Regular Languages and Regular Expressions

COMS3003A: Lecture Note 2

March 18, 2021

Outline

Table of Contents

Introducing Regular Languages

Recap from last week

Languages that can be accepted by finite automata are *regular languages*

Regular languages are the languages obtained from one-element languages by repeated applications of certain basic operations.

Regular languages can be described by either regular expressions or regular grammars

Regular Languages: Definition

Definition:
Let Σ be an alphabet. The class R of regular languages over Σ is defined as follows:

1. $\emptyset$ is an element of R.
2. $\{\lambda\}$ is an element of R.
3. For each $a \in \Sigma$, $\{a\}$ is an element of R.
4. If L_1 and L_2 are any elements of R, then
   a. $L_1 \bigcup L_2$ is an element of R.
   b. $L_1 L_2$ is an element of R.
   c. ${L_1}^*$ is an element of R.

Only languages that can be obtained by using statements 1–4 are regular languages.

Examples of Regular Languages

Let $\Sigma = \{0, 1\}$. Some regular languages over $\{0, 1\}$ are:

1. $\emptyset$
2. $\{\lambda\}$
3.
   1. $\{0\}$
   2. $\{1\}$
4.
   a. $\{0\} \bigcup \{1\} = \{0, 1\}$
   b. $\{0\}\{1\} = \{01\}$, $\{1\}\{0\} = \{10\}$
   c. $\{0\}^*$, $\{1\}^*$

Further Examples of Regular Languages

1. $\{0\} \bigcup \{10\} = \{0, 10\}$
2. $\{0\}\{01\} = \{001\}$
3. $(\{1\} \bigcup \{\lambda\})\{001\}$
4. $\{110\}^*\{0, 1\}$
5. $\{10, 111, 11010\}^*$
6. $\{0, 10\}^*(\{11\}^* \bigcup \{001, \lambda\})$

Table of Contents

Introducing Regular Expressions

A regular language can therefore be described by an explicit formula. By convention, we may simplify the formula slightly:

We

- leave out $\{\}$ or replace with $()$, and
- replace $\bigcup$ with $+$,

to obtain a regular expression.

Take care not to confuse the two notations—either write $\{0,1\}^*$, or $(0+1)^*$, but not $\{0+1\}^*$.

Examples of Regular Expressions

Some regular expressions are:

1. $\emptyset$
2. λ
3. 0; 1
4.
   a. $0 + 1$
   b. 01, 10
   c. 0^*, 1^*

Further examples of Regular Expressions

1. $0 + 10$
2. 001
3. $(1 + \lambda)\, 001$
4. $(110)^* (0 + 1)$
5. $(10 + 111 + 11010)^*$
6. $(0 + 10)^* (11^* + (001 + \lambda))$

Regular Expressions: Definition

Definition:

Let Σ be an alphabet.
The class RE of regular expressions over Σ is defined as follows:

1. $\emptyset$ is an element of RE.
2. λ is an element of RE.
3. For each $a \in \Sigma$, a is an element of RE.
4. If r_1 and r_2 are any elements of RE, then
   a. $r_1 + r_2$ is an element of RE.
   b. $r_1 r_2$ is an element of RE.
   c. ${r_1}^*$ is an element of RE.

Only expressions that can be obtained by using statements 1–4 are regular expressions.

Simplifications of Regular Expressions

1. Exponential notation: write (rr) as $\left(r^2\right)$
2. 'Plus' notation: write $((r^*)\, r)$ as (r^+)

If we form regular expressions strictly according to the definition, they are fully parenthesized. In order to simplify such expressions, we establish an order of precedence:

1. Highest: $*$
2. Then: concatenation
3. Lowest: $+$

Equivalence of Regular Expressions

If r_1 and r_2 are regular expressions, then $r_1 = r_2$ means that they correspond to the same languages.

Thus

$$(a + ((b^*)\, c)) = a + b^* c$$

However,

$$(a + b)^* \neq a + b^*$$

Some Regular Expression Identities

These regular expressions are equivalent. They are useful when you need to proof whether two REs are equivalent.

$$
\begin{aligned}
\phi^* &= \lambda \\
\lambda^* &= \lambda \\
\phi + R = R + \phi &= R \\
\lambda R = R\lambda &= R \\
R^* R^* &= R^* \\
RR^* &= R^* R \\
R + R &= R
\end{aligned}
$$

Are there more regular expression identities, find out?

Table of Contents

Example 1

The language $L \subseteq \{0,1\}^*$ of all strings of even length:
Valid strings include: 00, 01, 11, 0111, 111111 ...

Any string in L can be divided into a number of strings of even length.
Any concatenation of strings of even length produces a string in L.
Therefore we can write

$$L = \{00, 01, 10, 11\}^*$$

which corresponds to the regular expression

$$(00 + 01 + 10 + 11)^*$$

We can also write

$$L = \{\{0,1\}\{0,1\}\}^*$$

which corresponds to

$$((0+1)(0+1))^*$$

Example 2

The language $L \subseteq \{0,1\}^*$ containing an odd number of 1's:
Valid strings include: $1, 011, 1011, 0001000, 110000111, \ldots$
Any string in L must contain at least one 1, therefore we start with a string of the form

$$0^i 1 0^j$$

There is an even number of further 1's, each followed by zero or more 0's.
This we can see as a repetition of strings of the form

$$(10^m 10^n)$$

Therefore we can represent L by

$$0^* 1 0^* (10^* 10^*)^*$$

If we decide to stop the initial substring immediately after the 1, we can represent L as:

$$0^*1(0^*10^*1)^*0^*$$

We can also concentrate on the last 1 in the string:

$$(0^*10^*1)^*0^*10^*$$

or on a 1 in the middle, and make sure it has an even number of 1's on either side of it:

$$0^*(10^*10^*)^*1(0^*10^*1)^*0^*$$

There may be many ways to describe a typical element of a language, depending on which aspect we want to emphasize. There isn't always one that is the simplest or most natural.

The regular expression must be general enough to describe every string in the language, but not so general that it also describes strings not in the language.

Note: In this course, don't spend a lot of time trying to simplify a regular expression if it isn't necessary for further work.

$$(10^*10^*)^*10^*$$

is not general enough, since it doesn't include strings beginning with 0. Adding 0^* at the beginning to obtain

$$0^*(10^*10^*)^*10^*$$

rectifies that.

Example 3

Let $L \subseteq \{0,1\}^*$ be the set of all strings of length 6 or less.
A regular expression for L is

$$\lambda + 0 + 1 + 00 + 01 + \\ 10 + 11 + 000 + \ldots + 111110 + 111111$$

which isn't very elegant.
Let's first describe the set of strings of length exactly 6:

$$(0+1)\,(0+1)\,(0+1)\,(0+1)\,(0+1)\,(0+1)$$

or

$$(0+1)^6$$

To include strings of length less than 6, we allow each of the factors to be λ, i.e.,

$$(0+1+\lambda)^6$$

Example 4

Let

$$L = \{x \in \{0,1\}^* \mid x \text{ ends with } 1 \text{ and does not contain the substring } 00\}$$

We want a regular expression for L.
A string does not contain the substring 00
$\iff$ no 0 can be followed by 0
$\iff$ every 0 either comes at the end, or is followed by 1.

Every string in L ends with 1
$\Rightarrow$ copies of the strings 01 and 1 account for the entire string
$\Rightarrow$ every string in L corresponds to the regular expression $(1 + 01)^*$.

$(1+01)^*$ allows λ, which doesn't end with 1.

Therefore, $(1+01)^*$ is too general.

Possible fix - add 1 at the end - $(1+01)^*1$

Not correct, since we now exclude 01.

Add that possibility: $(1+01)^*(1+01)$

Therefore, final answer is $(1+01)^+$.

Example 5

Write a regular expression for the language of C identifiers

An identifier in C is any string of length 1 or more that contains only letters, digits, _'s and begins with a letter or _.

Let l (for 'letter') denote
$a + b + \ldots + z + A + B + \ldots + Z$

and d (for 'digit') denote
$0 + 1 + \ldots + 9$.

Written as regular expression:

Example 5

Write a regular expression for the language of C identifiers

An identifier in C is any string of length 1 or more that contains only letters, digits, _'s and begins with a letter or _.

Let l (for 'letter') denote
$a + b + \ldots + z + A + B + \ldots + Z$

and d (for 'digit') denote
$0 + 1 + \ldots + 9$.

Written as regular expression:

$$(l + \_)(l + d + \_)^*$$

Example 6

Let us assume the following are the rules to write a numeric literal in Pascal:

- First a sign,
- then one or more digits,
- then
  - a point and one or more digits,
  - *or* a point and one or more digits followed by an E, a sign and one or more digits,
  - *or* just the E, the sign and one or more digits.

If the constant is in exponential format, then no decimal point is needed. If there is a decimal point, there must be at least one digit on either side.

Let

- s (for 'sign') denote either λ, or a plus sign or a minus sign and
- p denote a point.

How would you write a regular expression to represent valid numeric literals?

Let

- s (for 'sign') denote either λ, or a plus sign or a minus sign and
- p denote a point.

How would you write a regular expression to represent valid numeric literals?

A regular expression to represent the numeric literal is:

$$sd^+(pd^+ + pd^+Esd^+ + Esd^+)$$

Discussion for the week.

Are there any practical/real-life applications or uses of regular expressions today?

A new discussion has been started on Moodle.

You will find it under the section 'Week 2: Regular Languages and Regular Expressions' (where you downloaded this note).

Do some reading, share your thoughts and see what others have to say.