^2 \)
- \( a b^5 c^2 d^2 \)
- \( a^2 b^4 c^6 d^2 \)

Typical strings in \( L \):
- Use start variable

<table>
<thead>
<tr>
<th>( a^i )</th>
<th>( b^k )</th>
<th>( c^j )</th>
<th>( d^l )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a^i )</td>
<td>( b^k )</td>
<td>( c^j )</td>
<td></td>
</tr>
</tbody>
</table>

Use variable for this type of strings:
\[ X \Rightarrow W \text{ if } \text{ } W \text{ is of form } b^i c^i \text{ when } i \geq 1 \]

So: \( S \rightarrow a \text{ } S \text{ } d \text{ } \ldots \text{ more ?} \)
Again, learn from Example 1:

\[ S \rightarrow aSd \quad \text{acting like the basis of the recursion} \quad S \rightarrow aSd \]

Then, with semantics: \( X \Rightarrow w \) if \( w = \text{at least } b^i c^j \text{ when } i \geq 1 \), we add:

\[ X \rightarrow bXc \mid bc \]

so the basis is not \( E \).

So, what are the semantics of \( X \Rightarrow S \)?
What are the 4-tuples of a desired \( CFG \) ?

Example 3': \( L = \{ a^i b^i c^j a^i \mid i \geq 1 \land j \geq 1 \} \)

So, \( S \rightarrow aSa \mid X \) would not work.

\[ X \rightarrow bXc \mid bc \]

Why?

How about at least one such matching pair!

\[ S \rightarrow aS'a \]

\[ S' \rightarrow aS'a \mid X \]

\[ X \rightarrow bXc \mid bc \]
Sometimes, the contents are "hidden".

Example 4: \[ L = \{ a^i b^j c^k \mid i, j, k \geq 0 \} \]

Strings in \( L \):
- \( e \)
- \( a \)
- \( a^2 b c \)
- \( a^3 c^3 \)

Not in \( L \):
- \( a b c \)
- \( b c^2 \)

Typical string: \( s \)

\[
\begin{array}{ccc}
\text{as} & \text{bs} & \text{cs} \\
\hline
\text{viewed as} & \text{as} & \text{bs} & \text{cs} \\
\end{array}
\]

This length = sum of these two lengths.

<table>
<thead>
<tr>
<th>Same length</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>So these two lengths are the same</td>
<td></td>
</tr>
</tbody>
</table>

Same idea used previously:

\[ S \rightarrow aSc \mid X \]
\[ X \rightarrow axb \mid e \]

Semantics of \( X \) and \( S \)?
Example 5: \( L = \{ a^i b^j \mid 1 \leq i < j \} \)

Typical strings: \( a^s \) \( b^s \)

Viewed as:

- Same length
- \( S \) \( \times \) \( B \)

\[ S \rightarrow \times B \]
\[ X \rightarrow a^x b^x \mid \epsilon \]
\[ B \rightarrow B b \mid b \]

\[ B \Rightarrow w \text{ iff } \frac{w}{a} \text{ is a } \lambda \text{ word form } B \text{ when } x = 1 \]

Example 6: \( L = \{ 0^x 1^{2x} \mid x \geq 1 \} \)

Typical string:

Viewed as:

- Same length
- \( S \) \( \times \) \( 1s \)

So:

\[ S \rightarrow \times Y \]
\[ X \Rightarrow w \text{ iff } w = 0^w 1^w \]

What about \( Y \)?

- \( Y \) cannot use "\( Y \) itself"
- Or "production rule" to remember the number of \( X \)-productions
- Is used in generating deriving \( a^i b^i \)
Does it mean that $L$ is not CF?

Not necessarily.

Typical string(s):

$0^s \rightarrow 1^s$  
$0 \rightarrow 0S11 \rightarrow \varepsilon$

See the recursion? If the original string is $0^s 1^s$ and $0^s 1^s \rightarrow 0^x 1^{s-x}$, then the string is of the form $0^x 1^{s-x}$ where $s-x = \frac{1}{2}(s-1)$.

So: $S \rightarrow 0S11 \mid \varepsilon$

Example 7: $L = \{ 0^\alpha 1^\beta \mid \alpha, \beta \geq 0, \alpha \leq 2\beta \}$

Strings in $L$:  
$\varepsilon$, $0^4 1^2$, $0^4 1^3$, $1^5$

Not in $L$:  
$0^3 1$, $0^5 1^2$

Exercise: Find a CFG $G$ generating $L$. i.e. $L(G) = L$.

You may be able to "guess" such a CFL.

Can you reason why your CFG works?
Example: Natural implementation of "or" (but not "and")

\[ L = \{ w \in a^*b^*c^* \mid \#a(w) \leq \max\{\#b(w), \#c(w)\} \} \]

\[ = \{ a^i b^j c^k \mid i \leq \max j, k \} \]

Typical strings:

- \( \varepsilon \)
- \( a \)
- \( a^3 b c^2 \)
- \( b c a \)
- \( a^{10} b^{10} c^{10} \)
- \( a^{10} b^9 c^{12} \)
- \( a^3 b^2 c^2 \)

No recursive view yet!

How about an interpretation of \( i \leq \max j, k \)?

\[ i \leq \max j, k \iff i \leq j \text{ or } i \leq k \]

Important to have "equivalence"
So, now we can view the language
\[ L = \{ a^i b^j c^k \mid i < j \} \cup \{ a^i b^j c^k \mid j < k \} \cup \{ a^i b^j c^k \mid i, j, k \geq 0 \} \]

**CFG \( G_1 \):**

Typical strings: \( as \quad bs \quad cs \)

Two views:

\[
\begin{align*}
S_1 & \rightarrow XC \\
X & \rightarrow aXb \mid \text{"what is basis?"} \\
X & \rightarrow aXb \mid B \\
B & \rightarrow Bb \mid \epsilon \\
C & \rightarrow Cc \mid \epsilon
\end{align*}
\]

\[
\begin{align*}
S_2 & \rightarrow XBC \\
X & \rightarrow aXb \mid \epsilon \\
B & \rightarrow Bb \mid \epsilon \\
C & \rightarrow Cc \mid \epsilon
\end{align*}
\]

So \( X \rightarrow^* \omega \) iff \( \omega \) is a string from \( a^i b^j \) when \( i \leq j \)

So \( C \rightarrow^* \omega \) iff \( \omega \) is a string from \( c^i \) when \( i \geq 0 \)
What about CFG $G_2$. What generator

$$S \rightarrow a^i b^j c^k \mid j \leq k, \ i \geq 0$$

Similar -- exercise.

So, a desired CFG $G$ for generating $L$:

$$S \rightarrow S_1 \mid S_2$$

Start variable of $G_2$  
Start variable of $G_1$  
$(V_1, \Sigma, P_1, S_1)$  
$(V_2, \Sigma, P_2, S_2)$

$$G = (V, \Sigma, P, S)$$

$V = \left\{ S \right\} \cup V_1 \cup V_2$

**Assume that we do not re-use variable names.**

$P = \left\{ S \rightarrow S_1 \mid S_2 \right\} \cup P_1 \cup P_2$

Idea used in this example:
Can be used to show that
No set-theoretic operation "U" preserves context freedom:
All CFLs $L_1$ and $L_2$, $L_1 \cup L_2$ is also CF.
Example 9: \[ L = \{ x \in \{ a, b \}^* \mid x^r = x \} \]

Set of all palindromes over \( \{ a, b \} \)

Strings in \( L \):
- \( \varepsilon \)
- \( a \)
- \( b \)
- \( bb \)
- \( aba \)
- \( baaabhb \)
- \( baab \)

Strings not in \( L \):
- \( ab \)
- \( bba \)
- \( aaabbb \)

Typical strings:

Recursive:

Recursive: \[ \text{This part is also a palindrome!} \]

\[ S \rightarrow aSa \mid bSb \mid \varepsilon \]

Example 10: \[ K = \{ x \in \{ a, b \}^* \mid x^r \neq x \} \]

Notice that \[ K = \overline{L} = \{ a, b \}^* - L \] (\( L \) in Example 9)

Wait! Remember that, in last class notes,

Compare:

we have: \( \{ a, b \}^* - \{ a^i b^i \mid i = j \} \)

\( \neq \{ a^i b^j \mid i \neq j \} \)
\[ K = \{ x \mid \text{CFA}^+ | x' = r \} \]

In Example 9, we have learned that \( L \) is CF,
so, maybe \( K = \overline{L} \)

(complementation of CF
is CF ??)

No, in general,
complementation does not
preserve context-freeness
(studied later)

But, for this specific case, \( \overline{L} = K \) is CF!

Strings in \( K \):
- \( a \ b \)
- \( b \ a \ a \ b \ a \ b \)
- \( a \ b \ b \ b \ a \ b \ b \ a \)

Strings not in \( K \):
- \( \epsilon \)
- \( a \)
- \( b \)
- \( a a \)
- \( a b a \)

Typical strings:

\[ \begin{array}{cccc}
\text{a} & \text{A} & \text{A} & \text{b} \\
\text{b} & \text{A} & \text{A} & \text{b} \\
\text{a} & \text{A} & \text{A} & \text{a} \\
\end{array} \]

There must exist a pair (at mirror positions) that are non-matching.

Maybe more than one such pair.
So: \( S \rightarrow aSa | bSb | \text{"even come to non-matching"} \)

\[ S \rightarrow aSa | bSb | N \]

\[ N \rightarrow a ? b | b ?? a \]

What should be \( ? \) or \( ?? \)?

Anything!! (since the existence of the non-matching pair are identified already)

\[ ? \rightarrow ? a | ? b | \epsilon \]

?? - same as ??

Semantics of \( S, N, ? \)
Example 11: \( L = \{ aw \mid w \in \{a,b\}^* \} \)

The language in Example 10 is "stack language".
The language (Example 11) is "queue language".

\( L \) is not CF. (Study later.)

Your current homework has an exercise about the complement of \( L \).

Example 12: \[ \text{[Sip12] Example 2.3, page 105} \]
Consider the CFG \( G = (\{S, E\}, \{a, b\}, \{S \rightarrow aSb \mid SS \mid \epsilon\}, S) \)

What is \( L(G) \)?

Typical strings generated/derived by \( S \):

\[
\begin{align*}
S & \Rightarrow \epsilon \quad \epsilon \\
S & \Rightarrow aSb \\
& \quad \Rightarrow aSb \epsilon \\
& \quad \Rightarrow aSb \epsilon \\
S & \Rightarrow SS \\
& \quad \Rightarrow SSS \\
& \quad \Rightarrow SS \epsilon
\end{align*}
\]

\( \epsilon \) is complicated?
After more example strings, maybe we guess that
$S \Rightarrow^{*} x \in \{a, b\}^{*}$
iff $x$ is a (well-formed) block-structured string of $A_s$ as are $br$ and $rb$ like in syntactically correct programs
\[
(() ())(()) ((()) ()))()
\]

Two problems:

1. Exactly, what do we mean by (well-formed) block-structured...?
   (if we cannot formulate this notion precisely ("mechanically"), then how can we have written a compiler that detects them!?)

2. Even, with a well-defined definition of "block-structured...", how can we convince ourselves that the given C&G G generating such language?
   "Nothing more! Nothing less!"

Exercises.