# Grammars

**Observations:**

- We have seen in the case of the palindrome generator that SRSs are well suited for generating strings with structure.
- By modifying the standard SRS just slightly we obtain a convenient framework for generating strings with desirable structure – *Grammars*

**Definition:** [Grammar] A *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*,[1]
- $\rightarrow$ is a set of rules of the form $u \rightarrow v$ with $u, v \in \Gamma^*$,
- $\gamma$ is called the *start symbol* and $\gamma \in N$.

---

[1] 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 $\gamma \in N$).

With grammars, derivations always start with the start symbol. 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*.

**Exercise:** Identify the rule that was applied at each rewrite step in the above derivation.
**Exercise:** Derive the string $((a))$.
**Exercise:** Derive the string $a + b * c$. Is the derivation unique? Why? Why not?

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,

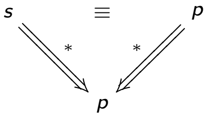
$$L(G) = \{q \mid \gamma \Rightarrow^* q \wedge q \in T^*\}.$$

**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.

# Grammars

**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*.
- We can prove language membership by showing that the start symbol is equivalent to the sentence in question,



# Grammars

**Observations:**

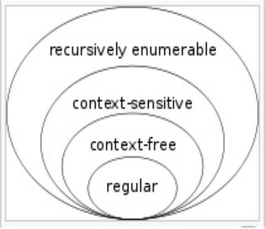
- By restricting the shape of the rewrite rules in a grammar we obtain different language *classes*.
- The most famous set of language classes is the *Chomsky Hierarchy*.

# Grammars

Table: The Chomsky Hierarchy

| Rules | Grammar | Language | Machine |
|---|---|---|---|
| $\alpha \longrightarrow \beta$ | Type-0 | Recursively Enumerable | Turing machine |
| $\alpha A \beta \longrightarrow \alpha \gamma \beta$ | Type-1 | Context-sensitive | Linear-bounded Turing machine |
| $A \longrightarrow \gamma$ | Type-2 | Context-free | Pushdown automaton |
| $A \longrightarrow a$ and $A \longrightarrow aB$ | Type-3 | Regular | Finite state automaton |

where $\alpha, \beta, \gamma \in \Gamma^*$, $A, B \in N$, $a \in T$. In Type-1 $\gamma$ is not allowed to be the empty string.



**Observation:** The most convenient language class for programming language specification are the context-free languages – they are decidable – pushdown automata can be efficiently implemented in order to prove language membership.

# 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,

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

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

Observe that this is a context-free grammar!

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

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

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

# Grammars

HW#1 – see website