Representing Languages

Discrete Mathematics
Evgeny Skvortsov

Previous Lecture

- Alphabets, strings, length of strings, concatenation and power
- Formal languages
- Constructing languages: Set theoretical operations
- Concatenation, property of concatenation

Kleene Star

- For a language A over an alphabet Σ:

$$A^0 = \{\lambda\},\ A^1 = A,\quad \text{and for } n \in \mathbb{N},\ A^n = A^{n-1}A = \underbrace{AA\ldots A}_{n \text{ times}}$$

$$A^+ = A \cup A^2 \cup A^3 \cup \ldots = \bigcup_{n=1}^{\infty} A^n \quad \text{the positive closure of } A$$

$$A^* = \{\lambda\} \cup A^+ = \bigcup_{n=0}^{\infty} A^n \quad \text{the Kleene closure or Kleene star}$$

- Let $A = \{aa,ab,ba,bb\}$. Then A^* is the language of all strings of even length.
- If $B = \{a,b\}$ then BA^* is the language of all strings of odd length
- What are $\{a\}\{ba\}^*$ and $\{ab\}^*\{b\}$?

Properties of Kleene Star

Lemma

Let Σ be an alphabet and A, B languages over Σ. If $A \subseteq B$ then for all $n \in \mathbb{N}$, $A^n \subseteq B^n$

Proof

We prove by induction on n. Since $A^1 = A \subseteq B = B^1$, the result is true for n = 1, that gives us the basis step of induction.

Suppose that the lemma is true for n = k. We have: if $A \subseteq B$ then $A^k \subseteq B^k$. Consider a string x from A^{k+1}. This string can be represented as follows x = uv, where $u \in A$ and $v \in A^k$.

If $A \subseteq B$, then $u \in B$, and also by inductive hypothesis $v \in B^k$.

Therefore $x = uv \in BB^k = B^{k+1}$, and $A^{k+1} \subseteq B^{k+1}$ Q.E.D

Properties of Kleene Star (cntd)

Theorem

For an alphabet Σ and languages A, B over Σ

(1) $A \subseteq AB^*$

(2) $A \subseteq B$ implies $A^+ \subseteq B^+$

(3) $AA^* = A^*A = A^+$

(4) $(A \cup B)^* = (A^* \cup B^*)^* = (A^*B^*)^*$

(5) $A \subseteq B^*A$

(6) $A \subseteq B$ implies $A^* \subseteq B^*$

(7) $A^*A^* = A^* = (A^*)^* = (A^*)^+ = (A^+)^*$

Proof

(2) Let $A \subseteq B$ and $u \in A^+$. Then $u \in A^n$ for some $n \in N$. By Lemma, it follows that $u \in B^n \subseteq B^+$, and so $A^+ \subseteq B^+$

Q. E. D.

Regular Expressions

- An atomic language is a language that contains only one string, and this string has length 1. {a}

  For short we denote such a language simply by a
- Every language that contains only one string can be represented as a concatenation of atomic languages. A = {abba} = abba

  (Careful!!! The abba in the parenthesis is a string, while the abba in the end is a concatenation of languages.)
- Any finite language is a union of concatenations of atomic languages. A = {ab,ba,abba} = ab ∪ ba ∪ abba
- An expression constructed from atomic languages by means of concatenation, union, intersection, complementation, and Kleene star is called a regular expression

Regular Expressions: Examples

- What the languages a^*ba^*b, $(a \cup b)^* c^*$, $((ab \cup ba)^* c)^*$ are?

- Write a regular expression for the language over {a,b,c} that contains strings with exactly one occurrence of c
- with exactly two occurrences of c
- over {a,b,c,d} with one occurrence of c and one occurrence of d
- with as many occurrences of c as you wish, but each such occurrence should be followed by an occurrence of d

Grammars

- What is a sentence in a natural language?
- One typical rule is: A sentence can consist of a noun phrase followed by a predicate.

  We may represent this rule as

  <sentence> → <noun_phrase> <predicate>

- This is not enough to deal with real sentences, and we need to explain <noun_phrase> and <predicate>

  <noun_phrase> → <article> <noun>

  <predicate> → <verb>

Grammars (cntd)

- This is still not enough as we need concrete words in a sentence, so we continue

  <article> → the | a

  <noun> → dog | boy | …

  <verb> → walks | runs | …

- If we describe all possible constructions, we get a full description of possible sentences

Grammars (example)

- Programming languages are described in terms of grammars

<conditional statement> ::= <if statement> | <if statement> **else** <statement> | <if clause> <for statement> | <label>: <conditional statement>

<if clause> ::= **if** <Boolean expression> **then**

<unconditional statement> ::= <basic statement> | <compound statement> | <block>

<if statement> ::= <if clause> <unconditional statement>

<Boolean expression> ::= <simple Boolean> | <if clause> <simple Boolean> **else** <Boolean expression>

<simple Boolean> ::= <implication> | <simple Boolean>

…

Definition

- A grammar G consists of
  - V set of variables
  - Σ set of terminals (or terminal symbols)
  - $S \in V$ a start symbol
  - P a set of rules or productions
- Every production has the form

  $x \rightarrow y$

  where x is a non-empty string consisting of terminals and variables, and y is any string consisting of terminals and variables

  <noun_phrase> runs → <article> dog runs

  <if clause> ::= if <Boolean expression> then

Derivation

- If we have a rule $x \rightarrow y$, and a string of the form $w = uxv$, then we can use the rule to derive the string $z = uyv$

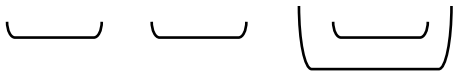
  $w = uxv \Rightarrow uyv = z$

  w derives z
- Rule $S \rightarrow aSb$

  $abSbaSbaSaba \Rightarrow abSbaSbaaSbaba$

  
- If $w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_n$ then we write $w_1 \Rightarrow^* w_n$
- Example

  $abSbaSbaSaba \Rightarrow^* abaSbbaaSbbaaaSbbaba$

Generating a Language

- Let $G = (V,\Sigma,S,P)$ be a grammar. Then the language $L(G)$ generated by G is the set of all strings of terminal symbols that can be derived from S

  Formally,

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

- What is the language generated by the following grammar?

  $S \rightarrow Aa$ $\quad A \rightarrow \lambda$

  $A \rightarrow Ba$ $\quad B \rightarrow \lambda$

  $A \rightarrow Bb$ $\quad B \rightarrow abba$

More Examples

- What is the language generated by the following grammar?

  $S \rightarrow aSb \qquad S \rightarrow \lambda$

- What is the language generated by the following grammar?

  $S \rightarrow SS \qquad S \rightarrow \lambda \qquad S \rightarrow aSb \qquad S \rightarrow bSa$

- Suggest a grammar that generates the language of properly placed parenthesis.

Homework

Exercises from the Book:

No. 17, 19 (page 318)

Suggest a grammar that generates correct algebraic expressions