CHAPTER 4: REGULAR EXPRESSIONS

Peter Cappello
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
cappello@cs.ucsb.edu

• The corresponding textbook chapter should be read before attending this lecture.

• These notes are not intended to be complete. They are supplemented with figures, and other material that arises during the lecture period in response to questions.

*Based on Theory of Computing, 2nd Ed., D. Cohen, John Wiley & Sons, Inc.

1
Defining Languages More Precisely

- We now focus on a more precise language to define a class of languages. This language, or meta-language, is called regular expressions.
- Regular expressions are a simple *declarative* programming language.
- Regular expressions are used in various places, including:
  - Search commands, such as UNIX grep, or what one finds in Web browsers.
  - Lexical analyzer generators, such as Lex.
  - These define the language’s *tokens* (e.g., keywords, identifiers, operators, numbers).
  - The lexical analyzer generator produces code that, when executed, returns the next token in its input stream.
**Regular Expression Operators**

We first remind ourselves what union, concatenation, and Kleene closure mean in the context of languages. Given languages $L$ and $M$:

- Their **union**, denoted $L \cup M$, is \{ $w$ | $w \in L$ or $w \in M$ \}.

- The **concatenation** of $L$ and $M$, denoted $LM$, is \{ $xy$ | $x \in L$, $y \in M$ \}.

- The **Kleene closure** of $L$, denoted $L^*$, is

$$
\bigcup_{i \geq 0} L^i = L^0 \cup L^1 \cup L^2 \cup \cdots,
$$

where $L^0 = \{ \Lambda \}$, $L^1 = L$, and $L^i = LL^{i-1}$.
If $L = \{0\}$ and $M = \{11\}$. Then:

- $L \cup M = \{0, 11\}$
- $LM = \{011\}$
- $ML = \{110\}$
- $L^* = \{\Lambda, 0, 00, \ldots\}$.

What is $\emptyset^*$?
What is $\{\Lambda\}^*$?
How many strings are in $(L \cup M)^3$?
A Recursive Definition Regular Expressions

Let $\Sigma$ be a set of symbols.

Basis rules

1. The constants $\epsilon$ and $\emptyset$ are regular expressions. $L(\epsilon) = \{\Lambda\}$; $L(\emptyset) = \emptyset$.
2. For each $a \in \Sigma$, $a$ is a regular expression. $L(a) = \{a\}$. 
Recursive rules

1. If $E$ and $F$ are regular expressions, $E + F$ is a regular expression. $L(E + F) = L(E) \cup L(F)$.

2. If $E$ and $F$ are regular expressions, $EF$ is a regular expression. $L(EF) = L(E)L(F)$.

3. If $E$ is a regular expression, $E^*$ is a regular expression. $L(E^*) = (L(E))^*$.

4. If $E$ is a regular expression, $(E)$ is a regular expression. $L((E)) = L(E)$.

- If $e$ is a regular expression, $L(e)$ is a regular language
- We refer to $L(e)$ as the language denoted by $e$. 
Example 1

Can we define the language whose words consist of alternating 0s and 1s using regular expressions?

- $01^*$ denotes the language whose strings begin with one 0, followed by zero or more 1s.
- $(01)^*$ denotes the language whose strings are the concatenation of zero or more occurrences of 01.
- $(1 + \epsilon)(01)^*$ denotes the set of strings consisting of alternating 0s and 1s that end in 1.
- $(1 + \epsilon)(01)^*(0 + \epsilon)$ denotes the desired language.
**Regular Expression Operator Precedence**

- Regular expression operator precedence: $* > \text{concatenation} > +$.
- Use parentheses to override these operator precedences.
- E.g., $01^* + 1 = (0(1)^*) + 1$.
- What language is denoted by the regular expression above?
- What language is denoted by $0(1^* + 1)$?
Example 2

Give a regular expression for the language $L$ over $\Sigma = \{a, b\}$ of words that contain exactly 2 or exactly 3 $b$’s.

- The set of all strings over $\Sigma$ that contain exactly 1 $b$ is denoted by the regular expression $a^*ba^*$.
- The set of all strings over $\Sigma$ that contain exactly 2 $b$’s is denoted by the regular expression $a^*ba^*ba^*$.
- The set of all strings over $\Sigma$ that contain exactly 3 $b$’s is denoted by the regular expression $a^*ba^*ba^*ba^*$.
- $L$ is denoted by the regular expression $a^*ba^*ba^* + a^*ba^*ba^*ba^*$.

We implement “or” with union, the $+$ operator.
Example 3

Give a regular expression for the language $L$ over $\Sigma = \{a, b\}$ of words that contain a number of $b$’s that is evenly divisible by 3.

- The set of all strings over $\Sigma$ that contain exactly 3 $b$’s is denoted by the regular expression $a^*ba^*ba^*ba^*$.
- $L$ is denoted by $(a^*ba^*ba^*b)^*a^*$. 
A Distributive Law for Regular Expressions

Theorem Let $r_1$, $r_2$, and $r_3$ be regular expressions. $L((r_1 + r_2)r_3) = L(r_1r_3 + r_2r_3)$.

Proof

$L((r_1 + r_2)r_3) = \{wx|w \in L(r_1 + r_2), x \in L(r_3)\}$

$= \{wx|w \in L(r_1) \cup L(r_2), x \in L(r_3)\}$

$= \{wx|w \in L(r_1) \text{ or } w \in L(r_2), x \in L(r_3)\}$

$= \{w|w \in L(r_1)L(r_3) \text{ or } w \in L(r_2)L(r_3)\}$

$= \{w|w \in L(r_1r_3) \text{ or } w \in L(r_2r_3)\}$

$= \{w|w \in L(r_1r_3) \cup L(r_2r_3)\}$

$= L(r_1r_3 + r_2r_3)$. 

11
Questions to ponder

• Given regular expression $e$, is $\overline{L(e)} = \Sigma^* - L(e)$ a regular language?
• If so, is there a mechanical way of generating a regular expression for it, given $e$?
• Given regular expressions $e$ and $f$, is $L(e) \cap L(f)$ a regular language?
• If so, is there a mechanical way of generating a regular expression for it, given $e$ and $f$?