COSE215: Theory of Computation

Lecture 5 — Regular Expressions

Hakjoo Oh
2018 Spring

Motivation: Searching for Patterns

```
theoretical
computer
science
formal
language
patterns
regular
expression
sequence
```

- Find all words that contain at least one consecutive `t`'s:

  `$ cat textfile | grep "t\+"`

- Find all words that contain at least two e's:

  `$ cat textfile | grep "e[a-z]*e"`

Regular expression

A regular expression denotes a language.
E.g., $(a + (b \cdot c))^*$ stands for:

$$\{\epsilon, a, bc, aa, abc, bca, bcbc, aaa, aabc, \ldots\}$$

Syntax

Definition (Syntax of regular expressions)

Regular expressions over alphabet Σ are constructed recursively:

1. (Basis) $\emptyset$, ϵ, and $a \in \Sigma$ are regular expressions.
2. (Induction)
   - If R_1 and R_2 are regular expressions, so are $R_1 + R_2$ and $R_1 \cdot R_2$.
   - If R is a regular expression, so are R^* and (R).

$$
\begin{array}{rcl}
R & \rightarrow & \emptyset \\
& \mid & \epsilon \\
& \mid & a \in \Sigma \\
& \mid & R_1 + R_2 \\
& \mid & R_1 \cdot R_2 \\
& \mid & R^* \\
& \mid & (R)
\end{array}
$$

Semantics

Definition (Semantics of regular expressions)

A regular expression $\boldsymbol{R}$ means a set of strings, denoted $\boldsymbol{L(R)}$, which is defined inductively:

$$
\begin{array}{rcl}
L(\emptyset) & = & \emptyset \\
L(\epsilon) & = & \{\epsilon\} \\
L(a) & = & \{a\} \\
L(R_1 + R_2) & = & L(R_1) \cup L(R_2) \\
L(R_1 \cdot R_2) & = & L(R_1) L(R_2) \\
L(R^*) & = & (L(R))^* \\
L((R)) & = & L(R)
\end{array}
$$

Example

$$L(a^* \cdot (a + b)) =$$

Exercises

Find the languages of the regular expressions and equivalent finite automata.

- $(a + b)^*$
- $(a + b)^*(a + b)$
- $(a \cdot a)^*(b \cdot b)^*b$

Exercises

Find regular expressions for the languages:

- $L = \{w \in \{0, 1\}^* \mid$ 0 and 1 alternate in $w\}$
- $L = \{w \in \{0, 1\}^* \mid w$ has at least one pair of consecutive zeros$\}$
- $L = \{a^n b^m \mid n \geq 3, m$ is even$\}$
- $L = \{a^n b^m \mid (n + m)$ is even$\}$
- $L = \{a^n b^m \mid n \geq 4, m \leq 3\}$

cf) Automatic Synthesis of Regular Expressions

- Regular expressions are useful for specifying string patterns, but constructing a regular expression is nontrivial and difficult for end-users.
- Ex) Find a regular expression for the language:

$$L = \{w \in \{0, 1\}^* \mid w \text{ has exactly one pair of consecutive 0s}\}$$

  - Positive examples: 00, 1001, 010010, 1011001110, ...
  - Negative examples: 01, 11, 000, 00100, ...
- Automatic synthesis of regular expressions from examples!

Regular Expression Synthesizer

Positive Examples

00,
1001,
010010,
1011001110

Negative Examples

01,
11,
000,
00100

$\Longrightarrow$ $\boxed{\text{RE Synthesizer}}$ $\Longrightarrow$ $\mathbf{(0?1)^*00(10?)^*}$

Summary

- Syntax and semantics of regular expressions.
- Automatic synthesis of regular expressions. Read the paper:
  - Mina Lee, Sunbeom So, and Hakjoo Oh.
    Synthesizing Regular Expressions from Examples for Introductory Automata Assignments. GPCE 2016.