THLR Quick Sheet

Author: Matthieu Stombellini
Last revision: 2019-09-03

EPITA did not give lecture notes this year, so I just made some.

Definitions

- **Alphabet**: A *finite* set of symbols (denoted Σ)
- **Word**: A finite sequence of symbols from Σ (denoted u)
- **Language**: A set of words on Σ (L). It does not have to be finite.
- $\boldsymbol{\Sigma}^*$ is the set of all the words that can be built from the alphabet Σ.
  e.g.: $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, \dots\}$
- $\boldsymbol{\varepsilon}$ (also denoted 1 or λ) is the "empty word", the word that has no symbols.
- $\emptyset$ denotes the empty language: the language with no words.

Operations on words

- **Concatenation**:
  $$u, v \in \Sigma^*, u = u_0 \dots u_n, \text{then } u \cdot v = u_0 \dots u_n\ v_0 \dots v_m$$

  e.g.: The result of the concatenation of "foo" and "bar" is "foobar"
  Concatenation is a **free monoid**, because it:
  - Is **stable:** the concatenation of a word to another word is a word
  - Is **associative:** $(u \cdot v) \cdot w = u \cdot (v \cdot w) = u \cdot v \cdot w$
  - Has a **neutral element:** ε because $a \cdot \varepsilon = \varepsilon \cdot a = a$
  - Gives a **unique decomposition** of words

  The concatenation behaves like a product.

  Another way to express concatenation is with an exponentiation notation
  $$u^n = \underbrace{u \cdot \dots \cdot u}_{n \text{ times}}$$
  $$u^0 = \varepsilon$$
- **Length:**
  The length of a word u is denoted $|u|$

Relations between words

- **Subword**
  $$u \text{ subword of } v$$
  $$\exists u_1, u_2 \in {\Sigma^*}^2, v = u_1 \cdot u \cdot u_2$$
- **Prefix**[1]
  $$u \text{ prefix of } v \quad u \preccurlyeq_p v$$
  $$\exists w \in \Sigma^*, v = u \cdot w$$
  e.g.: "ban" is a prefix of "banana".
  By this definition, "banana" is also a prefix of "banana" (in which case $w = \varepsilon$). A "proper prefix" is a prefix that is not the word itself (so a prefix such that $w \neq \varepsilon$)

  Prefix is an order relation but not a total one. It is:
  - **Reflexive:** $u \preccurlyeq_p u$
  - **Transitive:** if $u \preccurlyeq_p w$ and $v \preccurlyeq_p w$, then $u \preccurlyeq_p w$
  - **Antisymmetric:** if $u \preccurlyeq_p v$ and $v \preccurlyeq_p u$ then $u = v$
- **Suffix**[2]
  $$u \text{ suffix of } v \quad u \preccurlyeq_s v$$
  $$\exists w \in \Sigma^*, v = w \cdot u$$
- **Subsequence** or **scattered subwords:** Non-contiguous sequences from the original word that still respect the order in which symbols

[1] Prefixes are just a particular case of subwords (where $u_2 = \varepsilon$)

[2] Suffixes are just a particular case of subwords (where $u_1 = \varepsilon$)

appear in the original word. You can see that as just the original word from which you remove some symbols.
e.g.: “bd” is a subsequence of “abcde”

- **Lexicographic order:**

$$u \leq_{lex} v \Leftrightarrow \begin{cases} u = w \cdot u_0 u' \\ v = w \cdot v_0 v' \end{cases} \text{ with } u < v \\ \text{or } u \text{ prefix of } v$$

We cannot enumerate all the words since we would only get an infinity of $aaaa$ when running recursively.
e.g.: egg, example, reminded, reminder, road

- **Alphabetical/radicial/military order:**

$$u \leq v \Leftrightarrow \begin{matrix} |u| < |v| \\ \text{or } |u| = |v| \text{ and } u \leq_{lex} v \end{matrix}$$

e.g.: egg, road, example, reminded, reminder

Operations on languages

Let L_1 and L_2 be languages on the same alphabet Σ. We can have different operations:

- Since languages are sets, all the usual operations on sets apply
  - $L_1 \cup L_2$
  - $L_1 \cap L_2$
  - $\overline{L_1}$
  - $L_1 \setminus L_2$
  - $\emptyset$ (empty set, the language with no words)
- We can also lift operations on words into operations on languages, like **concatenation:**

$$L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1, v \in L_2\}$$

e.g.: $\{a\} \cdot \{b\} = \{ab\}$

$$L^n = \underbrace{L \cdot \ldots \cdot L}_{n \text{ times}}$$

$$L^0 = \{\varepsilon\}$$

This is associative, stable (a language concatenated with another language is still a language) and has a neutral element $\{\varepsilon\}$

- **Kleene star**

$$L^* = \bigcup_{n \in \mathbb{N}} L^n$$

It can also be defined as $u \in L^* \Leftrightarrow \exists n \in \mathbb{N}, u \in L^n$

$$\text{e.g. } \{a,b\}^* = \left\{ \underbrace{\varepsilon}_{L^0}, \underbrace{a, b}_{L^1}, \underbrace{aa, ab, ba, bb}_{L^2}, aaa \ldots \right\}$$

$$\{a\}^* = \{\varepsilon, a, aa, aaa, aaaa, \ldots\}$$

Note that L^* always includes the empty word ε (since $L^0 = \{\varepsilon\}$).
There is another notation which does *not* include the empty word ε:

$$L^+ = \bigcup_{n \geq 1} L^n$$

$$\text{e.g. } \{a,b\}^+ = \left\{ \underbrace{a, b}_{L^1}, \underbrace{aa, ab, ba, bb}_{L^2}, aaa \ldots \right\}$$

- **Language of prefixes**

$$\text{Pref}(L) = \{u \in \Sigma^* | \exists v \in \Sigma^*, u \cdot v \in L\}$$

In other words: $Pref(L)$ is the set of all the prefixes of the words of L.

- **Language of suffixes**

$$\text{Suff}(L) = \{u \in \Sigma^* | \exists v \in \Sigma^*, v \cdot u \in L\}$$

In other words: $Suff(L)$ is the set of all the suffixes of the words of L.

- **Language of subwords**

$$\text{Frac}(L) = \left\{u \in \Sigma^* \middle| \exists (v, w) \in \Sigma^{*2}, v \cdot u \cdot w \in L\right\}$$

In other words: $Frac(L)$ is the set of all the subwords of the words of L.

Recursive(ly enumerable)

- A set is **recursive** (or **decidable**) when there exists a program (called the *indicator function*) which *always terminates* which tests membership in the set.
- A set is **recursively enumerable** when there exists a program which generates each element of the set.

Rational expressions

- The following operations are said to be the "rational operations":
  - $\emptyset$
  - $\{\varepsilon\}$
  - $\{a\}, a \in \mathbb{Z}$
  - $L_1 \cup L_2$
  - $L_1 \cdot L_2$
  - ${L_1}^*$
- Languages that can be described using only rational operations are said to be **rational languages**.
- The operations can also be described using a better syntax: **rational expressions.** e and f are languages here.

Set theory syntax	Rational expression syntax
$\emptyset$	$\emptyset$
$\{\varepsilon\}$	ε
$\{a\}, a \in \Sigma$	a
$L_1 \cup L_2$	$e + f$
$L_1 \cdot L_2$	$e \cdot f$
L^*	e^*

- There are multiple ways to express the same languages. We say that two rational expressions are **equivalent** if they describe the same language.
  Here is a list of elementary equivalences

$$\emptyset e \equiv \emptyset \qquad \varepsilon e \equiv e$$

$$e\emptyset \equiv \emptyset \qquad e\varepsilon \equiv e$$

$$\emptyset^* \equiv \varepsilon \qquad \varepsilon^* \equiv \varepsilon$$

$$e + f \equiv f + e \qquad e + \emptyset \equiv e$$

$$e + e \equiv e \qquad (e^*)^* \equiv e^*$$

$$(ef)^* e \equiv e(fe)^* \qquad (e + f)g = eg + fg$$

$$(e + f)^* \equiv e^*(e + f)^* \qquad e(f + g) \equiv ef + eg$$

$$(e + f)^* \equiv (e^* f^*)^* \qquad (e + f)^* \equiv (e^* + f)^* \equiv (e^* f)^* e^*$$

- Additional syntaxes exist for writing cleaner expressions exist. They are not new operations, just shortcuts. They will appear in some exercises and in regular expressions (regex).
  - **Optional** $e^?$
    $$e^? = (e + \varepsilon)$$
    This simply makes this whatever it is applied to optional.
  - **Kleene plus** e^+
    $$e^+ = ee^*$$
    Just like when we were working with sets, we can use this notation to denote "1 or more" occurrences of e, while the Kleene star denotes "0 or more" occurrences.
  - **Character set** $[abc]$
    Take for example the alphabet $\Sigma = \{a, b, c, d, e, f\}$
    $$[abcd] = (a + b + c + d)$$
    This is similar to saying "match one of the characters in this set".
    We can negate this.
    $$[\hat{}ab] = [cdef] = (c + d + e + f)$$

Here, this simply means “match *one* character that is *not* in this set”.
We can also use ranges of letters instead of writing all of them. For example, if Σ is the Latin alphabet, then

$$[a - m]$$

will match a single character from a to m, all lowercase.[3] We can also use multiple ranges in the same character set.

$$[a - mA - X]$$

This will match a single lowercase letter from a to m and uppercase letters from A to X.
We can, of course, use quantifiers on our character set, for example $[a - c]^* = [abc]^* = (a + b + c)^*$.

[3] This implies that there is an order in our alphabet. For example, this order can be the natural order of numbers (e.g. $[0 - 9]$), lowercase or uppercase letters of the alphabet $[a - z]$ or $[A - Z]$)