# Metric Spaces for Regular Languages

Anton Xue
Adviser: Dana Angluin

Yale Computer Science Fall 2018 Senior Thesis

# Contents


**1 Introduction** **3**

**2 Background** **4**

2.1 Regular Languages and Finite Automata . . . . . 4

2.2 Metric Spaces . . . . . 6

2.3 Measure Theory . . . . . 6

2.4 Measure Induced Metric . . . . . 8

**3 Measure Theoretic Approaches** **9**

**4 Linear Operators** **11**

4.1 Automata Metrics . . . . . 14

4.1.1 Edit Distance Metrics . . . . . 14

**5 Separating Automata** **17**

**6 Bi-Directional Merging** **20**

**7 Conclusion and Future Work** **22**

7.1 Graph Kernels . . . . . 22

**A Bi-Merge Code** **24**

**Abstract**

Often an approximation is sufficient. It may be the case that a complex system can be well-approximated by the behavior of a much smaller and simpler proxy. A good proxy is one in which the representation is small yet the error of approximation can be rigorously bounded; it allows us to easily study and understand complex behavior at a simplified model.

In this work we study how regular languages can be embedded into metric spaces. This embedding allows us to rigorously discuss what it means for two regular languages to be similar: exactly what it means them to share many strings and differ on few. A good solution to this problem allows us to introduce novel approximation techniques to areas such as language learning, program synthesis, and software verification. For instance, achieving results in learning, synthesis, and verification that are not exact, but are good enough, may still be invaluable.

Our work explores this problem in the context of measure theory, algebraic formalization, and language separation. We demonstrate how the space of languages can be embedded into a measure space, from which a metric space can be induced. Additionally, we give an algebraic formulation of regular languages and non-deterministic finite automata, with an attention to linear spaces over semirings. Furthermore, we describe how to use boolean satisfiability to compute a minimal non-deterministic finite automata that can be used to classify two disjoint sets of strings. Finally, we present graphical visualization techniques for sets of strings that will benefit future research endeavors, with relevant code attached in the appendix.

# 1 Introduction

Language recognition is a fundamental problem in computer science. Given an alphabet of unique symbols $\Sigma$, let $\Sigma^\star$ denote the set of all possible finite strings over the alphabet $\Sigma$. A language $L \subseteq \Sigma^\star$ is nothing more than a set of strings, and we use these terms interchangeably. For some string $w \in \Sigma^\star$, we can then ask if $w \in L$. This is the language recognition problem, and is no different than a question about set membership.

A number of problems in theoretical computer science can be formulated in terms of language recognition. Does this string belong to the set (language) of valid email addresses? Does this stirng belong to the set (language) of valid computer programs written in my favorite programming language? Does this string belong to the set (language) of solutions to an instance of the boolean satisfiability problem?

Of the many questions that can be formulated in terms of language recognition, one of central questions pertains to recognition of regular languages. Regular languages are simple yet powerful: they are indispensible in their ability to model finite state machines, describe structural patterns in strings, and behave as a simple programming language model. Abstractly, regular languages are precisely the set of languages that can be described by the class of regular expressions. Equivalently, regular languages are the family of languages that are recognized by (non)-deterministic finite automata.

However, because of the way they are defined, a "flaw" is that two regular languages $L_1$ and $L_2$

may differ on a very small number of strings, yet have very different representation sizes. For two regular language $L_1$ and $L_2$ specified by, for instance, regular expressions. The set of strings shared $L_1 \cap L_2$ may be a large fraction of both languages, and we may wish to consider $L_1$ and $L_2$ to be similar. On the other hand, the (minimal) regular expression used to describe and represent $L_1$ may be very large, while the (minimal) regular expression for $L_2$ may be significantly smaller. So how different is $L_1$ from $L_2$? In other words, how might we assign a metric between $L_1$ and $L_2$, or more generally, over the space of regular languages?

One may attempt a classification with respect to their syntactic metric. That is, a metric distance is defined with respect to a representation without direct consideration for what strings the two sets may represent. This would be a metric over the representation, be it regular expressions, finite automata, or some other means.

Alternatively, one may seek to determine a metric in terms of the sets of strings in $L_1$ and $L_2$, which would be a semantic metric. This metric would be independent of representation, and would be defined over the space of regular sets.

Ideally, we would like a pair of metrics on the space of regular languages, one syntactic and one semantic, that are highly correlated. That is, the syntactic metric between $L_1$ and $L_2$ is small if and only if the semantic metric is also small. We set out to explore the different types of metrics for regular languages.

In this report, we present our efforts and progress towards various metric spaces on the space of (regular) languages. Our efforts span exploration into measure theory, rediscovering embedding of formal language theory into algebraic formulations, using automata to separate sets, and also interesting visualization techniques.

# 2 Background

We now introduce and give a quick overview of some of the background material relevant.

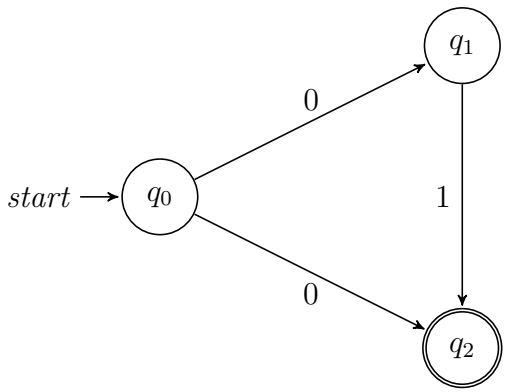
## 2.1 Regular Languages and Finite Automata

A NFA is a tuple $(\Sigma, Q, \delta, S, F)$ that represents a finite state transition machine which accepts or rejects strings. Here $\Sigma$ is the alphabet, $Q$ is the set of states, $\Delta : \Sigma \times Q \to 2^Q$ is the transition function, $S \subseteq Q$ is the set of initial states, and $F \subseteq Q$ is the set of final states.

A NFA accepts a string if there exists a sequence of transition starting from some $q_s \in S$ that ends in $q_f \in F$. As the transition function maps to a set of possible states that may be arbitrarily chosen, only the existence of a transition sequence is necessary, hence the term non-deterministic.

The size of an NFA is the number of states $|Q|$.

**Example 1** (NFA)**.** *The NFA below operators over the binary alphabet* $\Sigma = \{0, 1\}$.




*Here we would have the following configuration:*

$$\begin{aligned}
&\Sigma = \{0, 1\} \\
&Q = \{q_0, q_1, q_2\} \\
&\Delta = \{((0, q_0), \{q_1, q_2\}), \ ((1, q_1), \{q_2\})\} \\
&S = \{q_0\} \\
&F = \{q_2\}
\end{aligned}$$

*Of course the size of this NFA is* $|Q| = 3$.

*In order for a NFA to accept a string, there must exist a sequence of transitions (which may be non-unique). This particular NFA accepts precisely two strings:*

*(1) The string* 0 *through the transition sequence* $q_0 q_2$.

*(2) The string* 01 *through the transition sequence* $q_0 q_1 q_2$.

*Note that from state* $q_0$ *there are two out-edges that are both weighted with* 0. *This is what differentiates an NFA from a deterministic finite automata (DFA). In an NFA, out-edges from the same vertex may have shared labels, but in a DFA all out-edges from the same vertex may not share labels.*

A very nice property about regular languages is that they are closed under the basic set operations of complementation, union, and intersection, with only polynomial complexity in representation changes: be it regular expressions or NFAs.

We sketch a quick illustration for why infinite set operations may not work. Suppose that we define a sequence of languages over $\Sigma = \{a, b\}$ as follows:

$$L_1 = \{ab\} \qquad L_2 = \{aabb\} \qquad L_3 = \{aaabbb\} \qquad \ldots$$

Then the union effectively describes the following language:

$$\bigcup_{k=1}^{\infty} L_k = \left\{a^k b^k \ : \ k \in \mathbb{Z}^+\right\}$$

Where $a^k b^k$ is short-hand for a string that is $k$ consecutive $a$'s followed by $k$ consecutive $b$'s. It is known that such a language is not regular, and this is therefore an example where regular languages are not closed under infinite set operations.

## 2.2 Metric Spaces

The concept of a distance is formalized in mathematics through a metric space. A metric space is a pair $(M, d)$ where $M$ is a set and $d : M \times M \to \mathbb{R}$ is known as the metric, or distance, function that aims to assign a distance between any two members of $M$. A metric space comes equipped with the following axioms that must hold for any $x, y, z \in M$:

(1) Non-negativity of $d$: $d(x, y) \geq 0$.

(2) Identity of indiscernibles: $d(x, y) = 0$ if and only if $x = y$.

(3) Symmetry: $d(x, y) = d(y, x)$.

(4) Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$.

**Example 2** (Euclidean Metric)**.** *For some $x_i \in \mathbb{R}^n$, let $x_i$ be the ith coordinate of the vector. Then the Euclidean (L2) metric is defined by*

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

## 2.3 Measure Theory

In mathematical analysis, measure theory is concerned with the rigorous formulation of "size". Such notions of size have application in generalizations of familiar concepts such as length, area, and volume, as well as integration theory. Informally, for a set $X$, the goal of measure theory is to assign a measure (size) to subsets of $X$. Often $X$ is taken to be a space like $\mathbb{R}^n$, and common examples of subsets include intervals, rectangles, or boxes.

Formally, a measurable space is a pair $(X, \mathcal{E})$ where $X$ is a set and $\mathcal{E} \subseteq 2^X$ is called a $\sigma$-algebra on $X$ that satisfies the following properties:

(1) Inclusion of empty set and whole space: $\emptyset, X \in \mathcal{E}$.

(2) Closure under relative complement: $E^c \in \mathcal{E}$ if $E \in \mathcal{E}$.

(3) Closure under countable unions: $E_1, E_2, \ldots \in \mathcal{E}$ implies that

$$\bigcup_{i=1}^{\infty} E_i \in \mathcal{E}$$

The $\sigma$-algebra defined on $X$ need not be unique. For instance, the smallest $\sigma$-algebra for any set $X$ is $\{\emptyset, X\}$, while the largest $\sigma$-algebra is the power set $2^X$.

If $E$ belongs to the $\sigma$-algebra, that is, $E \in \mathcal{E}$, we say that $E$ is measurable. Otherwise for some $F \in 2^X \setminus \mathcal{E}$ we say that $F$ is un-measurable.

A measurable space can be extended into a measure space by equipping a measure $\mu : \mathcal{E} \to \mathbb{R}$ to form a triple $(X, \mathcal{E}, \mu)$. A measure satisfies the following properties:

(1) Empty set has trivial measure: $\mu(\emptyset) = 0$.

(2) Countable additivity: if $E_1, E_2, \ldots \in \mathcal{E}$ are pairwise disjoint, then

$$\mu\left(\bigcup_{i=1}^{\infty} E_i\right) = \sum_{i=1}^{\infty} \mu(E_i)$$

**Example 3** (Lebesgue Measure)**.** *Consider the real line $\mathbb{R}$ and some interval open $(a, b) \subseteq \mathbb{R}$ with $a < b$. Intuitively one may want to assign the interval a size of equal to its length. In other words, the measure of $(a, b)$ should be $b - a$.*

*The idea of using lengths (also, area, volume, etc) as a way to measure the size of a set in $\mathbb{R}^n$ gives rise to the Lebesgue measure $\lambda$. But to formally define the Lebesgue measure, we must first define the Lebesgue outer measure $\lambda^\star : 2^{\mathbb{R}} \to \mathbb{R}$ as follows:*

*In order to define the Lebesgue measure $\lambda$, we first define the Lebesgue outer measure $\lambda^\star$:*

$$\lambda^\star(E) = \inf\left\{\sum_{i=1}^{\infty} \ell(I_i) \ : \ \{(I_i)\} \text{ is a sequence of open intervals such that } E \subseteq \bigcup_{i=1}^{\infty} I_i\right\}$$

*Where $\ell(I_i)$ is the length of interval $I_i$.*

*The difference between an outer measure and a measure is that an outer measure is defined on the largest $\sigma$-algebra, in this case $2^{\mathbb{R}}$, while a measure tends to be defined on a restricted subset.*

*The Lebesgue $\sigma$-algebra is a subset of $2^{\mathbb{R}}$ that is defined as the collection of all sets $E$ such that for any $A \subseteq \mathbb{R}$ the following property holds with respect to the Lebesgue outer-measure $\lambda^\star$:*

$$\lambda^\star(E) = \lambda^\star(A \cap E) + \lambda^\star(A \cap E^c)$$

*This is called the Carathéodory criterion, and on such sets we set $\lambda(E) = \lambda^\star(E)$. The proof that the Lebesgue measure is a measure can be found in texts on real analysis and measure theory.*

**Example 4** (Counting Measure)**.** *Given a set $X$, the counting measure is defined on $2^X$, and just counts the cardinality of each $E \subseteq X$. The counting measure tends to see application in settings dealing with finite sets.*

**Example 5** (Probability Measure)**.** *Probability measures are measures defined on the probability measure space $(X, \Omega, \mu)$, where for the whole space $\mu(X) = 1$.*

## 2.4 Measure Induced Metric

For a measure space $(X, \mathcal{E}, \mu)$, an interesting consequence is that a metric space can be defined on $\mathcal{E}$ as follows:

$$d(A, B) = \mu(A \triangle B)$$

Where $\triangle$ is the symmetric set difference. We now set out to show this.

**Lemma 1.** $(A \triangle C) \subseteq (A \triangle B) \cup (B \triangle C)$.

*Proof.* Observe that we may rewrite the above as follows:

$$(A \setminus C) \cup (C \setminus A) \subseteq [(A \setminus B) \cup (B \setminus C)] \cup [(B \setminus A) \cup (C \setminus B)]$$

It then suffices to show that:

$$A \setminus C \subseteq (A \setminus B) \cup (B \setminus C) \qquad C \setminus A \subseteq (B \setminus A) \cup (C \setminus B)$$

We take turns examining these.

If $x \in A \setminus C$, then this implies that $x \in A$ and $x \notin C$. There are now two cases, where $x \in B$ or $x \notin B$. First assume that $x \in B$, which will imply that $x \in B \setminus C$. Now assume that $x \notin B$, which will imply that $x \in A \setminus B$. Either way, the implication is that $x \in (A \setminus B) \cup (B \setminus C)$, and so it follows that $A \setminus C \subseteq (A \setminus B) \cup (B \setminus C)$.

If $x \in C \setminus A$, then this implies that $x \in C$ and $x \notin A$. The argument is similar to the above, in which either $x \in B$ or $x \notin B$. If $x \in B$, then $x \in B \setminus A$, and otherwise if $x \notin B$ implies that $x \in C \setminus B$. Collectively, the two imply that $C \setminus A \subseteq (B \setminus A) \cup (C \setminus B)$.

Collectively, this shows that $(A \triangle C) \subseteq (A \triangle B) \cup (B \triangle C)$. □

**Theorem 1.** *If $(X, \mathcal{E}, \mu)$ is a measure space, then for $d : \mathcal{E} \times \mathcal{E} \to \mathbb{R}^{\geq 0}$ defined as:*

$$d(A, B) = \mu(A \triangle B)$$

*Is a metric function.*

*Proof.* We prove the conditions necessary for a metric: identity, symmetry, and triangle inequality.

As $\mu$ is a measure, then for any $A \in \mathcal{E}$:

$$d(A, A) = \mu(A \triangle A) = \mu(\emptyset) = 0$$

By the symmetry of symmetric set difference, for any $A, B \in \mathcal{E}$:

$$d(A, B) = \mu(A \triangle B) = \mu(B \triangle A) = d(B, A)$$

For any $A, B, C \in \mathcal{E}$, we have by convexity as shown in the lemma above:

$$A \triangle C \subseteq (A \triangle B) \cup (B \triangle C)$$

Then by sub-additivity of measures:

$$d(A, C) = \mu(A \triangle C) \leq \mu((A \triangle B) \cup (B \triangle C)) \leq \mu(A \triangle B) + \mu(A \triangle C) = d(A, B) + d(B, C)$$

□

# 3 Measure Theoretic Approaches

Our initial efforts began from a simple question: is it possible to assign a notion of size to a regular language? We are interested in this perspective for several reasons. First, because regular languages are closed under finite basic set operations, being able to quantiatively measure their intersection and set difference would allow us to measure some type of similarity. Second, being able to embed regular languages into a measure space, as noted earlier, would allow us to derive a metric function with respect to the symmetric set difference. The natural course of investigation leads to measure theory.

Consider a language $L \subseteq \Sigma^\star$. The $n$-splice of a language written as $L^n$ is defined as:

$$L^n = L \cap \Sigma^n$$

We then have the following relations:

$$L = \bigcup_{n=0}^{\infty} L^n = \bigcup_{n=0}^{\infty} (L \cap \Sigma^n) = L \cap \bigcup_{n=0}^{\infty} \Sigma^n = L \cap \Sigma^\star = L$$

Suppose that $\left(\mathbb{N}, 2^{\mathbb{N}}, \eta\right)$ is a probabiliy measure space on $\mathbb{N}$ with the probability measure $\eta$, one way to define a measure $\lambda_\eta$ is as follows:

$$\lambda_\eta (L) = \sum_{n=0}^{\infty} \frac{|L^n|}{|\Sigma^n|} \eta (n)$$

**Theorem 2.** $\left(\Sigma^\star, 2^{\Sigma^\star}, \lambda_\eta\right)$ *is a measure space.*

*Proof.* Since the $2^{\Sigma^\star}$ is the largest $\sigma$-algebra on $\Sigma^\star$, it suffices to show that $\lambda_\eta$ is a measure.

To see that $\emptyset$ is mapped to 0:

$$\lambda_\eta (\emptyset) = \sum_{n=0}^{\infty} \frac{|\emptyset|}{|\Sigma^n|} \eta (n) = \sum_{n=0}^{\infty} 0 = 0$$

Now take $(A_n) \subseteq \Sigma^\star$ to be a countable collection of disjoint sets. Write $A_n^k$ to denote the $k$ splice of the $n$th set. In other words:

$$A_n = \bigcup_{k=0}^{\infty} A_n^k$$

Observe that all such $A_n^k$ are pairwise disjoint by construction, and so:

$$\lambda_\eta \left( \bigcup_{n=0}^{\infty} A_n \right) = \lambda_\eta \left( \bigcup_{n=0}^{\infty} \bigcup_{k=0}^{\infty} A_n^k \right) = \sum_{n=0}^{\infty} \lambda_\eta \left( \bigcup_{k=0}^{\infty} A_n^k \right) = \sum_{n=0}^{\infty} \sum_{k=0}^{\infty} \frac{\left|A_n^k\right|}{|\Sigma^n|} \eta (k) = \sum_{n=0}^{\infty} \lambda_\eta (n)$$

We conclude that $\left(\Sigma^\star, 2^{\Sigma^\star}, \lambda_\eta\right)$ forms a measure space.

□

We can generalize this more. Suppose that $\nu = (\nu_n)$ is a countable collection of measures where each $\nu_n$ is defined on the splice $\Sigma^n$. Then we can extend a definition of $\lambda_{\eta,\nu}$ as:

$$\lambda_{\eta,\nu}(A) = \sum_{n=0}^{\infty} \nu(A^n)\,\eta(n)$$

**Theorem 3.** $\left(\Sigma^\star, 2^{\Sigma^\star}, \lambda_{\eta,\nu}\right)$ *is a measure space.*

*Proof.* As with before, we only show that $\lambda_{\eta,\nu}$ is a measure.

For $\emptyset$ we have again:

$$\lambda_{\eta,\nu}(\emptyset) = \sum_{n=0}^{\infty} 0 = 0$$

Again take $(A_n) \subseteq \Sigma^\star$ to be a countable disjoint collection of sets, and $A_n^k$ to be the $k$ splice of $A_n$. Then:

$$\lambda_{\eta,\nu}\left(\bigcup_{n=0}^{\infty} A_n\right) = \lambda_{\eta,\nu}\left(\bigcup_{n=0}^{\infty}\bigcup_{k=0}^{\infty} A_n^k\right) = \sum_{n=0}^{\infty} \lambda_{\eta,\nu}\left(\bigcup_{k=0}^{\infty} A_n^k\right) = \sum_{n=0}^{\infty}\sum_{k=0}^{\infty} \nu_k\left(A_n^k\right)\eta(k) = \sum_{n=0}^{\infty} \lambda_{\eta,\nu}(A_n)$$

This shows that $\left(\Sigma^\star, 2^{\Sigma^\star}, \lambda_{\eta,\nu}\right)$ is a measure space. □

Having defined a measure space corresponding to the space of languages, we can leverage the earlier observation that measures induce a natural metric through their symmetric set difference.

**Example 6.** *Fix an alphabet $\Sigma$ and a probability measure $\eta$ on the non-negative integers. For the measure space $\left(\Sigma^\star, 2^{\Sigma^\star}, \lambda_\eta\right)$ and languages $L_1, L_2 \subseteq \Sigma^\star$, the distance between $L_1$ and $L_2$ can then be defined as:*

$$d(L_1, L_2) = \lambda_\eta(L_1 \triangle L_2)$$

*Provided additional measures $(\nu_n)$ defined as above that extends to a measure space $\left(\Sigma^\star, 2^{\Sigma^\star}, \lambda_{\eta,\nu}\right)$, the definition then becomes:*

$$d(L_1, L_2) = \lambda_{\eta,\nu}(L_1 \triangle L_2)$$

This embedding gives us a generalized framework that allows us to talk about all languages over $\Sigma$ rather than just regular languages. However, there are several tradeoffs that must be addressed before this metric becomes practical in both intuition and usage.

First, we did not specify the probability distributions to be used. The specific distributions of interest would most likely be problem-specific, and would require at least some justification. When formalizing this metric we had in mind a goemetric distribution, which means that this metric would then favor shorter strings over longer ones. That is, two languages $L_1$ and $L_2$ would be considered more similar if they shared a large amount of shorter strings than longer ones.

Second, the summation here is infinite, meaning that a precise closed-form solution is only possible in very special cases. However because we probability distribution over the non-negative integers is used, one may leverage an approximate metric. That is, for a fixed probability distribution $\nu$ over the non-negative integers and $\varepsilon > 0$, there exists $N$ such that:

$$\sum_{k=0}^{N} \nu(k) > 1 - \varepsilon$$

By introducing such an $\varepsilon$ error, the sum becomes finite, and can be refined to arbitrary precision supposing that probability measure $\nu$ is computable.

# 4 Linear Operators

A common representation of NFAs is in terms of graphs, which in turn are often represented by adjacency matrices, adjacency lists, pointers, or some other representation. Here we are concerned with adjacency matrices. An interesting property of adjacency matrices is that they can be seen as a function that maps between vertices on a graph.

This view is interesting for several reasons. First, matrices are ways to represent linear operators between finite-dimensional vector spaces. Second, the space of linear operators can be endowed with a norm known as the operator norm defined as follows:

$$\|T\| = \inf \{c \in \mathbb{R} \ : \ \forall x \in X, \|Tx\| \leq c \|x\|\}$$

Where $T : X \to Y$ is a linear operator between normed vector spaces $X$ and $Y$, which need not be finite-dimensional. We remark that norm vector spaces that are complete with repsect to their norm are called Banach spaces. The definition of a norm like this allows us to then define a metric: two operators are close in metric if their difference has a small norm. The goal is to now extend this to the space of NFAs. To do this, we first require a rigorous algebraic formulation of formal language and automata theory [4, 2], much of which is re-invented here.

There are several immediate problems to consider. For one, it's not immediately obvious (if possible) how to embed regular languages into an algebraic field, which is required for a vector space. Existing literature (much of what was also accidentally re-invented below) on the algebraic study of formal language theory is grounded in the language of (semi)-rings, which has much weaker properties than what one may desire from a field. Nevertheless, many properties from linear algebra can still be recovered, although often with a caveat. For instance, the lack of multiplicative inverses means that matrix (linear operator) inverses are hard to compute, or do not make much sense at first glance. Because of this, we use the term linear space to do denote spaces closed under linear operation, which may not have elements taken over a field.

For a NFA $A$, with $A = (\Sigma, Q, \Delta, S, F)$, the goal is to view $A$ acting as a linear operator between spaces. In particular, the approach we take is to see $\Delta$ as a linear operator between spaces: this is perhaps the most obvious approach, because to begin with, $\Delta$ is already the transition function.

But the questions are then: what are the appropriate linear spaces, and what are the actions of the linear operator? One initial thought is that the transition function can, in some way, be seen

as a directed acyclic graph on the states $Q$, where each edge is weighted by the letters in the alphabet $\Sigma$ that induce the transition. The representation as an adjacency matrix does appear in literature [5]. Roughly, if $M$ is the transition matrix, then $M_{i,j} \in 2^{\Sigma}$ denotes the states that will transition state $q_i$ to $q_j$.

While such a matrix representation is useful, it is not immediately obvious how such a matrix does indeed correspond to a linear operator, especially on what linear spaces. One possible interpretation is to see the linear spaces as $|Q|$-dimensional, where each dimension of the space corresponds to one member of $Q$. The objects of the space is then sets of strings over $\Sigma$.

**Definition 1** (String Space)**.** *The string space of* $\Sigma$ *is a semiring* $\left(2^{\Sigma^\star}, \cup, \cdot, \mathbf{0}, \mathbf{1}\right)$ *such that:*

*(a) The semiring addition is the set union* $\cup : 2^{\Sigma^\star} \times 2^{\Sigma^\star} \to 2^{\Sigma^\star}$.

*(b) The semiring multiplication is the string concatenation* $\cdot : 2^{\Sigma^\star} \times 2^{\Sigma^\star} \to 2^{\Sigma^\star}$ *such that:*

$$A \cdot B = \{a \cdot b \ : \ a \in A, b \in B\}$$

Here the string space is the power set of all strings generated by $\Sigma$ through monoid multiplication (string concatenation). Defining the string space like this allows us to equip it with a semiring structure by viewing addition as set union. For convenience, we may write $R$ instead of $2^{\Sigma^\star}$ to denote the set corresponding to the string space.

Observe that $R$ has the structure of a 1-dimensional linear space. The big difference, however, is that the scalar elements are elements of a semiring rather than a field. Nevertheless, $R$ is still closed under linear operations, and is therefore a linear space.

**Theorem 4.** *A string space* $R$ *is a linear space.*

*Proof.* A semiring contains a zero element and is closed under (semiring) addition. Furthermore, it is closed under (left semiring) multiplication with respect to other elements of the semiring. □

The natural extension of a 1-dimensional linear space is a $n$-dimensional linear space.

**Definition 2** ($n$-String Space)**.** *For a string space* $R$ *and* $n \in \mathbb{Z}^+$*, the* $n$*-dimensional string space* $R^n$ *is then the free semimodule isomorphic to* $n$ *copies of* $R$*.*

A natural representation of $R^n$ is as a $n$-dimensional vector, and in this case we prefer row vectors to column vectors. We abuse notation to identify elements of $R^n$ with their row vector representation. Furthermore, we demonstrate that this is indeed still a linear space, where semiring operations are defined coordinate-wise. An immediate consequence is that this also forms a linear space, since it is an $n$-dimensional linear space.

In particular, it would be nice to have a linear operator between string spaces.

**Definition 3** (Linear String Space Operator)**.** *A linear string space operator is a linear operator* $A : R^n \to R^m$.

In particular, we are interested in a matrix representation.

**Definition 4** (Matrix Representation of Linear String Space Operator)**.** *The matrix representation of a linear string space operator $A : R^n \to R^m$ is a matrix $A \in M_{n\times m}(R)$ that acts on row vectors of $R^n$ by right multiplication.*

Here each entry of the matrix denotes the sets strings that are concatenated during transition. As with the $n$-dimensional string space $R^n$, we abuse notation for $A$ to stand in for both the linear operator and its matrix representation. From linear algebra, we know that the space of linear operators between linear spaces is itself a linear space.

Observe that the elements for the matrix of $A$ are drawn from $R$ (which is just $2^{\Sigma^\star}$) rather than $2^\Sigma$, which is what we would expect for an NFA. In other words, transitions in $A$ are given by sets of potentially long strings, rather than just single alphabets. The definition provided here is intended to be slightly more general, with the NFA case of single-letter transitions being a special case. Nevertheless, given a NFA, we are now ready to describe a particular matrix representation as a linear operator between $n$-string spaces.

**Definition 5** (Matrix Representation of $\Delta$)**.** *For a NFA $(\Sigma, Q, \Delta, S, F)$ with string space $R$ generated by $\Sigma$ and $n = |Q|$, the matrix representation of $\Delta$ as a linear operator is a matrix $A \in M_{n\times n}(R)$ where:*

$$A_{i,j} = \{a \ : \ ((a, q_i), B) \in \Delta, q_j \in B\}$$

Of course, this is a linear string space operator simply because every entry of the transition matrix will be a set of singleton strings.

**Example 7.** *Consider $\Sigma = \{a, b, c\}$ and the NFA below:*

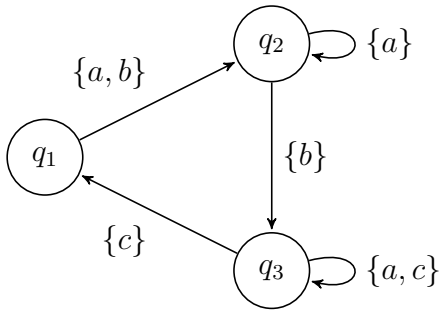



Figure 1: NFA Graph

*The most important distinction between this and a typical NFA is that we implicitly assume every state to be both starting and accepting. In other words, accepting strings is very permissive, which simplifies things for now. However, to embed this into our model there are several steps.*

*First, we have the string space $\left(2^{\Sigma^\star}, \bigcup, \cdot, \mathbf{0}, \mathbf{1}\right)$.*

*Next, for the matrix $A$ that we will construct, take $A_{i,j}$ to denote the transition from state $q_i$ to state $q_j$. The matrix is then:*

$$A = \begin{bmatrix} \mathbf{0} & \{a,b\} & \mathbf{0} \\ \mathbf{0} & \{a\} & \{b\} \\ \{c\} & \mathbf{0} & \{a,c\} \end{bmatrix}$$

*In order to perform string concatenation towards the right, transition matrices act by right-matrix multiplication. That is, if $v \in R^3$ is the initial $n$-dimensional string space, then the subsequent string space is $vA$.*

*To briefly demonstrate, two transitions of the matrix $A$ appears as follows:*

$$A^2 = \begin{bmatrix} \mathbf{0} & \{a,b\} & \mathbf{0} \\ \mathbf{0} & \{a\} & \{b\} \\ \{c\} & \mathbf{0} & \{a,c\} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \{a,b\} & \mathbf{0} \\ \mathbf{0} & \{a\} & \{b\} \\ \{c\} & \mathbf{0} & \{a,c\} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \{aa,ba\} & \{ab,bb\} \\ \{bc\} & \{aa\} & \{ab,ba,bc\} \\ \{ac,cc\} & \{ca,cb\} & \{aa,ac,ca,cc\} \end{bmatrix}$$

*In general, from graph theory, $A^k$ denotes the $k$th consecutive transition using $A$, and each entry $A^k_{i,j}$ is the set of strings that will get from state $q_i$ to $q_j$ in $k$ steps.*

## 4.1 Automata Metrics

Having established how to embed (regular) languages into linear spaces with NFAs as linear operators between these spaces, we now turn our attention to defining a metric over NFAs.

### 4.1.1 Edit Distance Metrics

Matrices are a common representation of linear operators between finite-dimensional linear spaces. Our first attempt is based on graph edit distance, which directly corresponds with matrix edit distance. We that in retrospect this was not a good idea, and experimental results were bad. Nevertheless, we proceed.

Our main objective here is to define a syntactic metric based on graph edit distance in addition to a semantic metric derived from earlier work with measure theory. First, we quickly recap the measure-based metrics.

**Definition 6** (String Space Measure)**.** *Let $(\Sigma^\star, \sigma(\Sigma^\star))$ be a measurable space defined on the strings $\Sigma^\star$. If $\lambda : \sigma(\Sigma^\star) \to \mathbb{R}^{\geq 0}$ is a measure, then we call it a string space measure, and the measure space $(\Sigma^\star, \sigma(\Sigma^\star), \lambda)$ a string measure space.*

The elements of a string measure space consists of strings finite, while the $\sigma$-algebra generated consists of sets of finite strings. A measure $\lambda : \sigma(\Sigma^\star) \to \mathbb{R}^{\geq 0}$ is also equipped, and may be defined. We have also shown previously that measures are able to define a metric.

**Definition 7** (String Space Metric)**.** *Let $(\Sigma^\star, \sigma(\Sigma^\star), \lambda)$ be a string measure space. The string metric space $\left(2^{\Sigma^\star}, \zeta\right)$ is a metric space with a metric function $\zeta : 2^{\Sigma^\star} \times 2^{\Sigma^\star} \to \mathbb{R}^{\geq 0}$ defined as:*

$$\zeta(A,B) = \lambda(A \triangle B)$$

**Theorem 5.** *The string space metric function $\zeta : 2^{\Sigma^\star} \times 2^{\Sigma^\star} \to \mathbb{R}^{\geq 0}$ is a metric function.*

*Proof.* The function induced by a measure and the symmetric set difference is a metric. □

Recall that although we write $2^{\Sigma^\star}$ here, they are synonymous with $R$. Having defined a metric on string spaces, we are interested in examining how we may syntactically compare two linear operators between string spaces. To do this, we formalize the notion of edit distance on matrices in terms of algebra.

First, for two operators $A_1, A_2 : R^n \to R^m$, it is possible that the two are isomorphic up to some permutation. Note that we want to consider the special case of $n = m$ first, for which theories of permutation matrix related to graph isomorphism are well-developed. After all, we are focused on NFAs here, which have nice an straightforward graph (matrix) representations.

**Theorem 6** (Square Operator Permutation)**.** *Let $P_n \subseteq M_{n\times n}(R)$ denote the family of $n$-dimensional permutation matrices. Two linear operators $A_1, A_2 : R^n \to R^n$ are isomorphic if there is a permutation matrix $P \in P_n$ such that:*

$$A_1 = PA_2P^T$$

*Proof.* The action of permutation matrices on the matrix representation on a graph is equivalent to relabelling the vertices, which preserves graph (and thus operator) isomorphism. □

Permutation matrices give us the ability to talk about isomorphisms up to permutation. However they lack the ability to talk about how syntactically similar two matrices might be up to permutation. To do this, for two matrices of the same dimensions, we may be interested in seeing how far the image of the matrices in each dimension differ. Without permutation matrices for two matrices $A$ and $B$, we may try something like this:

$$\sup_{1\leq i,j\leq n} \zeta\left(A_{i,j}, B_{i,j}\right)$$

In other words, we find entries of the matrix that yield the most distance with respect to the $\zeta$ string space metric function. With permutation matrices, we may try something like this:

$$\inf_{P\in P_n} \sup_{1\leq i,j\leq n} \zeta\left(A_{i,j}, B_{i,j}\right)$$

This is fairly straightforward, and more or less just amounts to saying to take the best permutation of one of the matrices with respect to the other, and then take the distance with respect to $\zeta$.

However, there are still several concerns here. First, we need to generalize this to cases where $A$ and $B$ are not the same dimension. In addition, we are not completely sure if taking the infimum of $P \in P_n$ retains a metric structure, which means that such an operation may need to be defined with respect to a fixed permutation matrix.

We first consider how square matrices of different dimensions may be compared.

**Definition 8** (Linear Operator Dimension)**.** *Let $A : R^n \to R^n$ be a square linear string space operator. For $m > n$, the dimension extension of $A$ is a square linear operator $\overline{A} : R^m \to R^m$ whose matrix representation has form:*

$$\overline{A} = \begin{bmatrix} A & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Although the newly appended dimensions of operator action effectively have trivial action, this still acts as an embedding into a higher dimension. With this, we then attempt to define a metric between two square linear matrices of different dimensions:

**Definition 9** (Non-Uniform Dimension Square Linear Operator Metric)**.** *For two square linear operators $A : R^n \to R^n$ and $B : R^m \to R^m$ where $m > n$, fix the permutation matrix $P \in P_m$, define the operator metric $Z_{p,n,m} : (R^n \to R^n) \times (R^m \to R^m) \to \mathbb{R}^{\geq 0}$ as:*

$$Z_{p,n,m}(A, B) = \sup_{1 \leq i,j \leq m} \zeta\left(\overline{A}_{i,j}, \left(PBP^T\right)_{i,j}\right)$$

In short, when given two square linear matrices $A$ and $B$ where $A$ is a smaller dimension than $B$, the goal of the metric function $Z$ is to first stretch $A$ into $\overline{A}$ to match the dimension of $B$, and then apply the supremum coordinate-wise metric on $\overline{A}$ and $B$.

**Theorem 7.** *The function $Z_{p,n,m} : (R^n \to R^n) \times (R^m \to R^m) \to \mathbb{R}^{\geq 0}$ defined above is a metric.*

*Proof.* Once the permutation matrix $P \in P_m$ is fixed, this reduces to applying the supremum over a finite set of metric functions for each entry of the matrix. □

The challenge, of course, is finding the correct permutation matrix $P$. We note that although we used the supremum here, another viable option could be general $p$-norm type summations, which also do induce a metric because they satisfy the triangle inequality.

Nevertheless, the painstaking process here allows us to algebraically formalize an edit distance metric for matrix representations of NFA.

**Example 8.** *Consider $\Sigma = \{a, b, c\}$ and two NFAs defined below:*

*The respective transition matrices are then:*

$$A = \begin{bmatrix} \mathbf{0} & \{a\} & \mathbf{0} \\ \mathbf{0} & \{b\} & \{a\} \\ \{c\} & \mathbf{0} & \mathbf{0} \end{bmatrix} \qquad \overline{A} = \begin{bmatrix} \mathbf{0} & \{a\} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \{b\} & \{a\} & \mathbf{0} \\ \{c\} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \qquad B = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \{a\} \\ \mathbf{0} & \mathbf{0} & \{a\} & \mathbf{0} \\ \{c\} & \mathbf{0} & \{b\} & \{a\} \\ \mathbf{0} & \{c\} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

*The corresponding permutation matrix is:*

$$P = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

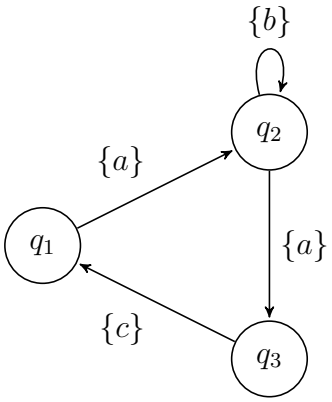



Figure 2: NFA $A$

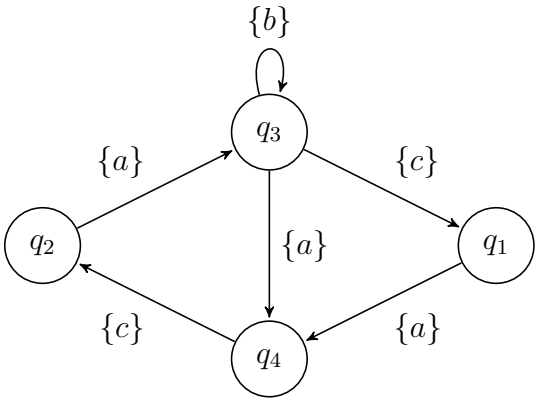



Figure 3: NFA $B$

*Indeed, we have:*

$$PBP^T = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \{a\} \\ \mathbf{0} & \mathbf{0} & \{a\} & \mathbf{0} \\ \{c\} & \mathbf{0} & \{b\} & \{a\} \\ \mathbf{0} & \{c\} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \{a\} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \{b\} & \{a\} & \{c\} \\ \{c\} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \{a\} & \mathbf{0} \end{bmatrix}$$

*If we proceed with the metric $\zeta$ calculation (assuming a counting measure) at each entry. We find the most significant differences occur at row-column index pairs $(2,4)$ and $(4,3)$:*

$$\zeta_{(2,4)}\left(\overline{A}_{2,4}, \left(PBP^T\right)_{2,4}\right) = \zeta_{(2,4)}\left(\mathbf{0}, \{c\}\right) = 1 \qquad \zeta_{(4,3)}\left(\overline{A}_{4,3}, \left(PBP^T\right)_{4,3}\right) = \zeta_{(4,3)}\left(\mathbf{0}, \{a\}\right) = 1$$

The experiments here did not work. In simulations run, there was very poor correlation between the syntactic graph edit distance metric with the semantic measure-based metric.

In part this may be because a small edit in a graph may induce a large change in the language accepted. For instance, disconnecting a single edge means that up to (countably) infinitely many words are no longer recognized by the automata that the graph represents.

This suggests that other syntactic metrics may be necessary, but nevertheless this is a preliminary step towards our goal.

# 5 Separating Automata

There are several ways to define a metric over the space of strings. For instance, the edit distance between two strings defines a metric.

Another metric that can be defined over two strings is with respect to an automata. For strings $w$ and $v$, let $\mathcal{A}$ be the smallest automata that accepts $w$ and rejects $v$. The distance can then be defined as follows:

$$d\left(w, v\right) = \frac{1}{2^{|\mathcal{A}|}}$$

Where $|A|$ here denotes the number of states of $\mathcal{A}$. Intuitively, the larger an automata is required to distinguish two strings, the closer they are.

This seems like a fairly natural metric, so the question is then how to promote this to compare over two sets of strings rather than just two strings itself. Since languages are no more than sets of strings, this upgrade will yield a metric over languages rather than over strings.

A simple way to upgrade a metric over a set to its power set (in this context from between strings to between languages) is the Hausdorff metric. In general, given any metric space $(M, d)$, the Hausdorff metric space for this space is $\left(2^M, d_H\right)$ given as:

$$d_H(X, Y) = \max\left\{\sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y)\right\}$$

In other words, between two sets $X, Y \subseteq M$, the Hausdorff distance $d_H$ between them is the distance of the most extreme pair of points $x \in X$ and $y \in Y$ with respect to the original metric $d$.

However we hope to do better. NFAs can be used to separate strings, but they can also be used to separate two sets of strings. That is, given two sets of strings $P$ and $N$, where we call $P$ the set of positive strings and $N$ the set of negative strings, let $\mathcal{A}$ be the smallest NFA that accepts all the strings in $P$ and rejects all the strings in $N$, where small refers to the number of states. can a metric be defined with respect to $\mathcal{A}$? We would like to make a statement like:

$$d(P, N) = \frac{1}{2^{|\mathcal{A}|}}$$

However it's not immediately clear that this forms a metric, or if it does at all. Nevertheless, the method of finding a minimal separating NFA is still of theoretical interest. This is type of problem is known to be $\mathbf{NP - Complete}$ [1], but having a clear embedding and decision procedure is still useful for exploring feasibility.

We achieve this by constructing a boolean satisfiability formula that encodes $P$ and $N$, and is satisfiable if and only if such $\mathcal{A}$ exists. Our embedding is inspired by techniques from [3]. There are several high-level insights that we leverage:

(1) NFAs can be represented as directed multi-edge graphs where each edge is labeled by one letter from $\Sigma$. Self-loops are permitted here. In other words, let $e_{i,j,\sigma}$ be an indicator variable encodes the indicator of a transition from state $q_i$ to state $q_j$ on the letter $\sigma$.

(2) For each $u \in P$, we can create a formula that forces a sequence of edge walks resulting in a final state in $\mathcal{A}$. Similarly for each $v \in N$ we can encode a sequence that will force a rejection of $v$ in $\mathcal{A}$.

We first construct a formula that will force $\mathcal{A}$ to accept a word $w$ if and only if the formula is satisfied. Let $y_{i,t}^w$ denote be an indicator variable to show that $\mathcal{A}$ is at state $q_i$ at time $t$, with $1 \leq i \leq n$ and $1 \leq t \leq |w| + 1$. Note that since each letter of $w$ acts as a transition, the automata

will occupy $|w|+1$ possibly repeated states during its accepting run. Then:

$$\rho_w \equiv \bigwedge_{1\le t\le |w|+1} \left[ \bigwedge_{1\le i,j\le n} \neg\left(y_{i,t}^w \wedge y_{j,t}^w\right)\right]$$

Forces the automata to be in only one state at any given time $t$ while reading $w$. To accompany this, let $e_{i,j,\sigma}$ to denote that $\mathcal{A}$ has a transition edge from $q_i$ to $q_j$ on letter $\sigma$. Similarly:

$$\pi_w \equiv \bigwedge_{1\le t\le |w|} \left[ \bigvee_{1\le i,j\le n} \left(y_{i,t}^w \wedge y_{j,t+1}^w \wedge e_{i,j,w_t}\right)\right]$$

Additionally, we can force boundary conditions to ensure that $\mathcal{A}$ begins reading $w$ on a starting state and ends on an accepting state:

$$\gamma_w \equiv \left[ \bigvee_{1\le i,j\le n} \left(e_{i,j,w_1} \wedge s_i\right)\right] \vee \left[ \bigvee_{1\le i,j\le n} \left(e_{i,j,w_{|w|}} \wedge f_j\right)\right]$$

Where $s_i$ and $f_j$ are indicator variables to express that $q_i$ is a starting state and $q_j$ is a final state respectively. Then finally we set:

$$\varphi_w \equiv \rho_w \wedge \pi_w \wedge \gamma_w$$

Then $\mathcal{A}$ will only accept $w$ if and only if $\varphi_w$ is satisfiable. Finally:

$$\Phi_{P,N} \equiv \left[\bigwedge_{u\in P} \varphi_u\right] \wedge \left[\bigwedge_{v\in N} \neg\varphi_v\right]$$

By construction, $\Phi_{P,N}$ is true if and only if $\mathcal{A}$ accepts all $P$ and rejects all $N$.

Suppose that $\Sigma$ is known. If $\Phi_{P,N}$ is satisfiable, then $\mathcal{A} = (\Sigma, Q, \Delta, S, F)$ can be extracted as:

$$\begin{aligned}
Q &= \{q_1, \ldots, q_n\} \\
\Delta &= \{((\sigma, q_i), \{q_k \ : \ e_{i,k,\sigma} = \top\}) \ : \ \exists j, \ e_{i,j,\sigma} = \top\} \\
S &= \{q_i \ : \ s_i = \top\} \\
F &= \{q_j \ : \ f_j = \top\}
\end{aligned}$$

In short, an automata is recovered based on the edge indicators $e_{i,j,\sigma}$, each of which denotes one transition. The formula for $\Delta$ is bulky as a consequence of attempting to recover an NFA.

**Theorem 8.** *$\varphi_w$ is satisfiable if and only if $\mathcal{A} = (\Sigma, Q, \Delta, S, F)$ accepts $w$.*

*Proof.* Consider a satisfiable configuration of $\varphi_w$, which implies that $\rho_w$, $\pi_w$, and $\gamma_w$ are all satisfied. From $\rho_w$ there exists exactly one sequence of $\left(y_{i,t}^w\right)$ indicator variables that are each $\top$:

$$y_{a,1}^w, y_{b,2}^w, \ldots, y_{z,|w|+1}^w$$

Where $1 \leq a, b, \ldots, z \leq n$ denote arbitrary state numbers. Furthermore, $\pi_w$ being $\top$ means that for each consecutive $y^w_{i,t}$ and $y^w_{j,t+1}$ the edge indicator variable $e_{i,j,w_t}$ is $\top$. In essence, this together $\rho_w$ and $\pi_w$ then forces the existence of a path of transitions induced by $w$. Additionally, the formula $\gamma_w$ forces $w$ to begin on an initial state and end on a final state. Thus, there is a path that makes $\mathcal{A}$ accept $w$.

For the converse, suppose that $\mathcal{A}$ accepts $w$. This implies that in the graph representation of $\mathcal{A}$ there exists a path on which $w$ is accepted. Consider the indicator variable $e_{i,j,w_t}$ corresponding to each transition of $w$. In $\pi_w$ this means that on each such transition we can set $y^w_{i,t} = \top$ and $y^w_{i,t+1} = \top$. This satisfies $\pi_w$, and also satisfies $\rho_w$ since we can just set all other $y$ variables of $w$ to $\bot$. Furthermore, since $\mathcal{A}$ accepting $w$ implies that $w$ must begin on a starting state and end on a final state, this is automatic for $\gamma_w$. Hence, $\varphi_w$ is satisfiable.

□

The consequence then extends to $\Phi_{P,N}$, which is a conjunction over all $\varphi_w$ and $\neg\varphi_w$ sub-formula.

**Theorem 9.** *$\Phi_{P,N}$ is satisfiable if and only if $\mathcal{A} = (\Sigma, Q, \Delta, S, F)$ accepts every string in $P$ and rejects every string in $N$.*

*Proof.* For each $w \in P \cup N$, this is a consequence of $\mathcal{A}$ accepting $w$ if and only if $\varphi_w$ is satisfiable. □

Furthermore, this NFA will be of size $n$ if $\Phi_{P,N}$ is satisfiable.

**Theorem 10.** *If $\Phi_{P,N}$ is satisfiable, then there exists an NFA of size $n$ that separates $P$ and $N$.*

*Proof.* The satisfiability of $\Phi_{P,N}$ implies the existence of a graph defined over at most $n$ vertices, each of which would correspond to a state of an NFA that separates $P$ and $N$. □

# 6 Bi-Directional Merging

Although it is not a metric, we introduce the concept of a merging one set with another. To begin with, for a string $w \in \Sigma^\star$ and a set $A \subseteq \Sigma^\star$. The merging cost of $x$ into $A$ can be defined as:

$$m_w(x, A) = \inf_{a \in A} d(x, a)$$

Where $d$ is a metric over strings. In particular here we consider the edit distance. The cost of merging one set $A$ into another set $B$ can then be extended as follows with overloaded notation:

$$m_W(A, B) = \sum_{a \in A} \inf_{b \in B} d(a, b)$$

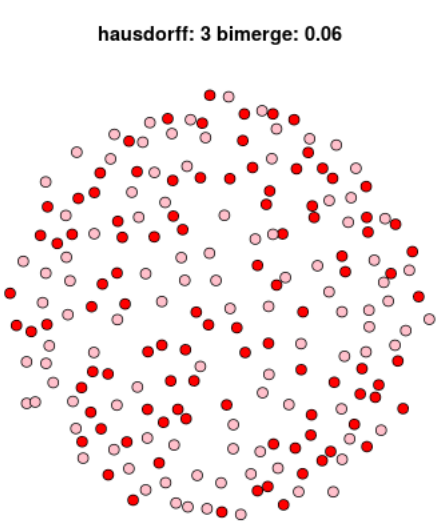



Figure 4: $\Sigma_1 = \Sigma_2 = \{A, B, C, D\}$

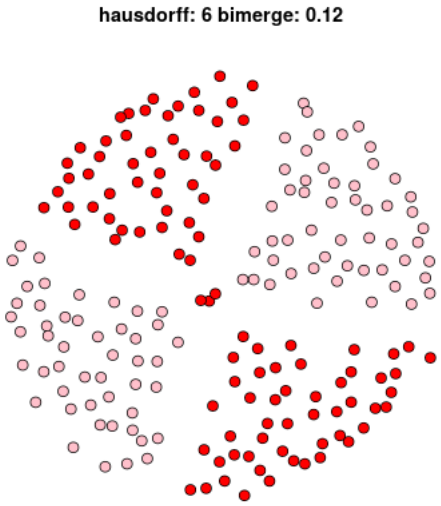



Figure 5: $\Sigma_1 = \{A, B, C, D\}$ and $\Sigma_2 = \{D, E, F, G\}$

This is just a summation of the notion of a merge cost defined earlier. However, it is only fair that we consider the cost of merging $A$ into $B$ and also $B$ into $A$, for which we have:

$$m(A, B) = \sum_{a \in A} \inf_{b \in B} d(a, b) + \sum_{b \in B} \inf_{a \in A} d(a, b)$$

So even though $m$ defined like this is not a metric function nevertheless we are interested in seeing how it might empircally perform against established metrics such as the Hausdorff metric. That is, we wanted to see how well measurements such as $m$ and established metrics such as the Hausdorff metric correlate with our intuitive understanding of string similarity (which we argue to be by edit distance).

A numerical experiment was run where 100 strings ($A$) of length 6 was uniformly generated at random from alphabet $\Sigma_1$, and another 100 strings ($B$) of length (6) was uniformly generated at random from alphabet $\Sigma_2$.

In one round of the experiment as shown in Figure 4, we set $\Sigma_1 = \Sigma_2 = \{A, B, C, D\}$. In other words, the set $A$ (dark red) and $B$ (light red) were generated from the same distribution. The figure displayed is a force plot on a complete graph (in this case $K_{200}$) where the edge weights are weighted by inverse edit distance. That is, each dot (vertex) represents a string, and dots that are visually closer to each other will tend to have higher weights (and thus higher attractive force) due to a smaller edit distance.

The idea behind this type of plot is to see how well a metric is able to correlate to our understanding of string similarity with respect to the edit distance. In the first round, because $A$ and $B$ were generated from the same probability distribution, the force plot has a difficult time separating them, and therefore yields a strong mixing. The Hausdorff metric is 3, meaning that all strings in $A$ and $B$ are at most edit distance 3 apart. The bi-merge cost $m$ is 0.06.

In the second round the experimental set up was the same and the results are displayed in Figure 5, except now we have $\Sigma_1 = \{A, B, C, D\}$ and $\Sigma_2 = \{D, E, F, G\}$. Again 100 strings of length 6 were generated uniformly at random from $\Sigma_1$ for $A$ and 100 strings of length 6 were generated uniformly at random from $\Sigma_2$ for $B$.

We note that this time the Hausdorff distance between $A$ and $B$ is 6, which is expected because the alphabets from which they were uniformly generated only shares $D$ in common. Likewise, the bi-merge cost is higher, now at 0.12. Finally, because $A$ and $B$ are generated from fairly different alphabets (which only share one letter), the corresponding languages have strong visual separation, which is desired.

In summary, although the bi-merge cost is not a metric, the techniques employed here for visualizing the distance of two strings appears to be useful, can be utilized for similar problems.

# 7 Conclusion and Future Work

In this work we present our efforts towards embedding regular languages into metric spaces. Our goal here is to formally define and explore how different metrics over regular languages work in both theory and practice. To that end, we approach this question through the perpective of measure theory, algebraic formulation, and boolean satisfiability. Additionally, we present visualization techniques that for strings in terms of graphs that may be useful for future related endeavors.

## 7.1 Graph Kernels

We emphasize that there is still much to be done in the study of regular languages and metric spaces. One particular approach that we aim to tackle in the near future is through the view of graph kernels [6]. Roughly speaking, graph kernels provide a way to embed graphs into Hilbert spaces, on which inner products may be done. We believe that this is an exciting direction with a lot of potential.

# References


[1] E Mark Gold. "Complexity of automaton identification from given data". In: *Information and control* 37.3 (1978), pp. 302–320.

[2] Dexter Kozen. "A completeness theorem for Kleene algebras and the algebra of regular events". In: *Infor. and Comput.* 110.2 (1994), pp. 366–390.

[3] Daniel Neider and Ivan Gavran. "Learning Linear Temporal Properties". In: *CoRR* abs/1806.03953 (2018). arXiv: 1806.03953. URL: http://arxiv.org/abs/1806.03953.

[4] Jean-Éric Pin. "Mathematical foundations of automata theory". In: *Lecture notes LIAFA, Université Paris* 7 (2010).

[5] John E Savage. *Models of computation*. Vol. 136. Addison-Wesley Reading, MA, 1998.

[6] S Vichy N Vishwanathan et al. "Graph kernels". In: *Journal of Machine Learning Research* 11.Apr (2010), pp. 1201–1242.

# A Bi-Merge Code

```
library("stringi");
library("igraph");

# Generate random strings
rand.strs <- function (num, len) {
  return (stri_rand_strings(num, len, "[A-D]"));
}

# Cost of merging a single string into a set
single.merge.cost <- function (key, base) {
  costs <- lapply(base, function (b) { return (adist(key, b)); });
  return (min(unlist(costs)));
}

# Cost of merging entire set into another set
whole.merge.cost <- function (keys, base) {
  costs <- lapply(keys, function (k) { return (single.merge.cost(k, base)); });
  return (max(unlist(costs)));
}

# Hausdorff distance
haus.dist <- function (set1, set2) {
  return (max(c(unlist(whole.merge.cost(set1, set2))),
                unlist(whole.merge.cost(set2, set1))));
}

bi.merge.cost <- function (set1, set2) {
  return ((sum(unlist(whole.merge.cost(set1, set2))) / length(set1)) +
          (sum(unlist(whole.merge.cost(set2, set1))) / length(set2)));
}

# Make the graph(s)

strs1 <- stri_rand_strings(100, 6, "[A-D]");
strs2 <- stri_rand_strings(100, 6, "[D-G]");
all.strs <- c(strs1, strs2);

strs1.len <- length(strs1);
strs2.len <- length(strs2);
all.len <- strs1.len + strs2.len;

all.ids <- c(1:all.len);

node.data <- data.frame(
  node.id = all.ids,
  node.str = all.strs,
  node.set = c(rep(c(1), times=strs1.len), rep(c(2), times=strs2.len))
)

edge.data <- data.frame(
  edge.from =
    unlist(lapply(all.ids, function (x) {
      return (rep(c(x), times=all.len));
    })),
  edge.to = rep(c(1:all.len), times=all.len),
  weight =
    unlist(lapply(all.ids, function (x) {
      return (unlist(lapply(all.ids, function (y) {
        return (1 / (1 + (adist(all.strs[x], all.strs[y]))));
    }))})))
)

net <- graph_from_data_frame(d=edge.data, vertices=node.data);

V(net)$color <- c("red", "pink")[V(net)$node.set];
V(net)$size <- 5;
V(net)$label <- NA;

E(net)$color <- "white";

haus <- haus.dist(strs1, strs2);
bimg <- bi.merge.cost(strs1, strs2);

title = paste(c("hausdorff:", haus, "bimerge:", bimg), collapse=" ");

l <- layout_with_kk(net);


png('AD6-DG6-100-IM.png');
plot(net, layout=l, main=title);
dev.off();
```