Lecture 2: Languages and Regular Expressions

Basic concepts

- *Alphabet* — a finite set of symbols, Σ.
- *word* (or *string*) — a finite sequence of symbols from an alphabet.

Alphabet	Words
{a, b, . . . , z}	man, abc, . . .
{0, 1}	000, 010101, . . .
{#, $, a, b, c}	#cb$, $$$, . . .

- $|w|$ — *length* of a word w, i.e. the number of symbols in w.
- e — the empty word containing no symbols, i.e. the word of zero length.
- To avoid confusion, e should not be in any alphabet.

Operations on words

- *Concatenation* merges two given words to form a new word:

$$\text{e.g. } \underline{abc}\ \underline{123} = abc123$$

  - Properties:

$$\begin{aligned} ew &= we = w \\ (uv)w &= u(vw) \end{aligned}$$

- *Reversal* reverses the order of all the symbols in a given w.

$$w = a_1 \cdots a_n \Rightarrow w^R = a_n \cdots a_1$$

  - Inductive definition:

$$\begin{array}{ll} (1) & e^R = e \\ (2) & (au)^R = u^R a, \text{ where } a \in \Sigma, u \in \Sigma^* \end{array}$$

- *Power* concatenates n copies of w to form a new word

$$w^n = \overbrace{ww \cdots w}^{n}$$

  - Inductive definition:

$$\begin{array}{ll} (1) & w^0 = e \\ (2) & w^{i+1} = w^i w, \text{ for any } i \geq 0. \end{array}$$

Theorem

$$(uw)^R = w^R u^R, \text{ where } u, w \in \Sigma^*$$

Proof: Prove by induction on $|u|$.

- *Basis step*: $|u| = 0$, i.e. $u = e$.

$$(ew)^R = w^R = w^R e = w^R e^R$$

- *Induction hypotheses*: Assume

$$(uw)^R = w^R u^R \text{ for } |u| \leq n,$$

- *Induction step*: Consider the case $|u| = n+1$.

Let $u = av$ for some $a \in \Sigma$ and $v \in \Sigma^*$ such that $|v| = n$.

$$\begin{aligned}
(uw)^R &= ((av)w)^R & \\
&= (a(vw))^R & \text{Associative law} \\
&= (vw)^R a & \text{Rule 2 of ind. definition} \\
&= w^R v^R a & \text{Induction hypothesis} \\
&= w^R (av)^R & \text{Rule 2 of ind. definition} \\
&= w^R u^R. &
\end{aligned}$$

Languages

- A *language* is a set of words defined over an alphabet Σ.

  Examples:

  1. Set of all English words — a language over $\{a, b, \ldots, z\}$.
  2. $\{01, 0101, 010101, \ldots\}$ — a language over $\{0, 1\}$.
  3. $\{e\}$ — a language over any alphabet.

- $\emptyset$ — the *empty language*, i.e. the language contains no words.
- Note: $\emptyset \neq \{e\}$.
- Σ^* — the set of all words over the alphabet Σ. It is called the universal language. Any language L is a subset of Σ^*.
- Connection with decision problems: A decision problem corresponds to the language that consists of all the yes-inputs.

Operations on languages

- *Concatenation:*

$$L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

- *Reversal:*

$$L^R = \{w^R \mid w \in L\}$$

- *Power:*

$$L^n = \{w_1w_2 \cdots w_n \mid w_1, w_2, \cdots, w_n \in L\}$$

  Inductive definition:

  1. $L^0 = \{e\}$.
  2. $L^{n+1} = L^nL, n \geq 0$.

- *Kleene star:*

$$L^* = \{w \in \Sigma^* \mid w = w_1w_2 \cdots w_k \text{ for some } k \geq 0 \text{ and some } w_1, \ldots, w_k \in L\}$$

  It is also called the *reflexive transitive closure* of L under concatenation.

- *Plus:*

$$L^+ = LL^*$$

  It is also called the *transitive closure* of L under concatenation.

Examples:

Let $\Sigma = \{a, b\}$, $L_1 = \{a, ab\}$, and $L_2 = \{e, ba\}$.

Then

$L_1^R = \{a, ba\}$.

$L_1 L_2 = \{a, ab, aba, abba\}$.

$L_1^2 = L_1 L_1 = \{aa, aab, aba, abab\}$.

$L_2^2 = L_2 L_2 = \{e, ba, baba\}$.

$\Sigma^* = \{e, a, b, aa, ab, ba, bb, aaa, \cdots\}$.

$\{e\}^{1000} =?$

$L\emptyset =?$

$\emptyset^* =?$

$e \notin L^+?$

$L^n \subseteq L^{n+1}?$

$L^n \subseteq L^*?$

1. Prove that $(w^R)^R = w$ for any string w.
2. Prove that $\{e\}^* = \{e\}$.
3. Prove that for any language L, $(L^*)^* = L^*$.

Regular expressions

Regular expressions are a *finite* representation of languages.

Inductive definition of *regular expressions* for languages over an alphabet Σ. A regular expression is a string over alphabet $\Sigma_1 = \Sigma \cup \{(,), \emptyset, \cup, ^*\}$.

1. $\emptyset$ and each $\sigma \in \Sigma$ are regular expressions.
2. If α and β are regular expressions, then

   $$(\alpha\beta), (\alpha \cup \beta), \alpha^*$$

   are regular expressions.
3. Nothing else is a regular expression.

Examples:
Let $\Sigma = \{a, b, c, d\}$.

- Regular expressions:

  $$a, \quad ((a{\cup}b)^*d), \quad (c^*(a{\cup}(bc^*)))^*, \quad \emptyset^*$$

- Not regular expressions:

  $$c{\cup}^*, (*)$$

Language represented by regular expressions

Let α denote a regular expression.
Let $L(\alpha)$ denote the language represented by a regular expression α.

The function L is defined as follows:

1. $L(\emptyset) = \emptyset$, $L(a) = \{a\}$ for each $a \in \Sigma$,
2. $L(\alpha\beta) = L(\alpha)L(\beta)$,
3. $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$,
4. $L(\alpha^*) = L(\alpha)^*$.

Example: What is $L[((a \cup b)^* a)]$?

$$\begin{aligned} L[((a \cup b)^* a)] &= L((a \cup b)^*)L(a) \\ &= (L(a \cup b))^* L(a) \\ &= (L(a) \cup L(b))^* L(a) \\ &= (\{a\} \cup \{b\})^* \{a\} \\ &= \{w \in \{a, b\}^* \mid w \text{ ends with } a\} \end{aligned}$$

Example: What is $L[(c^*(a \cup (bc^*))^*)]$?
An example of strings in the language is $cccaabcccbaaabbcca$.

Examples:

1. Write a regular expression for each of the following languages defined over $\Sigma = \{0, 1\}$:

   (a) $L = \{w \mid w \text{ contains at least two zeros}\}$

   (b) $L = \{w \mid w \text{ is of even length}\}$

   (c) $L = \{w \mid w \text{ has even number of 1's}\}$

2. Simplify $\emptyset^* \cup a^* \cup b^* \cup (a \cup b)^*$.

3. Simplify $(a \cup b)^* a (a \cup b)^*$.

4. Prove that

   $L[c^*(a\cup(bc^*))^*] = \{w \in \{a, b, c\}^* \mid w$ does not contain substring $ac\}$

   Proof:

   Suppose $w \in L[c^*(a\cup(bc^*))^*]$
   $\Rightarrow$ each occurrence of a in w is either at the end of the string, or is followed by another occurrence of a, or is followed by an occurrence of b.
   $\Rightarrow$ w does not have the substring ac.

   Suppose w is a string that does not contain ac.
   $\Rightarrow$ Let $w = uv$ where u consists of zero or more c's, then v has no substring ac and does not begin with

c.
$\Rightarrow v$ is a sequence of a's, b's and c's with any blocks of c's appearing only immediately after b's, not after a's and not at the beginning of the string. Thus $v \in L((a \cup bc^*)^*)$.
$\Rightarrow w \in L(c^*(a \cup bc^*)^*)$.

Notational simplifications:

1. A regular expression α also denotes the language $L(\alpha)$ represented by α. E.g., we may write $ab \in a^*b^*$.

2. Omit extra parentheses. E.g.,

   $(a \cup b) \cup c = a \cup (b \cup c) = a \cup b \cup c$

   $(ab)c = a(bc) = abc$

   $a \cup (bc) = a \cup bc \neq (a \cup b)c$

Regular language: A language that can be specified as a regular expression.

Closure Properties:

If A and B are two regular languages. Then the following languages are also regular

$$AB,\ A \cup B,\ A^*,\ A^R.$$

Proof:
Since A and B are regular languages, by definition, they can be represented by some regular expressions. Let $A = L(\alpha)$, $B = L(\beta)$, where α, β are regular expressions. Then we have

- $AB = L(\alpha)L(\beta) = L(\alpha\beta)$

That is, $\alpha\beta$ is a regular expression representing the language AB. Thus AB is a regular language.

The proofs for $A \cup B$ and A^* being regular are similar.

Try to prove yourself that A^R is regular, given A is regular.

Language generators vs language recognizers

A *Language generator* (e.g. a regular expression) represents a language by generating the words in the language

$$(c^*(a\cup(bc^*))^*) \Rightarrow$$

$$\{w\in\{a,b,c\}^* : w \text{ does not contain substring } ac\}.$$

A *language recognizer* (e.g., an algorithm) represents a language by recognizing its words.

Algorithm: recognizer(w)

- Input: w — a string.
- Output: YES or No.

1. If w=e, return YES.
2. flagA=FALSE.
3. Scan w from left to right. For each symbol:
   - If the current symbol is “a”, flagA=TRUE;
   - Else if the current symbol is “c”,
     (a) If flagA=TRUE, return NO.
     (b) flagA=FALSE.
   - Else flagA=FALSE.
4. Return YES.

$$\{w \in \{a, b, c\}^* : \text{ recognizer(w)=YES}\} =$$

$$\{w \in \{a, b, c\}^* : w \text{ does not contain substring } ac\}.$$