Context-free grammars and languages

Basics of string generation methods
What’s so great about regular expressions?

A regular expression is a string representation of a regular language.

This allows the storing a whole (eventually infinite) language as a (single) string.
Example

Let \( L = \{ w : w \in \{0,1\}^* \land w \text{ ends with } 1 \} \)

The regular expression

\[(0 + 1)^* \circ 1\]

describes completely this language.

Therefore \( L \), which is an infinite language, can be "stored" by storing \( R \) (a string of length 8).
A language that is not regular

Theorem 8.1: The language over the alphabet \( \{0, 1\} \)
\[
L = \{ w : (\exists n \in N) \land w = 0^n 1^n \} \text{ is not regular.}
\]

Proof:
By contradiction.
Assume that \( L \) is regular. Since \( L \) is infinite, its regular expression \( R \) must contain the star closure of a non-empty, non-null regular expression \( S \). Since the basic (\( i.e. \) alphabet) symbols are 0 and 1, such expression must be either:
\( 0^*, 1^*, (0+1)^*, \) or \((0\cdot1)^* \).

Now, since \( L((0+1)^*) = \{0, 1\}^* \), \((0+1)^* \) produces more string than those in \( L \), and therefore \( S \neq (0+1)^* \).
Proof (cont.)

Since \( L((0 \cdot 1)^*) = \{ \lambda, 01, 0101, 010101, \ldots \} \), \((0 \cdot 1)^*\) produces strings that are not in \( L \). Therefore, \( S \neq (0 \cdot 1)^* \).

Assume that \( S = 0^* \). Then, the language:

\[
M = \{ w : (\exists x, y \in \{0,1\}^*) \land (\forall n \in N) w = x0^n y \}
\]

must be a sub-language of \( L \). But this is not the case, as well.

Similarly, if \( S = 1^* \), the language:

\[
Q = \{ w : (\exists x, y \in \{0,1\}^*) \land (\forall n \in N) w = x1^n y \}
\]

must be a sub-language of \( L \). This is also false.

Therefore, there is no regular expression for \( L \). This contradicts the assumption that \( L \) is regular.
Motivating grammars

The language

\[ L = \{ w : w \in \{0,1\}^* \land (\exists n \in \mathbb{N}) \land w = 0^n1^n \} \]

Shows the existence of non-regular languages.

Can they be described in a single string?

The answer comes with the introduction of grammars. Grammars are designed for:

- describing,
- analyzing and **generating** languages
Definition of context-free grammar

**Definition 8.1:** A context-free grammar (CFG) is a four-tuple $G = (V, A, R, S)$ where

1. $V$ is a finite set of symbols called variables;
2. $A$ is an alphabet
3. $V \cap A = \emptyset$
4. $R \subseteq V \times (V \cup A)^*$ is the set of rules $(v, w) \in R$ is usually denoted $v \to w$ and called a production
5. $S \in V$ is the starting or initial variable.
Example

Let $G = (\{S\}, \{0, 1\}, R, S)$

where

$$R = \{ S \to 0S1 | 01 \}$$

Here | means “or”. Thus,

$$S \to 0S1 | 01 \iff S \to 0S1 \lor S \to 01$$

And thus,

$$R = \{ S \to 0S1, S \to 01 \}$$
The string derivation process

The generation of a string with CFGs is called **derivation**

Strings are **derived** as follows:

1. Select a production whose left hand side is the **start variable**. Ex: $S \rightarrow 0S1$

2. **Select a variable** in the rightmost side of this production. Ex: in the previous CFG we have no choice but $S$, as $S$ is the only variable.
3. **Select a production initiated with** the chosen variable. Ex: Assume that we choose:

\[ S \rightarrow 01 \]

4. **Replace the selected variable with the rightmost side of the selected production.** Ex: Choice 3. yields

\[ S \rightarrow 0S1 \rightarrow 0011 \]

5. Continue until **no variables are left**. In the example, no variables are left. The derivation ends with 0011
Denoting derivations

Let $G = (V, A, R, S)$ be a context-free grammar. Assume that $w \in (V \cup A)^*$ is a string derived from $G$ through a sequence of $n$ productions.

Then, we write

$$S \xrightarrow[G,n]{} w$$

or, if the number of productions is not important,

$$S \xrightarrow[G]{} w$$
Example

With the CFG of the previous example:

\[ S \rightarrow 0S1 \]

\[ \rightarrow 00S11 \text{ (after replacing } S \text{ with } S \rightarrow 0S1) \]

\[ \rightarrow 000111 \text{ (after replacing } S \text{ with } S \rightarrow 01) \]

These derivations are denoted:

\[ S \xrightarrow[1]{G} 0S1, \]

\[ S \xrightarrow[2]{G} 00S11, \text{ and} \]

\[ S \xrightarrow[3]{G} 000111; \text{ respectively.} \]
The language of a context-free grammar

**Definition 8.1:** The language of a context-free grammar \( G = (V, A, R, S) \), denoted \( L(G) \), is the set of all strings in \( A^* \) that are generated (this is, derived with) by \( G \). Thus,

\[
L(G) = \{ w : w \in A^* \land (\exists n \in N) \land S \xrightarrow[G,n]{} w \}
\]

**Definition 8.2:** If a language over an alphabet \( A \) can be generated by a context-free grammar, then the language is said to be a context-free language.
Representing context-free languages

Since a CFG is a four-tuple of finite mathematical objects, it has string representations. Thus, context-free languages can be stored (without loss of information) as a string representing its context-free grammar.

Unlike regular languages whose regular expressions are easy to find, either finding a context-free grammar for a language or identifying the language of a context-free grammar, may be much more complex.
A simple example of language identification

**Theorem 8.2:** The language of \( G = (\{S\}, \{0,1\}, \{S \rightarrow 0S1, S \rightarrow 01\}, S) \) is \( W = \{w : (\exists n \in N) \wedge (n \geq 1) \wedge w = 0^n1^n\} \)

**Proof:**

Recall that \( L(G) = \{w : w \in A^* \wedge (\exists n \in N) \wedge S \xrightarrow[G,n]{} w\} \)

To show: (a) \( W \subseteq L(G) \wedge (b) L(G) \subseteq W \)

Proof of (a):

We need to demonstrate the predicate: \( (\forall n \in N)Q(n) \), where

\( Q(n) : " n \geq 1 \Rightarrow (\exists \text{ a derivation}) S \xrightarrow[G,n]{} 0^n1^n " \)

Assume first that \( n = 1 \). Then, since the production \( S \rightarrow 01 \in R \), There is indeed a derivation \( S \xrightarrow[G,1]{} 01 \).
**Proof (cont.)**

Assume \( n \geq 2 \). We show \( Q(n) \) by induction on the auxiliary predicate:

Let \( P(n) :'' S \xrightarrow{G,n} 0^n S1^n '' \)

Base case: \( n = 1 \)

Since \( S \rightarrow 0S1 \in R \), there exists the derivation \( S \xrightarrow{G,1} 0S1 \)

Inductive hypothesis: \( (\exists n > 1) P(n) \)

Inductive thesis: \( P(n) \Rightarrow P(n+1) \)

Proof: By hypothesis, there exists the derivation \( S \xrightarrow{G,n} 0^n S1^n \).

Since \( S \rightarrow 0S1 \in R \), there exists the derivation

\( S \xrightarrow{G,n+1} 0^{n+1} S1^{n+1} \), as well.

Now, by applying \( S \rightarrow 01 \in R \) we get

\( (\forall n \in N)(n \geq 2) \Rightarrow S \xrightarrow{G,n-1} 0^{n-1} S1^{n-1} \rightarrow 0^n1^n \)

Thus, \( (\forall n \in N)(n \geq 1)Q(n) \) is a true predicate.
Proof (cont.)

Proof of (b): It will be shown that $G$ cannot generate strings that are not in $W$. Observe first that $G$ has two rules starting with the start variable, namely $S \rightarrow 01$ and $S \rightarrow 0S1$. Therefore, $G$ has only two substitutions. Thus,

$$(\forall w)w = w_1 \cdots w_n \in (A \cup V)^* \land (\forall i)(i = 1, \ldots, n) \land w_i = S$$

$w \rightarrow w_1 \cdots w_{i-1} 01w_{i+1} \cdots w_n \lor w_1 \cdots w_{i-1} 0S1w_{i+1} \cdots w_n.$

We will demonstrate by induction that there is at most one variable $S$ per string derived by $G$. This is, we'll demonstrate:

$P(n): \text{"} S \xrightarrow{G,n} w \Rightarrow w \text{ has one symbol } S \text{ or none } \text{"}.$

Base case: $n = 1$. In this case, clearly,

$S \xrightarrow{G,1} 01 \lor S \xrightarrow{G,1} 0S1.$

Inductive hypothesis: $(\exists n \geq 1) \land S \xrightarrow{G,n} w \Rightarrow w \text{ has one symbol } S \text{ or none}.$

Inductive thesis: $S \xrightarrow{G,n+1} w \Rightarrow w \text{ has one symbol } S \text{ or none}.$
Proof (cont.)

Proof of the induction: Assume that

\[ S \xrightarrow{G,n+1} w = w_1 \cdots w_m \land (\exists i)(1 < i < m) \land w_i = S. \]

Then, there exists \( \overline{w} \in (A \cup V)^* \), such that

\[ \overline{w} \xrightarrow{} w \] using \( S \xrightarrow{} 0S1 \) to substitute its own \( S \).

This is so, because the only other possible production

\( i.e. S \xrightarrow{} 01 \)

does not preserves \( S \).

But then, because of the form of the production,

\[ w_{i-1} = 0 \land w_{i+1} = 1. \]

Therefore,

\[ \overline{w} = w_1 \cdots w_{i-2}Sw_{i+2} \cdots w_m \land S \xrightarrow{G,n} \overline{w} \Rightarrow S \] is the only start

variable in \( \overline{w} \).

Therefore, \( w \) has only one start variable.
Parsing regular expressions

Is the same as analyzing a regular language: a recursive process for identifying the expression’s basic regular components. The recursion decomposes first the lowest hierarchy operation

Ex: The analysis of $a + b^* \circ c$ is represented in the tree
Method 8.1:
Context-free grammar generating the language of a regular expression

1. Write the parsing tree of the regular expression
2. Replace the root and each internal node that is not an operation, with a variable
3. Keep the terminals as the leaves, except for terminals under a sub-tree rooted by a variable \( V \) having the operation * as child. In these cases, replace the terminal \( a \), with the string \( aU \), where \( U \) is a variable
4. Write the rules as follows:
   4.1 The root is the starting variable, and each parent-children edge correspond to a rule. For building these rules:
   4.2 Assign to each triple of sibling nodes \( V, \circ, U \) the string \( VU \)
   4.3 Assign to each triple of sibling nodes \( V, +, U \) the string \( V / U \)
   4.4 For each node \( V \) having \( U \) and * as children, create \( V \rightarrow \lambda / U \)
   4.5 For each leaf \( aU \) in a sub-tree rooted by a variable \( V \) having * as child, create \( U \rightarrow V \) and the rule \( U \rightarrow \lambda \)
Example 1: Consider $R = 0+1^* \circ 0+1$

Parsing tree after replacements

Corresponding set of rules:

\[
R = \{ \\
S \rightarrow V_1 \mid 1, \\
V_1 \rightarrow 0 \mid V_2, \\
V_2 \rightarrow V_3 0, \\
V_3 \rightarrow 1V_4 \mid \lambda, \\
V_4 \rightarrow V_3 \mid \lambda \}
\]
Two productions with the previous grammar

The resulting grammar is:

\[ G=\left(\{S,V_1,\ldots,V_7\},\{0,1\}, R, S\right) \]

Where \( R \) is the set of rules defined in slide 21

- The strings \( w=0 \) and \( w=1 \), which are in \( L(0+1^*\circ 0+1) \), are produced by \( G \) as follows:
  \[ S \to 1 \]
  \[ S \to V_1 \to 0 \]

- The string \( w=1110 \), which is also in \( L(0+1^*\circ 0+1) \), is produced by \( G \) as follows:
  \[ S \to V_1 \to V_2 \to V_3 \to 0 \to 1V_4 \to 0 \to 1V_3 \to 0 \to 11V_4 \to 0 \to 1110 \]
Definition: The parsing of a given string is the generation of a tree representation of the derivation of the string by a grammar.

- The tree representing the string generation is called parse tree.

Example: Parse tree for $w=1110$ under the previous grammar.
Grammars vs. regular expressions

One can (easily) be convinced that the language of the grammar $G$ in the previous construction is the language of the regular expression. This is, $L(G)=L(0+1^* \circ 0+1)$

Q: Does this “double representation” constitutes a redundancy?
A: NO!. Unlike a regular expression whose role is merely descriptive, the grammar built by Method 8.1, (as any other grammar) constitutes (the basis of) a computational method (not necessarily the most efficient) for generating the strings of a language described by a regular expression

INDEED, underlying grammars is the idea of storing a language not as a (static) mathematical description but as a computational method

BE FULLY AWARE OF THIS CONCEPTUAL FACT!!!!
A fundamental theorem

**Theorem:** Every regular language is a context-free language

**Scheme of the proof:**
Let $L$ be a regular language and $w \in L$. Since $L$ is regular, $L$ is completely described by a regular expression $R$.

Let $G$ be the grammar built from the parsing of $R$.

Build the parsing tree of $w$ under $R$. Use this parsing tree to show that $S \xrightarrow{G} w$. This shows that $L \subseteq L(G)$.

If, on the other hand, $L \not\subseteq L(G)$, then $(\exists w \in L)(\neg(S \xrightarrow{G} w))$.

Then, there is no parse tree for $w$ in terms of the grammar $G$.

This implies that $w$ is not described by $R$. But this is a contradiction with the assumption that $w \in L$. 
