

Introduction

- We have seen that some languages can be represented by regular expressions, which can be also recognized by finite automata.
- There are languages, such as $\{0^n 1^n \mid n \geq 0\}$ that cannot be described (specified) by finite automata or regular expressions
- Grammars provide a more powerful mechanism for language specification beyond finite automata.

Grammars and Languages

Hantao Zhang

hzhang@cs.uiowa.edu

The University of Iowa, Department of Computer Science

Based on Professor Rus' Lecture Notes

More Terminology

- The specification rules of a grammar are also called *productions* or *substitution rules*
- Some symbols in the rules serve as a wildcard to denote a set of strings and are called *nonterminals* or *variables*, e.g. S and A in G_1 .
- Some symbols in the rules will appear in the language specified by the rules and are called *terminals*, e.g., 0 and 1 in G_1 . They are analogous to the input alphabet of an automaton.

Grammar: Informal

- A grammar consists of a collection of specification rules where one variable is designated as *start symbol*.
- **Example:** The grammar G_1 has the following specification rules:

$$S \rightarrow A$$
$$A \rightarrow 0A1$$
$$A \rightarrow \epsilon$$

Example String Generation

Using CFG G_1 we can generate the string 000111 as follows:

$S \Rightarrow A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000111$

- The sequence of substitutions used to obtain a string using a grammar is called a *derivation*.
- All strings of terminals generated in this way constitute the language specified by the grammar
- We write $L(G)$ for the language generated by the grammar G . Thus, $L(G_1) = \{0^n 1^n \mid n \geq 0\}$.

Language Specification

A grammar is used for a language specification by generating each string of the language in following manner:

1. Let x be the start variable; it is the *lhs* of one of the specification rules, the top rule, unless specified otherwise
2. Find a substring of x which is identical to the lhs of a rule. Replace the substring with the *rhs* of that rule in x .
3. Repeat step 2 until no variables remain in x thus generated

Formal Definition of a Grammar

A grammar is a 4-tuple $G = (V, \Sigma, R, S)$ where:

1. V is a finite set of symbols called the *variables* or *nonterminals*
2. Σ is an alphabet, disjoint from V , called *constants* or *terminals*
3. R is a finite set of *rules* (or specification rules) of the form $lhs \rightarrow rhs$, where $lhs \in (V \cup \Sigma)^+$, lhs contains at least one symbol in V , and $rhs \in (V \cup \Sigma)^*$
4. $S \in V$ is the start variable.

More on Specification Rules

- The *lhs* of a specification rule is a nonempty string of variables and terminals, containing at least one variable.
- The *rhs* of a specification rule consists of a string of variables and terminals

Derivation

- If $u, v, w \in (V \cup \Sigma)^*$ (i.e., are strings of variables and terminals) and $\alpha \rightarrow \beta \in R$ (i.e., is a rule of the grammar) then we say that $u\alpha v$ yields $u\beta v$, written $u\alpha v \Rightarrow u\beta v$
- We may also say that $u\beta v$ is *directly derived* from $u\alpha v$ using the rule $\alpha \rightarrow \beta$
- We write $u \xRightarrow{k} v$ if there exists a sequence $u_0, u_1, u_2, \dots, u_k \in (V \cup \Sigma)^*$, for $k > 0$, and $u = u_0 \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \dots \Rightarrow u_k = v$
- We also say that $u_0, u_1, u_2, \dots, u_k$ is a derivation of v from u
- We write $u \xRightarrow{*} v$ if $u = v$ or $u \xRightarrow{k} v$ for some $k > 0$.

Example of a Grammar

$G = (\{S, A\}, \{0, 1\}, P, S)$ where P :

$$S \rightarrow A$$

$$A \rightarrow 0A1$$

$$A \rightarrow \epsilon$$

Types of $\alpha \rightarrow \beta$

| α | β | Name |
|--|--|-------------------|
| $\alpha \in V$ | $\beta = \epsilon$ | ϵ -rule |
| $\alpha \in V$ | $\beta \in V$ | unit |
| | $\beta \in \Sigma^*$ | terminating |
| $\alpha \in V$ | $\beta \in (V + \epsilon)\Sigma^*$ | left-linear |
| $\alpha \in V$ | $\beta \in \Sigma^*(V + \epsilon)$ | right-linear |
| $\alpha \in V$ | $\beta \in \Sigma^*(V + \epsilon)\Sigma^*$ | linear |
| $\alpha \in V$ | | context-free |
| $\alpha = uAv, \beta = uwv,$ $u, v \in (V + \Sigma)^*, A \in V, w \in (V + \Sigma)^+$ | | context-sensitive |
| $ \alpha \leq \beta $ | | nondecreasing |
| | | phrase structure |

Language Specified by G

If $G = (V, \Sigma, R, S)$ is a grammar then the language specified by G (or the language of G) is

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

More notations

- To distinguish nonterminal from terminal strings we often enclose nonterminals in angular parentheses, $\langle \cdot \rangle$, and terminals in quotes " \cdot ".
- If two or more rules have the same *lhs*, as in the example $A \rightarrow 0A1$ and $A \rightarrow \epsilon$, we may compact them using the form
 $lhs \rightarrow rhs_1 \mid rhs_2 \mid \dots \mid rhs_n$
 where \mid is used with the meaning of an "or".
- E.g., the rules $A \rightarrow 0A1$ and $A \rightarrow B$ may be written
 $A \rightarrow 0A1 \mid \epsilon$.

Types of Grammars

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

| $\forall p \in P$ | $G \text{ \& } L(G)$ | Type |
|--------------------------|----------------------|------|
| p is left-linear | left-linear | 3 |
| p is right-linear | right-linear | 3 |
| p is linear | linear | 2.5 |
| p is context-free | context-free | 2 |
| p is context-sensitive | context-sensitive | 1 |
| p is nondecreasing | nondecreasing | 1 |
| p is phrase structure | phrase structure | 0 |

Note

- The CFG G_2 has ten variables (capitalized and in angular brackets) and 9 terminals (written in the standard English alphabet) plus a space character
- Also, the CFG G_2 has 18 rules
- Examples strings that belongs to $L(G_2)$ are:

a boy sees

the boy sees a flower

a girl with a flower likes the boy

CFG G_2

The CFG G_2 specifies a fragment of English

| | | |
|--|---------------|---|
| $\langle \text{SENTENCE} \rangle$ | \rightarrow | $\langle \text{NOUN - PHRASE} \rangle \langle \text{VERB - PHRASE} \rangle$ |
| $\langle \text{NOUN - PHRASE} \rangle$ | \rightarrow | $\langle \text{CP - NOUN} \rangle \mid \langle \text{CP - NOUN} \rangle \langle \text{PREP - PHRASE} \rangle$ |
| $\langle \text{VERB - PHRASE} \rangle$ | \rightarrow | $\langle \text{CP - VERB} \rangle \mid \langle \text{CP - VERB} \rangle \langle \text{PREP - PHRASE} \rangle$ |
| $\langle \text{PREP - PHRASE} \rangle$ | \rightarrow | $\langle \text{PREP} \rangle \langle \text{CP - NOUN} \rangle$ |
| $\langle \text{CP - NOUN} \rangle$ | \rightarrow | $\langle \text{ARTICLE} \rangle \langle \text{NOUN} \rangle$ |
| $\langle \text{CP - VERB} \rangle$ | \rightarrow | $\langle \text{VERB} \rangle \mid \langle \text{VERB} \rangle \langle \text{NOUN - PHRASE} \rangle$ |
| $\langle \text{ARTICLE} \rangle$ | \rightarrow | $a \mid the$ |
| $\langle \text{NPUN} \rangle$ | \rightarrow | $boy \mid girl \mid flower$ |
| $\langle \text{VERB} \rangle$ | \rightarrow | $touches \mid likes \mid sees$ |
| $\langle \text{PREP} \rangle$ | \rightarrow | $with$ |

Linear grammar rules

Let $G = (V, \Sigma, P, S)$ be a CFG and $r \in P$.

- r is linear if $rhs(r) \in \Sigma^* \circ V \circ \Sigma^*$
- r is *right-linear* if $rhs(r) \in \Sigma^* \circ V$
- r is *left-linear* if $rhs(r) \in V \circ \Sigma^*$
- r is *terminating* if $rhs(r) \in \Sigma^*$

Example Derivation with G_2

$\langle SENTENCE \rangle \Rightarrow \langle NOUN - PHRASE \rangle \langle VERB - PHRASE \rangle$
 $\Rightarrow \langle CP - NOUN \rangle \langle VERB - PHRASE \rangle$
 $\Rightarrow \langle ARTICLE \rangle \langle NOUN \rangle \langle VERB - PHRASE \rangle$
 $\Rightarrow a \langle NOUN \rangle \langle VERB - PHRASE \rangle$
 $\Rightarrow a \text{ boy } \langle VERB - PHRASE \rangle$
 $\Rightarrow a \text{ boy } \langle CP - VERB \rangle$
 $\Rightarrow a \text{ boy } \langle VERB \rangle$
 $\Rightarrow a \text{ boy sees}$

Theorem 2.19

If $L \subseteq \Sigma^*$ is a regular language then there is a right linear grammar G with $L(G) = L$

Proof: by construction. Since L is regular let DFA $D = (Q, \Sigma, \delta, q_0, F)$ with $L(D) = L$ and $\Sigma \cap Q = \emptyset$. Construct $G = (Q, \Sigma, P, q_0)$ where:

$P = \{q \rightarrow tq' \mid t \in \Sigma \wedge \delta(q, t) = q'\} \cup \{q \rightarrow \epsilon \mid q \in F\}$

1. G is right-linear and directly simulates D
2. if $x = x_1x_2 \dots x_n \in L(D)$ with $\delta(q_0, x_1) = q_1, \delta(q_1, x_2) = q_2, \dots, \delta(q_{n-1}, x_n) = q_n$ then $q_0 \Rightarrow x_1q_1 \Rightarrow x_1x_2q_2 \Rightarrow \dots \Rightarrow x_1x_2 \dots x_nq_n$
3. Since $q_n \in F$ and $q_n \rightarrow \epsilon \in P, x_1x_2 \dots x_n \in L(D)$
4. Hence, $L(D) = L(G)$

Example of Linear Grammar

$G = (\{A, B\}, \{0, 1\}, \{A \rightarrow 0A \mid B, B \rightarrow 1B \mid \epsilon\}, A)$
is a right-linear grammar

$A \Rightarrow 0A \Rightarrow 00A \Rightarrow 00B \Rightarrow 001B \Rightarrow 0011B \Rightarrow 00111B \Rightarrow 00111$
is a derivation in G

$L(G) = 0^*1^*$

Example

Consider $G = (\{A, B\}, \{a, b\}, \{A \rightarrow abA|abB, B \rightarrow aaaB|b\}, A)$. G' stated in Lemma 2.20 is $G' = (\{A, X_1, Y_1, B, Z_1, Z_2\}, \{a, b\}, P', A)$ where P' is:

$A \rightarrow aX_1|aY_1, X_1 \rightarrow bA, Y_1 \rightarrow bB,$

$B \rightarrow aZ_1|b, Z_1 \rightarrow aZ_2, Z_2 \rightarrow aB$

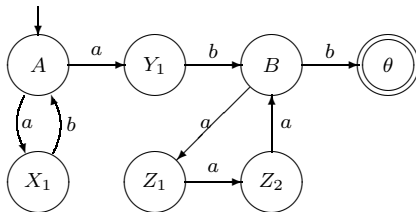
Lemma 2.20

For each right-linear grammar $G = (N, \Sigma, P, S)$ there is a right-linear grammar $G' = (V', \Sigma, P', S)$ where each non-terminating rule $A \rightarrow \alpha \in P'$ has $|\alpha| \leq 2$ and each terminating rule $A \rightarrow x \in P'$ has $|x| \leq 1$.

Proof idea: Note that G does not forbid erasing ϵ or unit rules (ϵ may belong to a regular language). For $A \rightarrow \alpha \in P$ with $|\alpha| > 2$, to transform it equivalently we can use construction from Fleck's book...

Example

The NFA M accepting the language specified by G' constructed in the above example



Theorem 2.21

For each right-linear grammar G , $L(G)$ is regular

Proof: by construction

- For $G = (V, \Sigma, P, S)$, using Lemma 2.20 we may assume that P has only productions of the form $X \rightarrow xY$ and $X \rightarrow z$, $X, Y \in V$, $z \in \Sigma \cup \{\epsilon\}$.
- An NFA M that accepts $L(G)$ is $M = (V \cup \{\theta\}, \Sigma, \delta, S, \{\theta\})$ where for each $X \in V$ and $z \in \Sigma \cup \{\epsilon\}$:
 - if $X \rightarrow z \in P$ then $\delta(X, z) = \{\theta\} \cup \{Y \mid X \rightarrow zY \in P\}$
 - if $X \rightarrow z \notin P$ then $\delta(X, z) = \{Y \mid X \rightarrow zY \in P\}$
- It can be directly checked that $L(M) = L(G)$

Relationship

- The results produced here for regular grammars show that any regular language, RL, is a context-free language, CFL.
- However, as we know, the language $L = \{0^n 1^n \mid n \geq 0\}$ is not regular
- A CFG that generates $L = \{0^n 1^n \mid n \geq 0\}$ is:
 $G = (\{S\}, \{0, 1\}, \{S \rightarrow 0S1 \mid \epsilon\})$ which shows that not every context free language is regular
- Hence, $RL \subset CFL$

Note

- There is a perfect duality between recognition and generation of regular languages: when an acceptor makes an atomic step it consumes one input symbol; when a grammar rule is used in an atomic step it produces one symbol
- The results established for right-linear grammars hold also for left-linear grammars:
Theorem: A language is right-linear iff it's left-linear.
- The term *regular grammar* is further used to refer to a grammar which is right-linear or left-linear.