Grammars

- Grammars play a crucial role in programming languages because they precisely capture the syntactic nature of programming languages.
- We start our discussion of grammars by looking at the nature of sequences of symbols, where sequences of symbols form the foundation of any language, both natural and artificial.
- We will call sequences of symbols *strings*.

Strings

Definition: [Strings over an Alphabet][1]

- An *alphabet* is a finite, nonempty set – we refer to the elements of an alphabet as *symbols*.
- A finite sequence of symbols from a given alphabet is called a *string over the alphabet*.
- A string that consists of a sequence $a_1, a_2, \ldots, a_n$ of symbols is denoted by the juxtaposition $a_1 a_2 \ldots a_n$.
- The length of some string s is denoted by $|s|$ and assumed to equal the number of symbols in the string.
- Strings that have zero symbols, called *empty strings*, are denoted by ϵ with $|\epsilon| = 0$.

[1] Based on material from the book "An Introduction to the Theory of Computation," Eitan Gurari, Ohio State University,Computer Science Press, 1989.

Strings

Example: Let $\Gamma_1 = \{a, \ldots, z\}$ and $\Gamma_2 = \{0, \ldots, 9\}$ is alphabets. *abb* is a string over Γ_1, and 123 is a string over Γ_2. *ba*12 is neither a string over Γ_1 nor a string over Γ_2 but it is a string over $\Gamma_1 \cup \Gamma_2$. The string $314\ldots$ is not a string over Γ_2, because it is not a finite sequence.

Some other observations:

- The empty string ϵ is a string over any alphabet.
- The empty set $\emptyset$ is not an alphabet because it contains no element.
- The set of natural numbers is not an alphabet, because it is not finite.

Strings

Definition: [Kleene Closure] Given some alphabet Γ then the set of all possible strings over Γ including the empty string ϵ is denoted by Γ^* and is called the *Kleene Closure of* Γ. (Similar to the power set construction with the exception that the constructed set is always infinite.)

Example: Let $\Gamma = \{a\}$, then $\Gamma^* = \{\epsilon, a, aa, aaa, aaaa, \dots\}$.

Example: Let $\Gamma = \{a, b\}$, then

$$\Gamma^* = \{\epsilon, a, b, aa, bb, ab, ba, aaa, aab, \dots\}.$$

Strings

Definition: [String Concatenation] Given some alphabet Γ with $s_1 \in \Gamma^*$ and $s_2 \in \Gamma^*$, then the *concatenation of the strings* written as $s_1 s_2$ also belongs to Γ^*, that is the string $s_1 s_2 \in \Gamma^*$.

Definition: [Context-Free Grammar] A *context-free grammar* is a triple $(\Gamma, \rightarrow, \gamma)$ such that,

- $\Gamma = T \cup N$ with $T \cap N = \emptyset$, where T is a set of symbols called the *terminals* and N is a set of symbols called the *non-terminals*,[2]
- $\rightarrow \subseteq \Gamma^* \times \Gamma^*$ is a set of rules of the form $u \rightarrow v$ with $u \in N$ and $v \in \Gamma^*$,
- γ is called the *start symbol* and $\gamma \in N$.

[2] The fact that T and N are non-overlapping means that there will never be confusion between terminals and non-terminals.

Grammars

Example: Grammar for arithmetic expressions. We define the grammar $(\Gamma, \rightarrow, \gamma)$ as follows:

- $\Gamma = T \cup N$ with $T = \{a, b, c, +, *, (,)\}$ and $N = \{E\}$,
- $\rightarrow$ is is defined as,

$$\begin{array}{lcl} E & \rightarrow & E + E \\ E & \rightarrow & E * E \\ E & \rightarrow & (E) \\ E & \rightarrow & a \\ E & \rightarrow & b \\ E & \rightarrow & c \end{array}$$

- $\gamma = E$ (clearly this satisfies $\gamma \in N$).

Rewriting Relation

In order for a grammar $(\Gamma, \rightarrow, \gamma)$ to be useful we allow rules to be applied to *substrings* of given strings; let $s = xuy, t = xvy$ with $x, y, u, v \in \Gamma^*$, and a rule $u \rightarrow v$, then we say that s *rewrites to* t and as before we write,

$$s \Rightarrow t.$$

More formally,

Definition: [one-step rewriting relation] Let $(\Gamma, \rightarrow)$ be a string rewriting system, then the *one-step rewriting relation* $\Rightarrow \subseteq \Gamma^* \times \Gamma^*$ is the set with $(s, t) \in \Rightarrow$ for strings $s, t \in \Gamma^*$ if and only if there exist $x, y, u, v \in \Gamma^*$ with $s = xuy, t = xvy$, and $u \rightarrow v$.

In plain English: any two string s, t belong to the relation $\Rightarrow$ if and only if they can be related by a rewrite rule.

Rewriting Relation

With grammars, derivations always start with the start symbol $\gamma \in \Gamma^*$. Consider,

$$E \Rightarrow E{*}E \Rightarrow (E){*}E \Rightarrow (E{+}E){*}E \Rightarrow (a{+}E){*}E \Rightarrow (a{+}b){*}E \Rightarrow (a{+}b){*}c.$$

Here, $(a + b) * c$ is a *normal form* often also called a *terminal*- or *derived-string*.
We often write,

$$E \Rightarrow^* (a + b) * c$$

stating that the normal form is derivable from the start symbol in zero or more steps.

Exercise: Identify the rule that was applied at each rewrite step in the above derivation.
Exercise: Derive the string $((a))$.
Exercise: Derive the string $a + b * c$.

Grammars

We are now in the position to define exactly what we mean by a *language*.

Definition:[Language] Let $G = (\Gamma, \rightarrow, \gamma)$ be a grammar, then we define the *language of grammar G* as the set of all terminal strings that can be derived from the start symbol s by rewriting using the rules in $\rightarrow$. Formally,

$$L(G) = \{q \mid \gamma \Rightarrow^* q \wedge q \in T^*\}.$$

Example: Let $J = (\Gamma, \rightarrow, \gamma)$ be the grammar of Java, then $L(J)$ is the set of all possible Java programs. The Java language is defined as the set of all possible Java programs.

Observations:

- With the concept of a language we can now ask interesting questions. For example, given a grammar G and some sentence $p \in T^*$, does p belong to $L(G)$?
- If we let J be the grammar of Java, then asking whether some string $p \in T^*$ is in $L(J)$ is equivalent to asking whether p is a *syntactically correct program*.

Grammars

Example: A simple imperative language. We define grammar $G = (\Gamma, \rightarrow, \gamma)$ as follows:

- $\Gamma = T \cup N$ where

$$T = \{\mathbf{0}, \ldots, \mathbf{9}, \mathbf{a}, \ldots, \mathbf{z}, \mathbf{true}, \mathbf{false}, \mathbf{skip}, \mathbf{if}, \mathbf{then}, \mathbf{else}, \mathbf{while}, \mathbf{do}, \mathbf{end} +, -, *, =, \leq, !, \&\&, ||, :=, ;, (,)\}$$

  and

$$N = \{A, B, C, D, L, V\}.$$

- The rule set $\rightarrow$ is defined by,

$$\begin{array}{lcl}
\mathrm{A} & \rightarrow & D \mid V \mid \mathrm{A} + \mathrm{A} \mid \mathrm{A} - \mathrm{A} \mid \mathrm{A} * \mathrm{A} \mid (\mathrm{A}) \\
\mathrm{B} & \rightarrow & \mathbf{true} \mid \mathbf{false} \mid \mathrm{A} = \mathrm{A} \mid \mathrm{A} \leq \mathrm{A} \mid !\mathrm{B} \mid \mathrm{B}\&\&\mathrm{B} \mid \mathrm{B}||\mathrm{B} \mid (\mathrm{B}) \\
\mathrm{C} & \rightarrow & \mathbf{skip} \mid \mathrm{V} := \mathrm{A} \mid \mathrm{C}\ ;\ \mathrm{C} \mid \mathbf{if}\ \mathrm{B}\ \mathbf{then}\ \mathrm{C}\ \mathbf{else}\ \mathrm{C}\ \mathbf{end} \mid \mathbf{while}\ \mathrm{B}\ \mathbf{do}\ \mathrm{C}\ \mathbf{end} \\
\mathrm{D} & \rightarrow & \mathrm{L} \mid -\mathrm{L} \\
\mathrm{L} & \rightarrow & \mathbf{0}\,\mathrm{L} \mid \ldots \mid \mathbf{9}\,\mathrm{L} \mid \mathbf{0} \mid \ldots \mid \mathbf{9} \\
\mathrm{V} & \rightarrow & \mathbf{a}\,\mathrm{V} \mid \ldots \mid \mathbf{z}\,\mathrm{V} \mid \mathbf{a} \mid \ldots \mathbf{z}
\end{array}$$

- $\gamma = \mathrm{C}$.

Grammars

Here are some elements in $L(G)$,

$$
\begin{array}{l}
x := 1; y := x \\
v := 1; \textbf{if } v \leq 0 \textbf{ then } v := (-1) * v \textbf{ else skip end} \\
n := 5; f := 1; \textbf{while } 2 \leq n \textbf{ do } f := n * f; n := n - 1 \textbf{ end}
\end{array}
$$

Exercise: Prove that they belong to $L(G)$.

Assignment #1 – see BrightSpace