Computer Science 351

More About Regular Expressions

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #8

Goal for Today

Goals for Today:

- Processes to solve the following problems:
  - ***Recognition:*** Given a string $R \in \Sigma^{\star}_{\text{regexp}}$, decide whether R is a regular expression over Σ.
  - ***Membership:*** Given a regular expression R over Σ, and a string $\omega \in \Sigma^{\star}$, decide whether ω belongs to the language, $L(R)$, of R.
- Introduce an important data structure — a ***parse tree*** — along the way.
- Provide more information about how regular expressions are used in practice.

Regular Expressions

- 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.

- Recall that 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.
- Recall, as well, that the ***language of a regular expression, over the alphabet Σ***, is a subset of $\Sigma^\star$ that is defined as follows.

Regular Expressions

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*** $\{\lambda\}$.

Regular Expressions

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.

Regular Expressions

5. 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***

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

(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***

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

Regular Expressions

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***

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

Regular Expressions

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***

$$L(R_1)^{\star}$$

Parsing — Useful Results

Let Σ and Σ_{regexp} be as above. The following results concern a string

$$\omega = \sigma_1 \sigma_2 \ldots \sigma_n \tag{4}$$

where $\sigma_1, \sigma_2, \ldots, \sigma_n \in \Sigma_{\text{regexp}}$ (so that $\omega \in \Sigma^\star_{\text{regexp}}$) and $n \geq 2$.

Parsing — Useful Results

Lemma 1. Let ω be as shown at line (4), and let k be an integer such that $1 \leq k \leq n-1$. Suppose that ω is a regular expression over Σ.

(a) If σ_n is not the symbol "$\star$", then the number of copies of "(" in the prefix $\sigma_1\sigma_2\ldots\sigma_k$ is strictly greater than the number of copies of ")" in this prefix.

(b) If σ_n is not the symbol "$\star$", then the number of copies of "(" in ω is equal to the number of copies of ")" in ω.

(c) If σ_n is the symbol "$\star$" and $k \leq n-2$ then the number of copies of "(" in the prefix $\sigma_1\sigma_2\ldots\sigma_k$ is strictly greater than the number of copies of ")" in this prefix.

(d) If σ_n is the symbol "$\star$" then the number of copies of "(" in the prefix $\sigma_1\sigma_2\ldots\sigma_{n-1}$ is equal to the number of copies of ")" in this prefix.

Parsing — Useful Results

Lemma 2. Let ω be a regular expression over Σ that is as shown at line (4), and suppose that $\sigma_n \neq$ "$\star$". Then there exists a unique integer k, such that $2 \leq k \leq n-1$, which satisfies the following properties.

(a) The number of copies of "(" in the string $\nu = \sigma_2\sigma_3 \ldots \sigma_{k-1}$ is equal to the number of copies of ")" in the above string ν.

(b) Either $\sigma_k =$ "$\cup$" or $\sigma_k =$ "$\circ$".

Furthermore, if k is the (unique) integer described above then both of the substrings

$$\nu = \sigma_2\sigma_3 \ldots \sigma_{k-1} \quad \text{and} \quad \varphi = \sigma_{k+1}\sigma_{k+2} \ldots \sigma_{n-1}$$

of ω are regular expressions over Σ, so that $\omega = (\nu \cup \varphi)$ if $\sigma_k =$ "$\cup$", and $\omega = (\nu \circ \varphi)$ if $\sigma_k =$ "$\circ$".

Parse Trees

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".

Parse Trees

(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.

Parse Trees

- A ***recursive algorithm*** can be used — along with Lemma #2 and the above definition of a parse tree — to either construct a parse tree corresponding to a given string $R \in \Sigma^{\star}_{\text{regexp}}$, or confirm that R is not a regular expression over Σ at all.
- A supplemental document on "Parsing Regular Expressions" includes this algorithm, along with more information about its correctness and efficiency.

Application — Identifying Regular Expressions

- Consider 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.

Application — Construction of a Nondeterministic Finite Automaton

- Consider the problem of using a given regular expression R over Σ to to produce a ***nondeterministic finite automaton*** with the same language as R.
- An algorithm for this computation can begin by constructing a ***parse tree*** for R.
- This can be used, along with the constructive proof of the ***closure*** of the set of regular languages — in the lecture material for “Regular Operations and Regular Expressions” — to complete an algorithm to construct a nondeterministic finite automaton (with alphabet Σ) with the same language as R.

Application — Deciding Membership in the Language of a Regular Expression

- Finally, consider the ***Membership*** problem: Given a regular expression R over an alphabet Σ, and a string $\omega \in \Sigma^\star$, decide whether $\omega \in L(R)$.
- In order to solve this problem, one can start by construction a nondeterministic finite automaton, M, with the same language as R, by applying the algorithm given above.
- The process of deciding membership in the language of a nondeterministic finite automaton, discussed in the lecture material about an “Introduction to Nondeterministic Finite Automata”, can then be used to decide whether $\omega \in L(M)$ — which is true if and only if $\omega \in L(R)$.

Applications — Further Information

- The supplemental document on “Parsing Regular Expressions” includes additional details about the applications and algorithms that have been described above.

Regular Expressions in Practice

- Regular expressions are used in a variety of software applications — including text editing, word processing, and data analysis.
- However, the regular expressions that are practice are somewhat different from the regular expressions that have been described in this course.
- Changes include:
  - Changes needed to define “regular expressions” for alphabets Σ that include some, or all, of the special symbols “λ”, “$\emptyset$”, “(”, “)”, “$\cup$”, “$\circ$”, and “$\star$”.
  - *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.

Regular Expressions in Practice

- 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.