

Theory of Computer Science

B2. Grammars

Gabriele Röger

University of Basel

March 10, 2021

Theory of Computer Science

March 10, 2021 — B2. Grammars

B2.1 Introduction

B2.2 Grammars

B2.3 Chomsky Hierarchy

B2.4 Summary

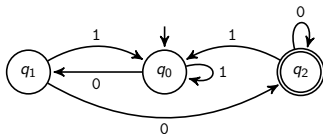
B2.1 Introduction

Reminder: Alphabets and Formal Languages

- ▶ An **alphabet** Σ is a finite non-empty set of **symbols**.
- ▶ A **word over** Σ is a finite sequence of elements from Σ .
- ▶ The **empty word** is denoted by ε .
- ▶ Σ^* denotes the set of **all words** over Σ .
- ▶ Σ^+ denotes the set of **all non-empty words** over Σ .
- ▶ A **formal language** (over alphabet Σ) is a subset of Σ^* .

Reminder: Finite Automata and Formal Languages

Example



The DFA recognizes the language $\{w \in \{0, 1\}^* \mid w \text{ ends with } 00\}$.

- ▶ A finite automaton defines a language, the language it **recognizes**.
- ▶ The specification of the automaton is always finite.
- ▶ The recognized language can be infinite.

Other Ways to Specify Formal Languages?

Sought: General concepts to define
(often infinite) formal languages
with finite descriptions.

- ▶ **today:** grammars
- ▶ **later:** more automata, regular expressions, ...

Grammar: Example

Variables $V = \{S, X, Y\}$

Alphabet $\Sigma = \{a, b, c\}$.

Production rules:

$$\begin{array}{lll} S \rightarrow \varepsilon & X \rightarrow aXYc & cY \rightarrow Yc \\ S \rightarrow abc & X \rightarrow abc & bY \rightarrow bb \\ S \rightarrow X & & \end{array}$$

You start from S and may in each step replace the left-hand side of a rule with the right-hand side of the same rule. This way, derive a word over Σ^* .

Exercise

Variables $V = \{S, X, Y\}$

Alphabet $\Sigma = \{a, b, c\}$.

Production rules:

$S \rightarrow \varepsilon$ $X \rightarrow aXYc$ $cY \rightarrow Yc$

$S \rightarrow abc$ $X \rightarrow abc$ $bY \rightarrow bb$

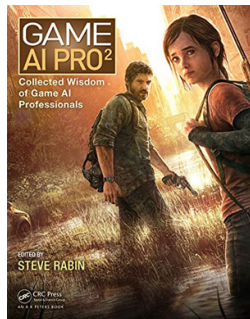
$S \rightarrow X$

Derive word aabbcc starting from S.



Application: Content Generation in Games

- ▶ <http://www.gameapro.com/>
- ▶ GameAIPro 2, chapter 40
**Procedural Content Generation:
An Overview** by Gillian Smith



B2.2 Grammars

Grammars

Definition (Grammars)

A **grammar** is a 4-tuple $\langle V, \Sigma, R, S \rangle$ with:

- ▶ V finite set of **variables** (**nonterminal symbols**)
- ▶ Σ finite alphabet of **terminal symbols** with $V \cap \Sigma = \emptyset$
- ▶ $R \subseteq (V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ finite set of **rules**
- ▶ $S \in V$ **start variable**

A rule is sometimes also called a **production** or a **production rule**.

German: Grammatik, Variablen, Terminalalphabet, Regeln/Produktionen, Startvariable

Rule Sets

What exactly does $R \subseteq (V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$ mean?

- ▶ $(V \cup \Sigma)^*$: all words over $(V \cup \Sigma)$
- ▶ for languages L and L' , their **concatenation** is the language $LL' = \{xy \mid x \in L \text{ and } y \in L'\}$.
- ▶ $(V \cup \Sigma)^* V (V \cup \Sigma)^*$: words composed from
 - ▶ a word over $(V \cup \Sigma)$,
 - ▶ followed by a single variable symbol,
 - ▶ followed by a word over $(V \cup \Sigma)$
 → word over $(V \cup \Sigma)$ containing at least one variable symbol
- ▶ \times : Cartesian product
- ▶ $(V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$: set of all pairs $\langle x, y \rangle$, where
 - x word over $(V \cup \Sigma)$ with at least one variable and
 - y word over $(V \cup \Sigma)$
- ▶ Instead of $\langle x, y \rangle$ we usually write rules in the form $x \rightarrow y$.

Rules: Examples

Example

Let $\Sigma = \{a, b, c\}$ and $V = \{X, Y, Z\}$.

Some examples of rules in $(V \cup \Sigma)^* V (V \cup \Sigma)^* \times (V \cup \Sigma)^*$:

$$X \rightarrow XaY$$

$$Yb \rightarrow a$$

$$XY \rightarrow \varepsilon$$

$$XYZ \rightarrow abc$$

$$abXc \rightarrow XYZ$$

Derivations

Definition (Derivations)

Let $\langle V, \Sigma, R, S \rangle$ be a grammar. A word $v \in (V \cup \Sigma)^*$ can be **derived** from word $u \in (V \cup \Sigma)^+$ (written as $u \Rightarrow v$) if

- 1 $u = xyz$, $v = xy'z$ with $x, z \in (V \cup \Sigma)^*$ and
- 2 there is a rule $y \rightarrow y' \in R$.

We write: $u \Rightarrow^* v$ if v can be derived from u in finitely many steps (i. e., by using n derivations for $n \in \mathbb{N}_0$).

German: Ableitung

Language Generated by a Grammar

Definition (Languages)

The **language generated** by a grammar $G = \langle V, \Sigma, P, S \rangle$

$$\mathcal{L}(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

is the set of all words from Σ^* that can be derived from S with finitely many rule applications.

German: erzeugte Sprache

Grammars

Example (Languages over $\Sigma = \{a, b\}$)

- ▶ $L_1 = \{a, aa, aaa, aaaa, \dots\} = \{a\}^+$
- ▶ $L_2 = \Sigma^*$
- ▶ $L_3 = \{a^n b^n \mid n \geq 0\} = \{\varepsilon, ab, aabb, aaabbb, \dots\}$
- ▶ $L_4 = \{\varepsilon\}$
- ▶ $L_5 = \emptyset$
- ▶ $L_6 = \{w \in \Sigma^* \mid w \text{ contains twice as many } a\text{'s as } b\text{'s}\}$
 $= \{\varepsilon, aab, aba, baa, \dots\}$

Example grammars: blackboard

Exercise

Specify a grammar that generates language

$$L = \{w \in \{a, b\}^* \mid |w| = 3\}.$$



B2.3 Chomsky Hierarchy

Noam Chomsky

- ▶ Avram Noam Chomsky (*1928)
- ▶ "the father of modern linguistics"
- ▶ American linguist, philosopher, cognitive scientist, social critic, and political activist
- ▶ combined linguistics, cognitive science and computer science
- ▶ opponent of U.S. involvement in the Vietnam war
- ▶ there is a wikipedia page solemnly on his political positions



CC BY 2.0 / Andrew Rusk

→ Organized grammars into the **Chomsky hierarchy**.

Chomsky Hierarchy

Definition (Chomsky Hierarchy)

- ▶ Every grammar is of **type 0** (all rules allowed).
- ▶ Grammar is of **type 1** (**context-sensitive**)
if all rules are of the form $\alpha B \gamma \rightarrow \alpha \beta \gamma$
with $B \in V$ and $\alpha, \gamma \in (V \cup \Sigma)^*$ and $\beta \in (V \cup \Sigma)^+$
- ▶ Grammar is of **type 2** (**context-free**)
if all rules are of the form $A \rightarrow w$,
where $A \in V$ and $w \in (V \cup \Sigma)^*$.
- ▶ Grammar is of **type 3** (**regular**)
if all rules are of the form $A \rightarrow w$,
where $A \in V$ and $w \in \Sigma \cup \Sigma V$.

special case: rule $S \rightarrow \varepsilon$ is always allowed if S is the start variable and never occurs on the right-hand side of any rule.

German: Chomsky-Hierarchie, Typ 0, Typ 1 (kontextsensitiv),
Typ 2 (kontextfrei), Typ 3 (regulär)

Chomsky Hierarchy

Definition (Type 0–3 Languages)

A language $L \subseteq \Sigma^*$ is of type 0 (type 1, type 2, type 3) if there exists a type-0 (type-1, type-2, type-3) grammar G with $\mathcal{L}(G) = L$.

Type k Language: Example (slido)

Example

Consider the language L generated by the grammar $\langle \{F, A, N, C, D\}, \{a, b, c, \neg, \wedge, \vee, (,)\}, R, F \rangle$ with the following rules R :

$$F \rightarrow A$$

$$A \rightarrow a$$

$$N \rightarrow \neg F$$

$$F \rightarrow N$$

$$A \rightarrow b$$

$$C \rightarrow (F \wedge F)$$

$$F \rightarrow C$$

$$A \rightarrow c$$

$$D \rightarrow (F \vee F)$$

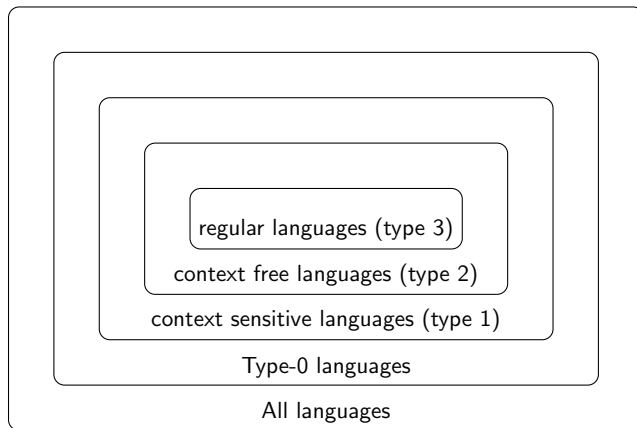
$$F \rightarrow D$$

Questions:

- ▶ Is L a type-0 language?
- ▶ Is L a type-1 language?
- ▶ Is L a type-2 language?
- ▶ Is L a type-3 language?



Chomsky Hierarchy



Note: Not all languages can be described by grammars. ([Proof?](#))

B2.4 Summary

Summary

- ▶ **Languages** are sets of symbol sequences.
- ▶ **Grammars** are one possible way to specify languages.
- ▶ Language **generated** by a grammar is the set of all words (of terminal symbols) **derivable** from the start symbol.
- ▶ **Chomsky hierarchy** distinguishes between languages at different levels of expressiveness.