Regular Expressions

Set operations provide a convenient way of specifying certain formal languages.

Let Σ be an alphabet disjoint from $\{(,),\emptyset,\bigcup,*\}$.

Regular expressions over Σ are strings over $\Sigma \cup \{(,),\emptyset,\bigcup,*\}$ defined according to the following rules:

1. The string $\emptyset$ is a regular expression.
2. If $a \in \Sigma$, then the the string a is a regular expression.
3. If α and β are regular expressions, so is the string $(\alpha\beta)$.
4. If α and β are regular expressions, so is the string $(\alpha \bigcup \beta)$.
5. If α is a regular expression, so is (α^*).
6. Only strings constructed according to the previous rules are regular expressions.

For example, $\emptyset$, a, b, (ab), $(a \bigcup b)$, $((b^*)^*)$, and $(((a \bigcup b)^*)(ab))$ are all regular expressions over $\{a, b\}$.

Finite Representation of Languages

Every regular expression defines a language. The relation is defined by the following rules, which define a function L from regular expressions to languages:

1. $L(\emptyset) = \emptyset$.

2. If $a \in \Sigma$, then $L(a) = \{a\}$.

3. If α and β are regular expressions, then $L(\alpha\beta) = L(\alpha)L(\beta)$.

4. If α and β are regular expressions, then $L(\alpha \bigcup \beta) = L(\alpha) \cup L(\beta)$.

5. If α is a regular expression, then $L(\alpha^*) = L(\alpha)^*$.

For example,

$$\begin{aligned}
L(a) &= \{a\} \\
L(b) &= \{b\} \\
L((ab)) &= \{ab\} \\
L((a \bigcup b)) &= \{a, b\} \\
L(((b^*)^*)) &= \{e, b, bb, bbb, \ldots\} \\
L((((a \bigcup b)^*)(ab))) &= \{w \in \{a, b\}^* : w \text{ ends with } ab\}
\end{aligned}$$

Examples of Regular Languages

If $L = L(r)$ for some regular expression r, then L is called a *regular language*.

Here are some examples of regular languages.

What language over $\Sigma = \{a, b\}$ is represented by $((a \bigcup b)^* a)$?

$\{w \in \{a, b\}^* \ : \ w$ end with an $a\}$

What language over $\Sigma = \{a, b, c\}$ is represented by

$$(c^*(a \bigcup (bc^*))^*)?$$

The set of all strings over Σ that do not contain the substring ac.

Exercise.

What language over $\Sigma = \{0.1\}$ is represented by

$$(0^* \bigcup (((0^*(1 \bigcup 11)))((00^*)(1 \bigcup (11)))^*)0^*))?$$

Language Generation

Regular expressions may be viewed as *language generators*, or methods for *generating* elements of a formal language.

For example, the expression $(e \bigcup b \bigcup bb)(a \bigcup ab \bigcup abb)*$, where e denotes the empty string, may be interpreted as follows:

1. Write down nothing or b or bb.

2. Either repeat the next step any number of times, or else skip it.

3. Write down a or ab or abb.

In other words, we interpret $\bigcup$ as a non-deterministic choice operation and $*$ as a repetition operation.

One key question in formal language theory is which languages can be generated in this sense via a certain specification mechanism, e.g., by regular expressions.

An example of a *non-regular* language is the set of all strings of a number of 1's followed by the same number of 0's, i.e., the set $\{e, 10, 1100, 111000, \ldots\}$. (Proving that this is a non-regular language is beyond the scope of this course, though.)

Language Recognition

Another important problem, known as *language recognition*, is to determine, for a formal language L, whether a given string w is an element of L.

Exercise.

> Design an algorithm for determining whether or not a string w is an element of
>
> $$L = \{w \in \{0, 1\}^* \ : \ w \text{ does not contain } 111\}.$$

Strings in ML

ML provides a data type **`string`**, the values of which are double-quoted character strings, such as **`"house"`** or **`"cat"`**.

The symbol ^ is used to denote the concatenation operation on strings.

```
- "house"  ^ "cat";
val it = "housecat" :  string

- size "housecat";
val it = 8 :  int
```

Note that there is a difference between a character string of length one and a single character.

The empty string is denoted by "".

```
- size "";
val it = 0 :  int
```