3 Regular Expressions, Regular Languages

[Linz, Ch. 3]

Regular Expressions over Σ

Let $\mathbf{\Sigma}$ be a finite alphabet.

A **regular expression** over $\mathbf{\Sigma}$ (or Σ-regular expression) is a string built up from the Σ-symbols, λ and $\varnothing$,
and the operators $+$, $\cdot$, $*$ and (,).

RegExp$(\mathbf{\Sigma})$ is the set of $\mathbf{\Sigma}$**-regular expressions**.

Recursive Definition of RegExp(Σ)

[Cf. p. 1-14 for recursive definition of Σ^*]

Basis:

- $a \in \Sigma \Longrightarrow a \in RegExp(\Sigma)$
- $\lambda \in RegExp(\Sigma)$
- $\emptyset \in RegExp(\Sigma)$

Recursive Steps:

- $r_1, r_2 \in \textbf{\textit{RegExp}}(\Sigma) \implies r_1 + r_2 \in \textbf{\textit{RegExp}}(\Sigma)$
- $r_1, r_2 \in \textbf{\textit{RegExp}}(\Sigma) \implies r_1 \cdot r_2 \in \textbf{\textit{RegExp}}(\Sigma)$
- $r \in \textbf{\textit{RegExp}}(\Sigma) \implies r^* \in \textbf{\textit{RegExp}}(\Sigma)$
- $r \in \textbf{\textit{RegExp}}(\Sigma) \implies (r) \in \textbf{\textit{RegExp}}(\Sigma)$

Alternatively, use **modified BNF** to define ***RegExp***(Σ):

Given Σ, with elements $a, \ldots,$

define ***RegExp***(Σ), with elements $r, r_1, r', \ldots$:

$$r ::= a \mid \lambda \mid \emptyset \mid (r_1 + r_2) \mid (r_1 \cdot r_2) \mid r^* \mid (r)$$

Note:

(1) Can **drop** "$\cdot$" for concatentation

(2) Can **drop parentheses**

- Use **Rules of Precedence**

$*$

$\cdot$

$+$

A Σ-regular expression r defines a Σ-language $L(r) \subseteq \Sigma^*$.

Definition: We define $L(r) \subseteq \boldsymbol{\Sigma^*}$ by **structural recursion** on $r \in \boldsymbol{RegExp}(\Sigma)$.

(Cf. definition of $\boldsymbol{RegExp}(\Sigma)$ by **structural recursion** on p. 3-1.)

Base cases:

$L(a) = \{a\}$

$L(\lambda) = \{\lambda\}$

$L(\varnothing) = \varnothing$

Recursive Steps:

$L(r_1 + r_2) = L(r_1) \cup L(r_2)$

$L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)$ $L(r^*) = L(r)^*$

$L((r)) = L(r)$

Note:
Alternatively, this definition is by (CV) recursion on $\boldsymbol{compl}(r)$ or $|r|$.

Exercises:

(a) Given a regular expression, describe what **language** it defines.

(b) Given a set $A \subseteq \Sigma$, find a regular expression, r, which defines A, i.e. s.t. $L(r) = A$.

Linz, section 3.1 has **many examples** of these.

Examples: [Assume $\Sigma = \{a, b\}$ unless otherwise stated.]

(1) Find r s.t. $L(r) = \{a^m b^n \mid m, n \geq 0, m \text{ even}, n \text{ odd}\}$

(2) Find r s.t. $L(r) = \Sigma^*$ if $\Sigma = \{a_1, ..., a_k\}$

(3) Find $r : L(r) =$ set of all Σ-strings with **no consec.** a's or b's.

Definition: **Equivalence of Regular Expressions**

$r_1 \equiv r_2 \Longleftrightarrow L(r_1) = L(r_2)$

Examples: Are the following pairs of regular expressions equivalent? (assume $\Sigma = \{a, b\}$)

(1) $r_1 + r_2 \stackrel{?}{\equiv} r_2 + r_1$

(2) $r_1 r_2 \stackrel{?}{\equiv} r_2 r_1$

(3) $(r^*)^* \stackrel{?}{\equiv} r^*$

(4) $r_1(r_2 + r_3) \stackrel{?}{\equiv} r_1 r_2 + r_1 r_3$

Examples: Try some of the following:

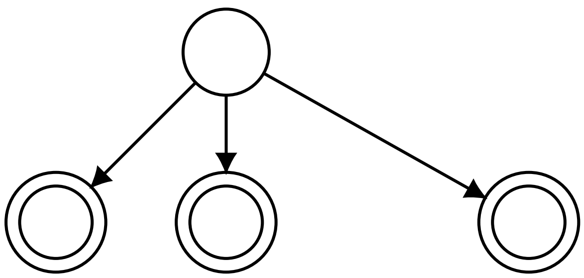
L6, §3.1 Exs p. 78-79:
Questions: 7, 8, 9.ab, 10, 15, 16, 18, 19.ab, 20.abc, 21.b

L5, §3.1 Exs p. 75-77:
Questions: 4, 5, 6.ab, 7, 11, 13, 15, 16.ab, 17.abc, 18.b

Note:

(1) We saw earlier (p. 2-16) that changing the definition of **nfa's** to allow more than one state would not make a significant difference, since any **nfa** with more than one start state can be replaced by an **equivalent nfa** with one start state.

(2) Similarly, we can assume without loss of generality (w.l.o.g. for future reference) that any **nfa** that we are using does _not_ have more than one final state. Why? Consider:



Assume, given an alphabet, Σ.

Definition: A Σ-language, $L \subseteq \Sigma^*$, is **regular** if it is defined by a **regular expression**, r. i.e. if

$$\boldsymbol{L = L(r)}$$

Note:

This is **not** Linz's definition of a regular language (Def. 2.3, p. 46 in L6), but it turns out to be equivalent.