Measures on Languages

1 Introduction

1.1 Regular Languages

Definition 1 (Alphabet)**.** An alphabet Σ is a countable set of distinct *characters*. $\square$

Example 1. The English alphabet of 26 (lower-case) characters forms an alphabet where we have $\Sigma = \{a, b, \ldots, y, z\}$. $\square$

Definition 2 (Language)**.** Let Σ^* be the set of all (possibly countably infinite) strings of a language. A language L is a subset $L \subseteq \Sigma^*$. $\square$

Definition 3 (Regular Language)**.** A regular language L_R is a language that can be *recognized* by a *deterministic finite automata* (DFA). In other words, there exists a DFA A_{L_R} such that:

$$\forall s \in L_R : A_{L_R}(s) = 1$$

$\square$

It is well known that DFAs and regular languages form a bijection. Furthermore, all DFAs can be reduced to a minimal form. If a DFA A_{L_R} recognizes a regular language L_R, write $\eta(A_{L_R})$ to be a canonical minimal DFA. DFA's can also be described via *regular expressions*.

Definition 4 (Regular Expression)**.** A regular expression E is a finite string over an alphabet Σ defined inductively, such that:

- The empty string ε is a regular expression
- If E_1 and E_2 are regular expressions, then $E_1 + E_2$ is a regular expression
- If E_1 and E_2 are regular expressions, then $E_1 \cdot E_2$ is a regular expression
- If E is a regular expression, then E^* is a regular expression

If E is a regular expression, denote the regular language generated by it as $L_R(E)$. Likewise, we abuse notation to write minimized, canonical regular expressions as $\eta(E)$. $\square$

Regular expressions textually represent DFAs, and can be thought of as rules for how to generate a *regular language*. Here, the $+$ operator can be thought of as string choice, the $\cdot$ operator can be thought of as string concatenation, and the $*$ operator (Kleene star) denotes zero or more occurrences of a regular expression. We often elide $\cdot$ when the context is clear. The $\cdot$ syntactically binds tigher than $+$, meaning that "multiplication" distributes over "addition".

Example 2. Consider the alphabet $\Sigma = \{a, b, c, d\}$, with regular expression $E = ab + c + d^*$. Then the regular language $L_R(E)$ is the set:

$$L_R(E) = \{ab, c, \varepsilon, d, dd, ddd, \dots\}$$

$\square$

Finite language union, intersection, and complement with respect to a common alphabet is also well-defined for regular languages.

1.2 Measure Theory

Some talk about measure theory goes here

2 Pre-Measures on Regular Languages

We observe a few things. First, regular languages are inherently finite in representation. Furthermore, they are also closed under finite language union, intersection, and complement. This means that they form an algebra under these operations. Therefore, it makes sense to begin by defining a pre-measure on regular languages.

2.1 Counting Pre-measure

Let $\mu_{0,c}$ be the counting pre-measure defined for countable sets, such that:

$$\mu_{0,c}(L) = |L|$$

For a regular expression E, we wish to say something meaningful about the pre-measure of its generated regular language $\mu_{0,c}(L_R(E))$. Perhaps $\mu_{0,c}$ may be defined inductively in terms of the regular expression as follows:

$$\begin{aligned}
\mu_{0,c}(\varepsilon) = 0 \qquad \forall a \in \Sigma : \mu_{0,c}(a) = 1 \\
\eta(E) = E_1 + E_2 \implies \mu_{0,c}(E) = \mu_{0,c}(E_1) + \mu_{0,c}(E_2) \\
\eta(E) = E_1 \cdot E_2 \implies \mu_{0,c}(E) = \mu_{0,c}(E_1) \cdot \mu_{0,c}(E_2)
\end{aligned}$$