Regular Expressions

Alyssa Lytle

Fall 2025

1 Another way to think about lanugages

Definition 1

Let Σ be a finite alphabet. A *pattern* is a string of symbols representing a set of strings in Σ^*

Definition 2

A string *matches* a pattern $\iff$ it's accepted by that pattern's language.

$$L(\alpha) = \{x \in \Sigma^* | x \text{ matches } \alpha\}$$

Application

The UNIX commmands `grep`, `fgrep`, and `egrep` are basic pattern-matching utilities that use DFA/NFA hybrids in their implementation! [Koz07]

1.1 Atomic Patterns and Compound Patterns

The *atomic patterns* are:[Koz07]

- a, for each $a \in \Sigma$; $L(a) = \{a\}$
- $\boldsymbol{\epsilon}$; $L(\boldsymbol{\epsilon}) = \{\epsilon\}$
- $\boldsymbol{\emptyset}$; $L(\boldsymbol{\emptyset}) = \emptyset$.
- $\#$ (any symbol in Σ); $L(\#) = \Sigma$
- $@$ (any string in Σ^*); $L(@) = \Sigma^*$

Compound patterns are made using the operators:

- $L(\alpha \cup \beta)$ (or $L(\alpha + \beta)$)$= L(\alpha) \cup L(\beta)$ [1]
- $L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$
- $L(\alpha \circ \beta) = L(\alpha) \circ L(\beta) = \{yz | y \in L(\alpha) \text{ and } z \in L(\beta)\}$
- $L(\sim \alpha) =\sim L(\alpha) = \Sigma^* - L(\alpha)$
- $L(\alpha^*) = \{x_1 x_2 \ldots x_n | n \geq 0 \text{ and } x_i \in L(\alpha), 1 \leq i \leq n = L(\alpha)^*$

[1] Kozen uses $+$ and Sipster uses $\cup$. [Koz07, Sip96]

[Koz07, Sip96]

You'll note that $\Sigma^* = L(@) = L(\#^*)$. In fact, a few equivalences like this can be made!

$\#$ and α are both considered redundant "base cases" since they can be covered by applying compound patters to a single atomic pattern a.

Similarly, $\cap$ is redundant because of DeMorgan's! $(\alpha \cap \beta \equiv \sim (\sim \alpha + \sim \beta)$

$\sim$ is also redundant, but this is a complicated proof, so just trust me on this one!

Therefore we can reduce our "set" of notations for patterns to be:

- a
- $\boldsymbol{\epsilon}$
- $\boldsymbol{\emptyset}$
- $\cup$
- $\circ$
- $*$

1.2 Regular Expressions

You can use an expression to represent a language. Unsurprisingly, *regular expressions* are used to represent regular languages.

Theorem 1

Let $A \subseteq \Sigma^*$. The following three statements are equivalent:

1. A is a regular language
2. $A = L(\alpha)$ for some pattern α
3. $A = L(\alpha)$ for some regular expression α

Definition 3

The set of regular expressions can be defined *inductively* using atomic patterns and operators.

R is a regular expression if R is

1. $a \in \Sigma$
2. $\boldsymbol{\sigma}$
3. $\boldsymbol{\emptyset}$
4. $R_1 \cup R_2$ where R_1 and R_2 are regular expressions
5. $R_1 \circ R_2$ where R_1 and R_2 are regular expressions
6. R_1^* where R_1 is a regular expression

References

[Koz07] Dexter C Kozen. *Automata and computability*. Springer Science & Business Media, 2007.

[Sip96] Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.