Lecture #4: Regular Operations and Regular Expressions
Key Concepts

Regular Operations

This lecture introduced the ***regular operations*** (union, concatenation, and Kleene star) on languages. Let Σ be an alphabet and let $A, B \subseteq \Sigma^\star$, so that A and B are *languages* with alphabet Σ.

- The ***union*** of the languages A and B is the language

$$A \cup B = \{\omega \in \Sigma^\star \mid \omega \in A \text{ or } \omega \in B \text{ (or both)}\}.$$

- The ***concatenation*** of the languages A and B is the language

$$A \circ B = \{\omega_1 \cdot \omega_2 \mid \omega_1 \in A \text{ and } \omega_2 \in B\}.$$

- The ***Kleene star*** of the language A is the language

$$A^\star = \{\omega_1 \cdot \omega_2 \ldots \omega_k \mid k \geq 0 \text{ and } \omega_i \in A \text{ for } 1 \leq i \leq k\}$$

  This language is also, sometimes called the ***Kleene closure*** of A — or the ***star*** of A.

Closure Properties

A ***closure property*** for a set S of languages over an alphabet Σ, is a property stating — for an *operation* on languages over Σ — that if the operation is applied to languages that all belong to the set S, then the result is a language that belongs to the set S, as well. The following result describes several closure properties for the set of regular languages.

Theorem 1. *Let Σ be an alphabet, and let $A, B \subseteq \Sigma^\star$.*

(a) If A and B are regular languages then $A \cup B$ is a regular language, as well.

(b) If A and B are regular languages, then $A \circ B$ is a regular language, as well.

(c) If A is a regular language then $A^\star$ is a regular language as well.

Regular Expressions and Their Languages

Definition

Let Σ be an alphabet that *does not* include any of the symbols

$$\lambda, \emptyset, \Sigma, (,), \cup, \circ, ^\star$$

and let

$$\Sigma_{\text{regexp}} = \Sigma \cup \{\lambda, \emptyset, \text{“}\Sigma\text{”}, (,), \cup, \circ, ^\star\}$$

so that Σ_{regexp} incudes a copy of the *symbol*, “Σ”, that we are also using as the name of the language we are starting with.

A ***regular expression over the alphabet Σ*** is a kind of string of symbols, in $\Sigma^\star_{\text{regexp}}$, as defined by the following list of seven rules — and he ***language of a regular expression, over the alphabet Σ***, is a subset of $\Sigma^\star$ that is defined as follows, as well.

1. For every ***symbol*** $\sigma \in \Sigma$, the ***string*** σ, with length one in $\Sigma^\star_{\text{regexp}}$ is a regular expression over Σ. The ***language***, $L(\sigma)$, of the regular expression σ, is the ***set*** $\{\sigma\}$.

2. The ***string*** λ, with length one in $\Sigma^\star_{\text{regexp}}$, is a regular expression over Σ. The ***language***, $L(\lambda)$, of the regular expression λ, is the ***set*** $\{\lambda\}$.

3. The ***string*** $\emptyset$, with length one in $\Sigma^\star_{\text{regexp}}$ is a regular expression over Σ. The ***language***, $L(\emptyset)$, of the regular expression $\emptyset$, is the ***set*** $\emptyset$.

4. The ***string*** Σ, with length one in $\Sigma^\star_{\text{regexp}}$, is a regular expression over (the alphabet) Σ. The ***language***, $L(\Sigma)$, of the regular expression Σ, is the ***finite set*** Σ.

5. If $R_1 \in \Sigma^\star_{\text{regexp}}$ and $R_2 \in \Sigma^\star_{\text{regexp}}$ are ***regular expressions over Σ*** then the ***string***

$$(\, R_1 \,\cup\, R_2\,) \tag{1}$$

   (of symbols in Σ_{regexp}) is a regular expression over Σ. The ***language*** $L(R)$, of the regular expression R at line (1), is the ***set***

$$L(R_1) \cup L(R_2).$$

6. If $R_1 \in \Sigma^\star_{\text{regexp}}$ and $R_2 \in \Sigma^\star_{\text{regexp}}$ are ***regular expressions over*** $\boldsymbol{\Sigma}$ then the ***string***

$$(R_1 \circ R_2) \tag{2}$$

(of symbols in Σ_{regexp}) is a regular expression over Σ. The ***language*** $L(R)$, of the regular expression R at line (2), is the ***set***

$$L(R_1) \circ L(R_2).$$

7. If $R_1 \in \Sigma^\star_{\text{regexp}}$ is a ***regular expression over*** $\boldsymbol{\Sigma}$ then the ***string***

$$(R_1)^\star \tag{3}$$

(of symbols in Σ_{regexp}) is a regular expression over Σ. The ***language*** $L(R)$, of the regular expression R at line (3), is the ***set***

$$L(R_1)^\star$$

Equivalence

Theorem 2. *Let Σ be an alphabet that does not include any of the symbols "λ", "$\emptyset$", "Σ", "$($", "$)$", "$\cup$", "$\circ$", or "*", and let $L \subseteq \Sigma^\star$.*

Then L is a **regular language** *if and only if there exists a regular expression R, over the alphabet Σ, such that $L = L(R)$.*

Parsing

Definition: A ***parse tree*** for a regular expression ω over Σ is a rooted tree that can be defined as follows, for a given regular expression $\omega \in \Sigma^\star$:

(a) If ω is a symbol $\sigma \in \Sigma$ then the parse tree for ω has a single node, with label σ.

(b) If ω is the symbol "λ", then the parse tree for ω has a single node, with label "λ".

(c) If ω is the symbol "$\emptyset$", then the parse tree for ω has a single node, with label "$\emptyset$".

(d) If ω is the symbol "Σ", then the parse tree for ω has a single node, with label "Σ".

(e) If ω is a regular expression "$(\mu \cup \nu)$", for regular expressions μ and ν over Σ, then the parse tree for ω has a root with label '$\cup$" with two children. The first child is the root of a parse tree for μ, and the second child is the root of a parse tree for ν.

(f) If ω is a regular expression "$(\mu \circ \nu)$, for regular expressions μ and ν over Σ, then the parse tree for ω has a root with label "$\circ$" with two children. The first child is the root of a parse tree for μ, and the second child is the root of a parse tree for ν.

(g) If ω is a regular expression "$(\mu)^\star$, for a regular expression μ over Σ, then the parse tree for ω has a root with label "$\star$", with one child. The child is the root of a parse tree for μ.

An efficient algorithm for the computation of a parse tree, from a given regular expression over an alphabet Σ, is described in the lecture material.

Applications

Three ***applications*** of parse trees — involving useful computational problems concerning regular expressions — were described in the lecture material.

- The ***Recognition*** Problem: Given a string $R \in \Sigma^\star_{\text{regexp}}$, decide whether R is a regular expression over Σ.

  This problem can be solved, efficiently, by applying the above "parsing" algorithm, to try to construct a parse tree for R: R is a regular expression over Σ if and only if this attempt is successful.

- ***Conversion to NFA:*** Using a regular expression R, over an alphabet Σ, to produce a ***nondeterministic finite automaton*** with the same alphabet, Σ, and with the same language as R.

  A parse tree for the given regular expression can be used with a recursive algorithm to generate the desired nondeterministic finite automaton (using material from the constructive proof of the closure of the set of regular languages, under the regular operations, that is included in the material for this lecture.

- The ***Membership*** Problem: Given a regular expression R over an alphabet Σ, and a string $\omega \in \Sigma^\star$, decide whether ω is in the language of R.

  This problem can be solved by constructing a nondeterministic finite automaton M (with alphabet Σ) with the same language as R, and then deciding whether $\omega \in L(M)$, using the process for this that was described for this in the lecture material on "Nondeterministic Finite Automata".

Regular Expressions in Practice

Regular operations used in software operations are not, quite, the same as the regular expressions defined in this course. Changes include the following:

- Changes needed to define "regular expressions" for alphabets Σ that include some, or all, of the special symbols "λ", "$\emptyset$", "(", ")", "$\cup$", "$\circ$", and "*".

- *Shortcuts* allowing “simpler” regular expression that do not include unnecessary brackets and support common cases — resulting in regular expressions that are shorter, but can also be harder to parse.

Industry standards for regular expressions, and software packages supporting their use, have been developed and used. Information concerning “regular expressions in practice” — which is ***for interest only*** in this course — is available in a supplemental document about this.