Instituto Superior Técnico

Identification of Regular Languages with Computable Scientists

André Cheng
andrecheng@tecnico.ulisboa.pt

Advisor: José Félix Costa
July, 2023

1 Introduction

An application that simulates a *scientist* capable of identifying *regular languages* will be implemented. It will run through an enumeration of *deterministic finite automata* given by Reis in [11]. Thus, the concept of *deterministic finite automaton*, DFA, will be defined as well as other definitions related to DFAs. At last, an introduction is made to the Learning Theory focusing on the main concepts that are related to our scientist.

To have a general idea of the Learning Theory the reader can refer to [3]. Previous work related to topics presented in this paper can be consulted in [4], [8], and [14]. In [4] only positive data was provided to learn regular languages and in [14] a more efficient algorithm to identify DFAs was implemented.

Compared to the work done by Gabriel Gamatos in [8], we only have presented the main concepts and results that were related to the application implemented. As such, this paper has a narrower range of definitions and results; however, we have added proofs to all statements.

2 Definitions

To work with regular languages, we will first define Deterministic Finite Automata (DFA) and state which types of DFAs we will be working with.

Definition 2.1 (Deterministic Finite Automaton [13]):
A DFA is a 5-tuple consisting of (a) a finite set Σ named ***alphabet****, (b) a non-empty finite set Q of* ***states*** *, (c) an* ***initial state*** *$q_0 \in Q$, (d) a set of* ***final states*** *$F \subseteq Q$ and (e) a* ***transition function*** *δ mapping $Q \times \Sigma$ to Q.*

In the following examples and in the application, all the DFAs will have an alphabet $\Sigma = \{0, 1\}$, i.e., we will be considering only binary words.

A DFA is an abstract machine, which given a word $w \in \Sigma^\star$ will start a computation from its initial state. The DFA will then read the word w symbol by symbol while performing transitions between states according to the transition function and each symbol read from w. Finally, the computation will halt when there are no more symbols to be read.

The empty word is denoted by ε and for all $v, w \in \Sigma^\star$, vw denotes the result of concatenating w onto the end of v.

Definition 2.2 (Extented Transition Function):
Given a DFA $D = (\Sigma, Q, q_0, F, \delta)$, the extension of δ, $\delta^* : Q \times \Sigma^\star \to Q$ is such that:

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon \\ \delta^*(\delta(q, a), v) & \text{if } w = av \end{cases}$$

Even though we have just introduced a new notation, we may just use δ in place of δ^*, as it is clear when we are using words or symbols.

Definition 2.3 (Language Recognized by a DFA):
Given a DFA $D = (\Sigma, Q, q_0, F, \delta)$, a word $w \in \Sigma^\star$ is said to be ***accepted*** *by D, if $\delta(q_0, w) \in F$. The* ***language recognized*** *by D is $L(D) = \{w \in \Sigma^\star : \delta(q_0, w) \in F\}$.*

Definition 2.4 (Regular Language):
A language that is recognizable by DFAs is said to be a ***Regular Language****. The set of all regular languages is denoted by $\mathcal{REG}$.*

There might be DFAs with states that are inaccessible from the initial state, meaning that the computation starting from the initial state on any word will not halt on those states, for instance, take D_1 in Figure 1:

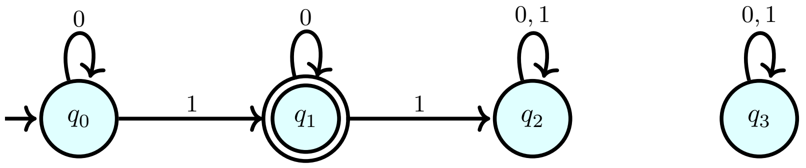


Figure 1: Example of a not initially connected DFA D_1

As the reader might notice, all the states inaccessible from the initial state add no information to $L(D_1)$, so we might as well just remove them. We do this because the enumeration by Reis in [11], which will be used later in the specification of our Scientist, enumerates only *Initially Connected* DFAs.

Definition 2.5 (Initially Connected DFA):
*Given a DFA $D = (\Sigma, Q, q_0, F, \delta)$, D is **initially connected** if and only if for all $q \in Q$, there is a word $w \in \Sigma^\star$, such that $\delta(q_0, w) = q$.*

As such, we will only consider Initially Connected DFAs from now on. We may now turn our attention to these two DFAs Figure 2:

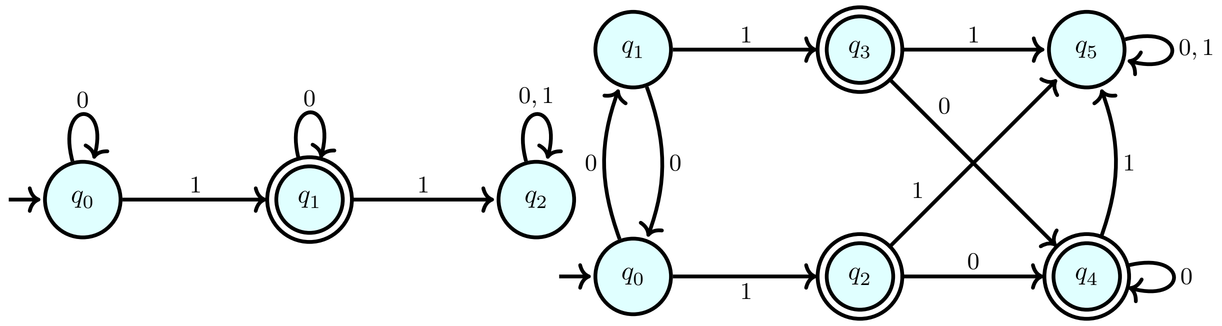


Figure 2: 2 DFAs accepting the same language, D_2 (left) being *minimal* and D_3 (right) not

Analyzing both of them, we can see that they recognize the same language, $L(D_2) = L(D_3)$. A word to be accepted by any of them has to have at least one 1, but with just one more 1, they make a transition to a non-final state from which they cannot get out of. As such, we conclude that both recognize the language $L = \{w \in \{0, 1\}^* :$ w has one and only one $1\}$.

Definition 2.6 (Minimal DFA):
*Given a DFA $D = (\Sigma, Q, q_0, F, \delta)$, D is said to be **minimal** if it is a DFA with fewer states that recognizes $L(D)$.*

The example in Figure 2, D_2 is a minimal DFA, meaning D_2 is the smallest DFA that recognizes $L(D_2)$, however, D_3 is not. Even though there exists DFAs that are not minimal, our Scientist gains no value from them, since the enumeration by Reis in [11] enumerates DFAs in ascending order in the number of states. Indeed, a minimal DFA would always be enumerated and processed first, and as such there is no need to process another DFA with more states, which recognizes the same language.

Thus, we need to know whether a DFA is minimal or not.

Definition 2.7 (Equivalent and Distinguishable states):
*Given a DFA $D = (\Sigma, Q, q_0, F, \delta)$, we say that two states $p, q \in Q$ are **equivalent**, we write $p \equiv q$, if and only if for all $w \in \Sigma^\star$ we have $\delta(p, w) \in F$ if and only if $\delta(q, w) \in F$. Otherwise, they are said to be **distinguishable**.*

Note that the relation defined above is indeed an equivalence relation, so in the future, we might say two states belong to the same class of equivalence, and in that case, we are referring to this relation.

At this point, one might expect that a DFA is minimal if and only if there are no equivalent states and that's actually the case. Since to prove such a result we will need other definitions and results, we will first specify an Algorithm 1 to find equivalent and distinguishable states given any DFA.

Lemma 2.1:
Given a DFA $D = (\Sigma, Q, q_0, F, \delta)$ and $p, q \in Q$. Then p is distinguishable from q if and only if the pair (p, q) is marked by the previous algorithm.

Proof:
(Necessary condition): Assuming that p, q are distinguishable, then exists $x \in \Sigma^\star$ that distinguishes them.

By induction on the size of x:

Algorithm 1: DFA minimization algorithm, p.70 in [7]

```
Input: DFA
Output: Bool
for $p$ in $F$ and $q$ in $Q - F$ do
    mark $(p, q)$;
end
for $(p, q)$ in $(F \times F) \cup ((Q - F) \times (Q - F))$ and $p \neq q$ do
    if for some symbol $a$, $(\delta(p, a), \delta(q, a))$ is marked then
        put $(p, q)$ on the list to_mark;
        while to_mark is not empty do
            mark the first pair;
            add all pairs from its list to to_mark;
            remove the pair;
        end
    end
    else
        for all symbols $a$ do
            if $\delta(p, a) \neq \delta(q, a)$ then
                add $(p, q)$ to the list for $(\delta(p, a), \delta(q, a))$;
            end
        end
    end
end
if all $(p, q)$ are marked, $p \neq q$ then return True;
return False
```

Base case: If $x = \varepsilon$, this means that one of them is a final state and the other is not and as such the pair (p, q) is marked at step (2).

Induction step: By hypothesis if $w \in \Sigma^\star$ such that $|w| < i$, $i \geq 1$ distinguishes two states then the pair is marked. Now, let $x \in \Sigma^\star$ be such that $|x| = i$ and $x = ay$, $t = \delta(p, a)$ and $u = \delta(q, a)$, since p, q are distinguishable so are t, u with y distinguishing them. By hypothesis the pair (t, u) will be marked, since $|y| = i - 1$.

We are not left with 2 cases, at the time when (t, u) is marked:

1. (p, q) was considered before. When that happened (p, q) either was already marked, nothing to prove here, or it was added to the list of (t, u) at (16) and so the pair will be marked at $(7 - 11)$.

2. (p, q) has yet to be considered. In this case, the if clause at (5) is true for the symbol a, and as such the pair will be marked at $(7 - 11)$.

(Sufficient condition): Again by induction, but on the number of marked pairs:

Base case: We have marked 1 pair of states, then it could've only happened at (1) and so they are distinguishable.

Induction step: Assuming that the n pairs that were marked before are distinguishable, we show the next pair states to be marked are also distinguishable. For a new pair of states to be marked, it has to happen at (2) or $(7 - 11)$:

2. One of them is final and the other is not, which means they are distinguishable

7-11. For this to happen, the if clause at (3) is true for some symbol a, meaning $(t, u) = (\delta(p, a), \delta(q, a))$ is marked which by hypothesis means that they are distinguishable by some word y, but then $x = ay$ distinguishes (p, q). □

An equivalence relation's index is its number of equivalence classes. An arbitrary language L induces a natural equivalence relation R_L; namely for all $x, y \in \Sigma^\star$, xR_Ly if and only if for all $w \in \Sigma^\star$, $xw \in L$ if and only if $yw \in L$. We say an equivalence relation R is right invariant, with respect to concatenation, if for all $x, y \in \Sigma^\star$ such that xRy then for all $w \in \Sigma^\star$ $xwRyw$.

We are now left to show that a DFA is minimal if and only if all of its states are distinguishable, we do that with the help of the next theorem:

Theorem 2.1 (The Myhill-Nerode Theorem p.65 in [7]):
The following three statements are equivalent:

1. *$L \subseteq \Sigma^\star$ is accepted by some DFA;*
2. *L is the union of some of the equivalence classes of a right invariant equivalence relation of finite index;*
3. *The equivalence relation R_L; defined as follows: for all $x, y \in \Sigma^\star$, xR_Ly if and only if for all $w \in \Sigma^\star$, $xw \in L$ if and only if $yw \in L$; is of finite index;*

Also any relation in (2) *is a refinement of R_L and R_L is right invariant.*

Proof:
$(1 \Rightarrow 2)$: Let $D = (\Sigma, Q, q_0, F, \delta)$ be the DFA which accepts L and R_M the equivalence relation xR_My if and only if $\delta(q_0, x) = \delta(q_0, y)$. R_M is right invariant and its index is finite. Also, L is the union of the classes which have words x such that $\delta(q_0, x) \in F$.

$(2 \Rightarrow 3)$: Let $x, y \in \Sigma^\star$ such that xEy. E being right invariant we have $xz \in L$ if and only if $yz \in L$, for all $z \in \Sigma^\star$ and so xR_Ly. We conclude the index of R_L is not greater than the index of E and E is a refinement of R_L.

$(3 \Rightarrow 1)$: First, R_L is right invariant, meaning if xR_Ly then xwR_Lyw, for all $w \in \Sigma^\star$, i.e, for all $z \in \Sigma^\star, xwz \in L$ if and only if $ywz \in L$. We have, by definition of R_L, for all $v \in \Sigma^\star, xv \in L$ if and only if $yv \in L$, so we just let $v = wz$.
Finally, we have to define the DFA: The set of states, Q' will be the set of equivalence classes of R_L. Let $[x] \in Q'$, we define $\delta'([x], a) = [xa]$ which is consistent, since R_L is right invariant. Let $q'_0 = [\varepsilon]$ and $F' = \{[x] : x \in L\}$. This DFA accepts L, since $\delta'(q'_0, x) = [x]$ and thus $x \in L$ if and only if $[x] \in F'$. □

With the previous theorem in mind, we can finally show what we wanted:

Theorem 2.2:
Given a DFA $D = (\Sigma, Q, q_0, F, \delta)$.
D is minimal if and only if all $p, q \in Q$ are distinguishable.

Proof:
(Necessary condition): If there existed a pair of states that were not distinguishable in a minimal DFA, those 2 states belonged in the same class of equivalence, which means this equivalence relation would have fewer classes than R_L.

(Sufficient condition): If all states are distinguishable, then each state would be its class of equivalence, we will show that the index of this relation is the same as R_L.
Because if that was not the case, then there are $p, q \in Q$ such that $p \not\equiv q$. Let $x, y \in \Sigma^\star$ be such that $\delta(q_0, x) = p$ and $\delta(q_0, y) = q$. By hypothesis, xR_Ly, but then we would have $p \equiv q$, otherwise, there would be a $w \in \Sigma^\star$ which distinguishes them, but R_L is right invariant.
□

3 DFA Enumeration

The enumeration we are using enumerates DFAs without the information about its final states, leaving us to add the final states to the DFA. This is exactly what we want, considering that for each structure of a DFA with n states, there are 2^n possible ways to choose the final states, therefore processing each one of those DFAs is not efficient.
Furthermore, the enumeration will be enumerating DFAs with a new representation, *standard representation.* To define it, we will first order the alphabet and set of states. The alphabet is rather easy to order, consequently, we let $\Pi : \Sigma \to \{0, \cdots, k-1\}$ to be a bijection.
The ordering of states is done according to the function φ that is given by Algorithm 2.

Algorithm 2: State ordering algorithm, p.40 of [11]

```
Input: DFA $D$
Output: $\varphi$
1  $\varphi(q_0) = 0$;
2  $i = 0$; // index given to the next found state
3  $s = 0$; // current state that is being processed
4  do
5      for $j \in 0..k-1$ do
           // State Processing
6          if $\delta(\varphi^{-1}(s), \Pi^{-1}(j)) \notin \varphi^{-1}(0..i)$ then
7              $\varphi(\delta(\varphi^{-1}(s), \Pi^{-1}(j))) = i + 1$;
8              $i$ += 1;
9          end
10     end
11     $s$ += 1;
12 while $s \leq i$;
```

Lemma 3.1:
The function φ defined by the Algorithm 2 is bijective.

Proof:
φ is injective, as we are assigning a new value for each state found, at 7) and 8). To show that it is bijective, let $q \in Q$, since the DFA is initially connected, there are a sequence of states $\{q'_i\}_{i\in\{0,\ldots,kn-1\}}$ and symbols $\{\sigma_i\}_{i\in\{0,\ldots,j-1\}}$, with $j < n$, such that:

$$q'_0 = q_0 \text{ and for all } m \in \{0, \ldots, j-1\}\ \delta(q'_m, \sigma_m) = q'_{m+1} \text{ and } q'_j = q$$

Since $\varphi(q'_0) = 0$ and for $m \in \{0, \ldots, j-1\}$, if $q'_m \in \varphi^{-1}(\{0, \ldots, n-1\})$ then $q'_{m+1} \in \varphi^{-1}(\{0, \ldots, n-1\})$, by hypothesis and step (7). At last, we get $\varphi^{-1}(\{0, \ldots, n-1\}) = Q$. □

Definition 3.1 (Standart Representation):
Given a DFA $D = (\Sigma, Q, q_0, F, \delta)$, its Standart Representation $(q_i)_{i\in\{0,\ldots,kn-1\}}$ is as follows:

$$q_i = \varphi(\delta(\varphi^{-1}(\lfloor i/k \rfloor),\ \Pi^{-1}(i \mod k)))$$

A DFA will then be represented by a list of integers of size $|Q||\Sigma| = nk$. For example, $[0, 1, 1, 2, 2, 2]$ would be the following DFA:

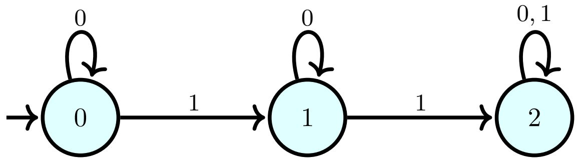


Figure 3: DFA represented by $[0, 1, 1, 2, 2, 2]$

Here, the states go from 0 to $n-1$, 0 being the initial state and the transition function is represented by the list, as follows: the first k integers are the states on which the computation will lend on if, at the first state, we read each of the symbols in the alphabet, in order.

In the last example in Figure 3, if on the initial state, we read the symbol 0 the computation will lend on the same state, but if read 1, it makes a transition to the state numbered 1. The same applies to the rest of its states.

Given the nature of φ we can expect a few things from this new representation, first at step (6) and (7), when we are assigning a new value to a new state, the variable `i` must have had gone through all the values before getting to the current one, meaning that all the states with values smaller than the current, must have been found earlier and so they will appear first in the representation (R1).

Second, at step (6) we are processing states that have been found already, otherwise, they wouldn't have a value and so we can expect in the representation, a state will appear before the part of its processing (R2).

Lemma 3.2:
Let $(q_i)_{i\in\{0,\ldots,kn-1\}}$ be the representation of a DFA with n states, in the normal form, and the alphabet with k symbols, then:

$$\textit{there exists } j \in \{0,\ldots,kn-1\} \textit{ such that } q_j = n-1 \tag{R0}$$

$$\begin{gathered}\textit{for all } m \in \{2,\ldots,n-1\} \textit{ and } i \in \{0,\ldots,kn-1\}\\ \textit{if } q_i = m \textit{ then there exists } j \in \{0,\ldots,i-1\} \textit{ such that } q_j = m-1\end{gathered} \tag{R1}$$

$$\textit{for all } m \in \{1,\ldots,n-1\} \textit{ there exists } j \in \{0,\ldots,km-1\} \textit{ such that } q_j = m \tag{R2}$$

Proof:
R0 is a result of R2. R1 is a consequence of φ, because as we have explained earlier, to assign a new value to a newly found state, `i` must have been incremented to the current state, and for that to happen all the previous states must have been found. Suppose, now, that R2 does not verify, then there is a state $q \in Q$, such that $\varphi(q) = m$, but m is not in the first km symbols of its representation. Though, this is impossible, as it would mean that q is not accessible from the initial state. □

Lemma 3.3:
A sequence $(q_i)_{i\in\{0,\ldots,kn-1\}}$ with $q_i \in \{0,\ldots,n-1\}$ which satisfies R1 and R2, defined previously, represents a DFA with n states and an alphabet of k symbols.

Proof:
Let $Q = \{q_i : i \in \{0,\ldots,kn-1\}\}$, by R2, $(n-1) \in Q$, plus R1, we have $Q = \{0,\ldots,n-1\}$. Defining a DFA $D = (\{0,\ldots,n-1\},\{0,\ldots,k-1\},\delta,0)$ with $\delta(r,\sigma) = q_{kr+\sigma}$, we are left to show that it is initially connected. By R2, there is $j < mk$ such that $q_j = m$. This means that $\delta(\lfloor j/k \rfloor, j \bmod k) = m$ and, so, we just have to repeat the process until we get to the initial state, $j = 0$. □

With the previous few lemmas, we end up with a bijection between the set of all sequences $(q_i)_{i\in\{0,\ldots,kn-1\}}$, with $q_i \in \{0,\ldots,n-1\}$, which satisfy R1 and R2 and the set of all DFAs with n states and alphabet with k symbols.

Although we are ready to enumerate all DFAs, we will use another set of conditions, which are equivalent to the previous one. Before proceeding to those conditions, we define, for a standard representation of a DFA, the sequence $(f_i)_{i\in\{0,\ldots,n-1\}}$ named the flags, which correspond to the first index in which the state i appears.

Finally, the conditions G1 and G2, to be listed below, are equivalent to conditions (R0, R1) and R2, respectively.

$$\text{for all } j \in \{2,\ldots,n-1\}\ f_j > f_{j-1} \tag{G1}$$

$$\text{for all } m \in \{1,\ldots,n-1\}\ f_m < km \tag{G2}$$

After obtaining a sequence of flags, all remaining symbols in the sequence of a DFA should be generated according to the following conditions:

$$i < f_1 \Rightarrow q_i = 0 \tag{G3}$$

$$\text{for all } j \in \{1,\ldots,n-2\}\ f_j < i < f_{j+1} \Rightarrow q_i \in \{0,\ldots,j\}) \tag{G4}$$

$$i > f_{n-1} \Rightarrow q_i \in \{0,\ldots,n-1\} \tag{G5}$$

In the end, we have the algorithms which enumerate all DFAs, starting with the Algorithm 3 which returns the next sequence of flags which fulfill conditions G1 and G2, given a correct sequence of flags.

Given a correct sequence of flags and the previously generated DFA, the Algorithm 4 generates the next one according to conditions G3, G4 and G5.

In the thesis by Reis in [11], there is an algorithm that generates all DFAs of size n and with an alphabet of size k; however, we won't be using that one, instead by starting with the first sequence of flags, $f_i = ki - 1$ for $i \in \{1,\ldots,n-1\}$, we can generate the first DFA by filling all the remaining places with 0s and enumerate the following ones as we need.

4 Learning Theory

Moving on to the Learning Theory, we want to specify a Scientist that is capable of identifying regular languages, given limited information about that language. Hence we will be defining those concepts among others, which together are said to be a *paradigm*. The following definitions can be found in [5], [9], and [10].

Algorithm 3: NextFlags, p.55 in [11]

```
Input: list flag, int $n$, int $k$
if flag is empty then
    for $i$ in $0..n-1$ do
        flag[$i$] = $ik - 1$;
    end
end
else if $flag[n-1] = n-2$ then
    empty($flag$);
    return;
end
$j = 1$;
for $i$ in $n-1..2$ do
    if $flag[i-1] \neq flag[i] - 1$ then
        $j = i$;
        break;
    end
end
$flag[j]$ -= 1;
for $i$ in $j+1..n-1$ do
    $flag[i] = ki - 1$;
end
```

4.1 A theoretically possible reality

This tells us about the set from which the language we are trying to identify was chosen from, in other words, this is the set of all possible conjectures. In our case, it would be the class of all regular languages.

4.2 Intelligle hypothesis

Most of the time we are trying to identify an infinite language, thus it won't be possible to conjecture a language by returning the whole language. Since we will be considering recursively enumerable languages, we may identify each conjecture by the Turing Machines that enumerate the language, or rather, given an enumeration of computable functions, we associate with each language the indexes of such Turing Machines. When we are referring to an index of a language L, it means the index of a Turing Machine that enumerates L.

There are different Turing Machines that enumerate the same language, though, in this paper, we are only interested in one of them since we want a *scientist* to *converge* to a single index of a language.

Thus, we need an enumeration of recursively enumerable languages, for that, we first index the set of computable functions.

Let $\mathcal{F}_n$ be the set of functions from $\mathbb{N}^n$ to $\mathbb{N}$ and $\mathcal{C}_n$ the set of computable functions from $\mathbb{N}^n$ to $\mathbb{N}$:

Definition 4.1 (Universal function):
A function $u \in \mathcal{F}_2$ is said to be universal for $\mathcal{A} \subseteq \mathcal{F}_1$ if $\mathcal{A} = \{u_p : p \in \mathbb{N}\}$.

Definition 4.2 (s-m-n property):
A function $u \in \mathcal{F}_2$ is said to have the s-m-n property if for all computable functions $v \in \mathcal{C}_2$, there is a computable map s such that $v_p = \phi^u_{s(p)}$.

A computable universal function with s-m-n property is said to be a proper universal function and there exist proper universal functions for $\mathcal{C}_1$, take Algorithm 5. Let γ be a computable injective enumeration of all words, such enumeration exists since it is decidable and infinite.

The function *univ* tries to parse the first argument, p, into a program, and if it succeeds checks the result of that program with input n expecting a natural number. If everything goes fine, simply outputs the result, else it will either raise an exception or loop indefinitely.

As a result, fix the enumeration of $\mathcal{C}_1$ to be given by $\phi_p := \phi_p^{univ} = univ(p, x)$. We are left to enumerate the recursively enumerable languages, let W_i be the set of the values enumerated by the Turing Machine of index i.

To get a better understanding of the previous definitions and results, please refer to [12].

Algorithm 4: nextDfa, p.56 in [11]

```
Input: list $delta$, list $flag$, int $n$, int $k$
$j = 0$;
$l = 1$;
if $delta$ is not empty then
    $i = n - 1$;
    for $j$ in $]kn, ..., 0]$ do
        if $i \neq 0$ and $flag[i] == j$ then
            $i$ -= 1;
        end
        else if $delta[j] < i$ then
            break;
        end
    end
    if $j == 0$ and ($flag[1] == 1$ or $delta[0] == 1$) then
        empty($delta$);
        return;
    end
    else
        $delta[j]$ += 1;
        $j$ += 1;
        $l = i + 1$;
    end
end
else
    $delta = [0$ for $i$ in $0..kn - 1]$;
    for $i$ in $0..n - 1$ do
        $delta[flag[i]] = i$;
    end
end
for $i$ in $j..kn - 1$ do
    if $l < n$ and $i == flag[l]$ then
        $l$ += 1;
    end
    else
        $delta[i] = 0$;
    end
end
```

4.3 Environment

The environment corresponds to the method by which the information about the language is being presented. We considered two environments **text** and **informant**.

Definition 4.3 (Text):
*A **text** $T : \mathbb{N} \to \mathbb{N} \cup \{\#\}$ is a infinite sequence of members of $\mathbb{N} \cup \{\#\}$. The set of numbers appearing in a text T is denoted by content(T). And we say a text T is for a set $S \subseteq \mathbb{N}$ if content$(T) = S$.*

A text T can be for any set S, whether it be finite or not, as such repetitions of elements are allowed in a text T. Let $S = \{1, 2, 3\}$, the following texts are both for S: $T_1 = 1\#2\#3\#\#\#\cdots$ and $T_2 = 1\#2\#3\#1\#2\#3\#\cdots$.

Information presentation by text only gives information about what is in the language, but nothing is told about the complement, as opposed to an informant. Since there are bijections between $\mathbb{N}^2$ and $\mathbb{N}$, for example J[1], we define $\langle \cdot, \cdot \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $\langle n, m \rangle := J(n, m)$.

[1] $J : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ given by $J(n, m) = i + \frac{1}{2}((i + j)(i + j + 1))$

Algorithm 5: univ

```
function(p,x):
    y = $\gamma$(p)(x);
    if($is\_natural$(y)) then:
        return $y$;
    else:
        while($True$) do ($Null$)
```

Definition 4.4 (Informant):
A text T is an ***informant*** *for a set $S \subseteq \mathbb{N}$ if content$(T) = \{\langle x, y\rangle : x \in S$ and $y = 1$, or $x \notin S$ and $y = 0\}$.*

An informant is a special kind of text that tells us, explicitly, what is in the language and what is not. If a text is an informant for some set, we represent it by I. We denote x as $\pi_0(\langle x, y\rangle)$ and y as $\pi_1(\langle x, y\rangle)$. An example of informant for the set $S = \{1, 2, 3\}$ would be $I = \langle 1, 1\rangle\#\langle 2, 1\rangle\#\langle 3, 1\rangle\#\langle 4, 0\rangle\#\langle 5, 0\rangle\# \cdots$.

4.4 Scientist

Now that we have all the possible conjectures, a name for each one of the languages, and a way to present words of a language, we need an algorithm to guess it. That will be so-called a *scientist.* But before properly defining it, we need to introduce some notation.

Definition 4.5:

1. *The n^{th} number of a text T is denoted by $T(n)$ and the prefix of length n by $T[n] = T(0)\cdots T(n-1)$.*
2. *SEQ $= \{T[n] : T$ a text for sets and $n \in \mathbb{N}\}$, in other words, SEQ is the collection of all finite sequences over $\mathbb{N} \cup \{\#\}$.*

A scientist will be receiving information through text or informant and at the n^{th} moment $T[n]$ or $I[n]$ will be all the information available to him. Based on that, the scientist may return a conjecture.

Definition 4.6 (Scientist):
A ***scientist*** *is any function $\mathbf{F}$: SEQ $\to \mathbb{N}$.*

One of our goals is to build a program that simulates a scientist, as such, we will restrict ourselves to computable scientists.

4.5 Language identification

Finally, what is left to do is to define a criterion of successful inquiry, i.e., when do we say that a scientist has successfully guessed the language? Let $\mathcal{E}$ be the set of all recursively enumerable languages.

Definition 4.7 (Language identification):
Given a computable scientist $\mathbf{M}$, $\mathscr{L} \subseteq \mathcal{E}$, $L \in \mathscr{L}$, and $i \in \mathbb{N}$:

- $\mathbf{M}$ *converges to i on a text T, if for all but finitely many $n \in \mathbb{N}$, $\mathbf{M}(T[n]) = i$.*
- $\mathbf{M}$ *identifies a text T for L if there is $j \in \mathbb{N}$ such that $\mathbf{M}$ converges to j on T and $W_j =$ content(T).*
- $\mathbf{M}$ **TxtEx**-*identifies L, written $L \in$* **TxtEx**($\mathbf{M}$)*, just in case $\mathbf{M}$ identifies all texts T for L.*
- $\mathbf{M}$ *operating on informants identifies L, written $L \in$* **InfEx**($\mathbf{M}$) *just in case $\mathbf{M}$ identifies all informants I for L.*
- **TxtEx** $= \{\mathscr{L} \subseteq \mathcal{E}$: *there exists a computable scientist* $\mathbf{M}$ *such that* $\mathscr{L} \subseteq$ **TxtEx**($\mathbf{M}$)$\}$
- **InfEx** $= \{\mathscr{L} \subseteq \mathcal{E}$: *there exists a computable scientist* $\mathbf{M}$ *such that* $\mathscr{L} \subseteq$ **InfEx**($\mathbf{M}$)$\}$

We have the criterion of convergence such that when a scientist converges on a text, it converges to a single index on a text. This criterion corresponds to a syntactic convergence.

As we know languages can be recognized by Turing Machines with different indexes. Hence, it would be plausible to allow a type of convergence where the scientist, after some time n, will always conjecture values from a set of indexes, with these indexes corresponding to different Turing Machines that are function-wise the same. This type of converge is said to be semantic convergence.

Definition 4.8 (Canonical Text):
*A text T for $L \in \mathcal{E}$ is said to be **canonical** just in case for all $n \in \mathbb{N}$, $T(n) = \langle n, y \rangle$, with $y = 1$ if $n \in L$ or $y = 0$ otherwise. A prefix $\sigma \in SEQ$ for some informant is in **canonical order** just in case $\sigma = \langle 0, y_0 \rangle \# \langle 1, y_1 \rangle \# \cdots \# \langle n, y_n \rangle$, with $n = |\sigma| - 1$. Let $INIT$ be the collection of all $\sigma \in SEQ$ that are in canonical order.*

An informant for some language L can be converted into a canonical text also for L.

Definition 4.9:
For all $\sigma \in SEQ$ for some informant, $F_A(\sigma)$ is the longest $\tau \in INIT$ such that content$(\tau) \subseteq$ content(σ).

$F_A(\sigma)$ will get the longest prefix in canonical order that can be constructed using elements of σ and it is computable, indeed, take the following Algorithm 6:

Algorithm 6: F_A

```
Input: $\sigma$
Output: $F_A$
for $i \in 0..|\sigma| - 1$ do
    for $j \in 0..|\sigma| - 1$ do
        if $\pi_0(I(j)) == i$ then
            $F_A(i) = I(j)$;
            break;
        end
    end
    break;
end
```

Then, the canonical text converted from an informant I is denoted by $\lim_{n\to\infty} F_A(I[n])$[2].
We do this due to the following proposition:

Proposition 4.1:
A collection of languages $\mathscr{L} \subseteq \mathcal{E}$ is identifiable by some computable scientist operating on informants if and only if there is a computable scientist operating on informants that identifies the canonical text for any member of $\mathscr{L}$.

Proof:
(Necessary condition): If some scientist $\mathbf{M}$ operating on informants identifies $\mathscr{L}$, by definition, $\mathbf{M}$ identifies every informant for $L \in \mathscr{L}$. Since a canonical text for a language is also an informant, $\mathbf{M}$ identifies the canonical text for any member of $\mathscr{L}$.
(Sufficient condition): Suppose a scientist $\mathbf{M}$ operating on informants identifies the canonical text for any member of $\mathscr{L}$. For all prefixes $\sigma \in SEQ$ for some informant, let $\mathbf{M}^*$ be given by $\mathbf{M}^*(\sigma) = \mathbf{M}(F_A(\sigma))$. $\mathbf{M}^*$ is computable because it is the composition of two computable functions.
For any $L \in \mathscr{L}$, $\mathbf{M}$ identifies the canonical text C for L, then given any informant I for L and $n \to \infty$ we have that $F_A(I[n]) \to C$. Thus $\mathbf{M}^*$ converges to $i \in \mathbb{N}$ on informant I for $L \in \mathscr{L}$ if $\mathbf{M}$ converges to i on the corresponding canonical text. Since $\mathbf{M}$ identifies the canonical text for each $L \in \mathscr{L}$, $\mathbf{M}^*$ identifies each $L \in \mathscr{L}$ and thus it identifies $\mathscr{L}$. □

Thus, a computable scientist operating on informants in some way is equivalent to some scientist operating on canonical texts. We will return to these concepts later on when we discuss *strategies*.
A scientist, although computable, can be a partial or total function, however hereafter, we will restrict to the usage of total scientist and this does not limit the cognitive power:

Proposition 4.2:
Let $\mathbf{M}_1, \mathbf{M}_2, \ldots$ be an enumeration of all computable scientists for sets, then there exists a computable map $f : \mathbb{N} \to \mathbb{N}$ such that for all $i \in \mathbb{N}$:

1. *$\mathbf{M}_{f(i)}$ is total,*
2. *for all $L \in \mathcal{E}$, if $\mathbf{M}_i$ identifies L then $\mathbf{M}_{f(i)}$ identifies L.*

[2]This notation is defined in Definition 4.10

Proof:
We would like $\mathbf{M}_{f(i)}$, given σ as input, to simulate $\mathbf{M}_i$ on all prefixes $\tilde{\sigma}$ of σ, for at most $|\sigma|$ steps and to return the result of $\mathbf{M}_i$ on the longest prefix τ such that $\mathbf{M}_i(\tau)$ converges. Define: $\mathbf{M}_{f(i)}(\sigma) = \mathbf{M}_i(\tilde{\sigma})$ with $\tilde{\sigma}$ the longest prefix of σ such that $\mathbf{M}_i(\tilde{\sigma})$ is defined in at most $|\sigma|$ steps, if such $\tilde{\sigma}$ exists; $\mathbf{M}_{f(i)}(\sigma) = 0$ otherwise.

By definition, $\mathbf{M}_{f(i)}$ is total. To see that $\mathbf{M}_{f(i)}$ identifies every language L that $\mathbf{M}_i$ identifies, let I be an informant for L. Then there is $n \in \mathbb{N}$ and j index for L, such that for all $m \geq n$, $\mathbf{M}_i(I[m]) = j$. Let $s \in \mathbb{N}$ be the running time of $\mathbf{M}_i(I[m])$. Then for all $m \geq \max\{n, s\}$ $\mathbf{M}_{f(i)}(I[m]) = \mathbf{M}_i(I[n]) = j$, and so $\mathbf{M}_{f(i)}$ identifies L. □

One might expect that a scientist who is receiving information from an informant is capable of identifying a wider collection of languages than from a text, as an informant can be related to decidability and a text to listability.

Proposition 4.3:
TxtEx $\subseteq$ InfEx

Proof:
Supposing that $\mathbf{M}_{txt}$ **TxtEx**-identifies $\mathscr{L}$. The idea is that we let the scientist $\mathbf{M}_{inf}$ do the same as $\mathbf{M}_{txt}$ while ignoring the information about the words which are not in the language. $\mathbf{M}_{inf}$ given a prefix σ of an informant, creates a prefix $\tilde{\sigma} = \{x \in \mathbb{N} : \langle x, y\rangle \in \sigma \text{ and } y = 1\}$ for a text and feeds it into $\mathbf{M}_{txt}$, returning its result. □

We start with results on **TxtEx**-identification:

Proposition 4.4:
The collection of all finite languages, $\mathcal{FIN} = \{D \subseteq \mathbb{N} : D \textit{ is finite}\}$, is **TxtEx***-identifiable.*

Proof:
Let $\mathbf{M}_{finite}(\sigma)$ = index of a Turing Machines that enumerates *content*(σ). $\mathbf{M}_{finite}$ **TxtEx**-identifies $\mathcal{FIN}$, since given any $L \in \mathcal{FIN}$ and text T for L, *content*$(T) = L$, then there exists n such that *content*$(\sigma[n])$ = *content*$(T) = L$, from here and onwards, for all $m \geq n$, $\mathbf{M}_{finite}(\sigma[m]) = i$, with $W_i = L$. □

Corollary 4.1:
$\mathcal{FIN}$ is identifiable by some computable scientist operating on informants.

To show that some languages are not **TxtEx**-identifiable, we introduce some new notations:

Definition 4.10:
Let $\sigma, \tau \in SEQ$:

- *σ is consistent with L if $\langle x, 1\rangle \in content(\sigma)$ implies that $x \in L$ and if $\langle x, 0\rangle \in content(\sigma)$ implies that $x \notin L$.*
- *$\sigma \diamond \tau$ denotes the result of concatenating τ onto the end of σ. However, when σ and τ are prefixes of informants such operation is well defined when for any pairings $\langle x, y\rangle, \langle x', y'\rangle \in \sigma \diamond \tau$, if $x = x'$ then $y = y'$ or, equivalently, $\sigma \diamond \tau$ is defined if there is $L \in \mathcal{E}$ such that both σ and τ are consistent with.*
- *We write "$\sigma \subseteq \tau$" if σ is a prefix of τ, and "$\sigma \subset \tau$" if σ is a proper prefix of τ.*
- *Let the sequence $\sigma^0, \sigma^1, \sigma^2 \cdots$ be such that $\sigma^0 \subset \sigma^1 \subset \sigma^2 \cdots$ and $lim_{i\to\infty}|\sigma^i| = \infty$. Then there is a unique informant I such that for all $n \in \mathbb{N}$, $\sigma^n = I[|\sigma^n|]$. This is denoted $\lim_{i\to\infty} \sigma^i$.*

Theorem 4.1 (Blum and Blum [2]):
Suppose that scientist $\mathbf{F}$, not necessarily computable, operating on informants, identifies $L \in \mathcal{E}$, then there is $\sigma \in SEQ$ such that:

1. *σ is consistent with L ,*
2. *$W_{\mathbf{F}(\sigma)} = L$,*
3. *For all $\tau \in SEQ$, if τ is consistent with L, then $\mathbf{F}(\sigma \diamond \tau) = \mathbf{F}(\sigma)$.*

Proof:
By contradiction, we assume that there is no prefix $\sigma \in SEQ$ meeting the previous three conditions. Then we would have that for all $\sigma \in \text{SEQ}$ such that σ is consistent with L and $W_{\mathbf{F}(\sigma)} = L$, there is some $\tau \in \text{SEQ}$ such that τ is consistent with L and $\mathbf{F}(\sigma \diamond \tau) \neq \mathbf{F}(\sigma)$.

We show that there is an informant I for L which $\mathbf{F}$ does not identify, contrary to the hypothesis that $\mathbf{F}$ identifies L. Let $U = u_0\#u_1\#u_2...$ be an arbitrary informant for L. The informant I is constructed in stages.

Stage 0 : $\sigma^0 = \varepsilon$
Stage $n+1$: With σ^n being already defined, σ^{n+1} is given by:

$$\sigma^{n+1} = \begin{cases} \sigma^n \diamond u_n & \text{if } W_{\mathbf{F}(\sigma^n)} \neq L \\ \sigma^n \diamond \tau \diamond u_n & \text{if } W_{\mathbf{F}(\sigma^n)} = L \end{cases}$$

At all stages, σ^n is consistent with L, since at (0) it is empty and in each step, we are adding finite sequences which are consistent with L, with τ being given by our hypothesis. Note that, all concatenations performed at each stage are defined because every prefix concatenated is consistent with L.

By construction $\sigma^n \subset \sigma^{n+1}$ for all $n \in \mathbb{N}$, as such, we let $I = \lim_{n\to\infty}\sigma^n$. I is indeed an informant for L, since u_n is added at stage (n+1) and all sequences added are consistent with L. Finally, $\mathbf{F}$ does not converge on I to an index for L, since for each $n+1 \in \mathbb{N}$ either $W_{\mathbf{F}(\sigma^n)} \neq L$ or, for some $\tau \in \text{SEQ}$, $\mathbf{F}(\sigma \diamond \tau) \neq \mathbf{F}(\sigma)$. □

A sequence that meets the previous statement is denoted **locking sequence** for $\mathbf{F}$ on L. Note that, the same result holds true for scientists operating on texts, the proof being very similar.

With this result, we can show that some collections of languages are not **TxtEx**-identifiable:

Proposition 4.5:

1. *$\{\mathbb{N}\} \cup \mathcal{FIN}$ is not* **TxtEx**-*identifiable.*
2. *$\{\mathbb{N}\} \cup \{\mathbb{N} - \{x\} : x \in \mathbb{N}\}$ is not* **TxtEx**-*identifiable.*

Proof:
By contradiction, if it is **TxtEx**-identifiable, there is a scientist $\mathbf{M}$ that **TxtEx**-identifies the collection, which means, by the previous theorem, there is a locking sequence σ for $\mathbf{M}$ on $\mathbb{N}$:

1. Since $|\sigma| < \infty$, *content*$(\sigma) \in \mathcal{FIN}$, as such, there is a text T for *content*(σ) such that $T[|\sigma|] = \sigma$. But then $\mathbf{M}$ does not identify *content*(σ) since σ is a locking sequence for $\mathbf{M}$ on $\mathbb{N}$.
2. Similarly, $|\sigma| < \infty$ so we choose $x \notin$ *content*(σ). Then, on any text T for $\mathbb{N} - \{x\}$ such that $T[|\sigma|] = \sigma$, $\mathbf{M}$ converges to an index for $\mathbb{N}$. □

One would suggest that since Theorem 4.1 holds for informants, the previous two collections are also not identifiable by computable scientists operating on informants, though that is not the case. The argument we have used to prove the previous proposition relies on the fact that since $\mathbb{N}$ is the entire set, any finite sequence for a text for finite language is also a finite sequence for a text for $\mathbb{N}$, however, this is not true for informants. This means that condition (3) in Theorem 3.1 at some point will be false for informants.

Theorem 4.2 (Angluin [1]):
$\mathscr{L} \subseteq \mathcal{E}$ is identifiable by a scientist, not necessarily computable, operating on texts if and only if for all $L \in \mathscr{L}$ there is finite $D_L \subseteq L$ such that for all $L' \in \mathscr{L}$ and $L \neq L'$, if $D_L \subseteq L'$ then $L' \not\subset L$

Proof:
(Necessary condition): Let $\mathbf{F}$ be the scientist operating on texts that identifies $\mathscr{L}$ and σ_L a locking sequence for $\mathbf{F}$ on $L \in \mathscr{L}$. Since *content*(σ_L) is finite, we are left to show that for all $L' \in \mathscr{L}$ if *content*$(\sigma_L) \subseteq L' \Rightarrow L' \not\subset L$.

If that wasn't the case, i.e., there is $L' \in \mathscr{L}$ such that *content*$(\sigma_L) \subseteq L'$ and $L' \subseteq L$, we would have a text T for L' such $T[|\sigma_L|] = \sigma_L$. We would expect that $\mathbf{F}$ would converge to an index for L' on T, since $\mathbf{F}$ identifies $\mathscr{L}$, but since σ_L is a locking sequence for $\mathbf{F}$ on L, it doesn't happen.

(Sufficient condition): We need to find a scientist **F** operating on texts that identifies $\mathscr{L}$, we define it as follows:

$$\mathbf{F} = \begin{cases} \text{The least index } i \text{ for L, such that } D_L \subseteq \mathit{content}(\sigma) \subseteq L & \text{, if there exists such } i \\ 0 & \text{, otherwise} \end{cases}$$

Fix $L \in \mathscr{L}$ and a text T for L, then there is n_0 such that $D_L \subseteq \mathit{content}(T[n_0]) \subseteq L$. Let i be the least index for L.

We know that **F** will not converge to indexes bigger than i, so we are left to show that for all $j < i$ and $m \in \mathbb{N}$, if j is index for some $L' \in \mathscr{L}$ and $D_{L'} \subseteq T[m]$, then at some point $n \in \mathbb{N}$, $\mathit{content}(T[n]) \nsubseteq L'$. Assuming those hypothesis, we would have that $L \neq L'$ and $D_{L'} \subseteq \mathit{content}(T[n]) \subseteq L$, since i is the least index for L, which means there exists $x_0 \in L - L'$, by hypothesis, and as T is text for L, at some point $n \in \mathbb{N}$, $x_0 \in \mathit{content}(T[n]) \Rightarrow \mathit{content}(T[n]) \nsubseteq L'$. □

Obviously, the theorem does not hold for informants, otherwise, the collection of languages identifiable by some scientist operating on texts and informants would be the same, which later on we will prove to be different. Namely, the necessary condition does not hold, indeed, just like before the argument using a locking sequence doesn't do the trick. But, nevertheless, we have successfully characterized the collection of languages identifiable by some scientist operating on texts.

Finally, we will show that a scientist operating on informants has more cognitive power than one operating on texts.

Proposition 4.6 (Gold [5]):
$\mathcal{E}$ is identifiable by a non-computable scientist operating on informants.

Proof:
We define a scientist **F** by:

$$\mathbf{F}(\sigma) = \begin{cases} \text{least index } i \text{ of a language for which } \sigma \text{ is consistent with} \\ 0 \text{ , if such index does not exist} \end{cases}$$

Given $L \in \mathscr{L}$, it is easy to see that **F** converges to the least index i of L. Indexes greater than i won't be a problem, since **F** goes through i and will settle with it. As for an index lesser $j < i$, **F** won't converge to any of them, because the language it indexes is different from L and, as such, there is either a word which is in L and not the other, or vice-versa, meaning at some point σ is not consistent with the language of index j. □

Corollary 4.2 (Gold [5]):
The set of classes of languages identifiable by some scientist operating on texts is strictly contained in the set of classes of languages identifiable by some scientist operating on informants.

Proposition 4.7:
$\mathcal{FIN} \cup \{\mathbb{N}\}$ is identifiable by some computable scientist operating on informants.

Proof:
Define **M**:

$$\mathbf{M}(\sigma) = \begin{cases} \text{index of a Turing Machine that enumerates } \mathbb{N} & \text{, if } y = 1 \text{ for all } \langle x, y \rangle \in \sigma \\ \text{index of a Turing Machine that enumerates } \{x \in \mathbb{N} : \langle x, 1 \rangle \in \sigma\} & \text{, otherwise} \end{cases}$$

Given $L \in \mathcal{FIN} \cup \{\mathbb{N}\}$, **M** will converge to an index of L. If $L = \mathbb{N}$ then no negative information will ever be provided thus **M** converge to 0 on any informant for L. If not, then L is finite and there is $x \notin L$, which means for any informant I for L there are $n, m \in \mathbb{N}$ such that $\{x \in \mathbb{N} : \langle x, 1 \rangle \in I[n]\} = L$ and $\langle x, 0 \rangle \in I[m]$. Thus for all $t \geq \max\{n, m\}$, $\mathbf{M}(I[t]) =$ index of a Turing Machine that enumerates L. □

Proposition 4.8:
$\{\mathbb{N}\} \cup \{\mathbb{N} - \{x\} : x \in \mathbb{N}\}$ is identifiable by some computable scientist operating on informants.

Proof:
We define **M** by:

$$\mathbf{M}(\sigma) = \begin{cases} \text{index of a Turing Machine that enumerates } \mathbb{N} & \text{, if } y = 1 \text{ for all } \langle x, y \rangle \in \sigma \\ \text{index of a Turing Machine that enumerates } \mathbb{N} - \{x\} & \text{, otherwise} \end{cases}$$

$\mathbf{M}$ identifies all $L \in \{\mathbb{N}\} \cup \{\mathbb{N} - \{x\} : x \in \mathbb{N}\}$. Indeed, if $L = \mathbb{N}$ then negative information won't ever be provided, thus $\mathbf{M}$ returns the index of a Turing Machines that enumerates $\mathbb{N}$ for all prefixes of any informant for L.

Else there is one $x \notin L$. Then for any informant I for L, there is $n \in \mathbb{N}$ such that for all $m \geq n$, $\langle x, 0\rangle \in I[m]$. Hence for $m \geq n$, $\mathbf{M}(I[m])$ = index of a Turing Machine that enumerates $\mathbb{N} - \{x\}$. □

Corollary 4.3 (Gold [5]):
TxtEx $\subset$ **InfEx**.

Although the act of restriction ourselves to consider only total scientists does not have any impact on the classes of languages that are identifiable, the same does not apply when we go from any scientist to a computable one, i.e., an arbitrary scientist is capable of identifying a wider class of language than what a computable scientist is:

Proposition 4.9 (Gold [5]):
$\mathcal{E}$ is not identifiable by a computable scientist operating on informants.

Proof:
By contradict on, we suppose that there is a computable scientist $\mathbf{M}$ operating on informants that identifies $\mathcal{E}$. We proceed to construct an informant I for a language $L \in \mathcal{E}$ such that $\mathbf{M}$ cannot identify.

First, we claim that given any prefix σ for an informant, such that $x \leq n$ for every pairing $\langle x, y\rangle \in \sigma$, then there are numbers j and k such that $\tau = \sigma \diamond \langle n+1, 0\rangle \# \cdots \# \langle n+j, 0\rangle$ and $\tau' = \tau \diamond \langle n+j+1, 1\rangle \# \cdots \# \langle n+j+k, 1\rangle$ such that $\mathbf{M}(\tau) \neq \mathbf{M}(\tau')$.

To prove the claim, notice that $\sigma \diamond \langle n+1, 0\rangle \# \cdots \# \langle n+j, 0\rangle \# \cdots$ is an informant for some finite language $L_0 \in \mathcal{E}$, which by hypothesis $\mathbf{M}$ identifies, then there is j such that if $\tau = \sigma \diamond \langle n+1, 0\rangle \# \cdots \# \langle n+j, 0\rangle$, $\mathbf{M}(\tau)$ is an index for L_0. And let $\tau \diamond \langle n+j+1, 1\rangle \# \cdots \# \langle n+j+k, 1\rangle \# \cdots$ an informant for an infinite language $L_1 \in \mathcal{E}$, for the same reason there exists k such that if $\tau' = \tau \diamond \langle n+j+1, 1\rangle \# \cdots \# \langle n+j+k, 1\rangle$ and $\mathbf{M}(\tau')$ is an index for L_1. As L_0 is finite and L_1 is not, $\mathbf{M}(\tau) \neq \mathbf{M}(\tau')$ and the pair (j, k) can be found recursively, since $\mathbf{M}$ is computable and $\mathbb{N}^2$ enumerable. Then we construct σ^n in stages:

Stage $0 : \sigma^0 = \langle 0, 0\rangle$
Stage $n+1$: With σ^n defined, $\sigma^{n+1} = \tau'$, τ' given by the previous claim.

Notice that, at each stage, we are concatenating prefixes with new values of x, so it is defined.

Finally $I = \lim_{n\to\infty} \sigma^n$ is an informant for some $L \in \mathcal{E}$. However, $\mathbf{M}$ does not converge on I, since it changes its guess at least once for each $n \in \mathbb{N}$. □

Corollary 4.4:
An arbitrary scientist operating on informants is capable of identifying a strictly wider set of class of languages than a computable scientist operating on informants.

5 Strategies for Learning

In this section, we will consider some other constraints on scientists, meaning that alternative subsets of the class of computable scientists will be defined and we would like to see on which our scientist falls into. Any such subset is called *strategy*.

For that purpose, a pseudocode of our scientist will be presented in Algorithm 7.

The scientist will first check if σ is still consistent with the language accepted by the current conjecture, returning it if true, else it will go through the enumeration until it finds a minimal DFA that recognizes a language for which σ is consistent with.

We begin by defining all constraints and, then, explaining on which strategies our scientist falls into.

The following constraints are defined for a scientist operating on texts, but since our scientist operates on informants, we will assume that $\sigma, \tau \in SEQ$ are prefixes of some informant.

Definition 5.1 (Totality):

- $\mathbf{M}$ *is always defined on* L*, if for every* $\sigma \in SEQ$ *consistent with* L*, we have* $\mathbf{M}(\sigma)\downarrow$.
- $\mathbf{M}$ *is always defined on* $\mathscr{L}$*, if* $\mathbf{M}$ *is always defined on each* $L \in \mathscr{L}$.

Algorithm 7: Pseudocode of a scientist operating on informants that identifies regular languages

```
class scientist:
    dfa $conjecture = get\_next\_dfa()$;
    def $get\_conjecture$(list $strings$):
        if $strings$ is consistent with $L(conjecture)$ then
            return $conjecture$;
        end
        do:
            $conjecture = get\_next\_dfa(conjecture)$;
        while($strings$ is not consistent with $L(conjecture)$ or $conjecture$ is non-minimal)
        return $conjecture$;
```

- $\mathbf{M}$ *is always defined, if* $\mathbf{M}$ *is always defined on all* $L \in \mathcal{E}$.

Definition 5.2 (Consistency):

- $\mathbf{M}$ *is consistent on* L*, if for all* $\sigma \in SEQ$ *consistent with* L*,* σ *is also consistent with* $W_{M(\sigma)}$.
- $\mathbf{M}$ *is consistent on* $\mathscr{L}$*, if* $\mathbf{M}$ *is consistent on every* $L \in \mathscr{L}$.
- $\mathbf{M}$ *is consistent, if* $\mathbf{M}$ *is consistent on* $\mathcal{E}$.

Definition 5.3 (Prudence):
$\mathbf{M}$ *is said to be prudent, if for all* $\sigma \in SEQ$*,* $\mathbf{M}(\sigma)$ *is defined, then* $\mathbf{M}$ *identifies* $W_{\mathbf{M}(\sigma)}$.

Definition 5.4 (Set-driven):
$\mathbf{M}$ *is set driven, if for all* $\sigma, \tau \in SEQ$*, if content(*σ*) = content(*τ*), then* $\mathbf{M}(\sigma) = \mathbf{M}(\tau)$.

Definition 5.5 (Rearrangement-independent):
$\mathbf{M}$ *is rearrangement-independent, if for every* $\sigma, \tau \in SEQ$ *with content(*σ*) = content(*τ*) and* $|\sigma| = |\tau|$*, we have* $\mathbf{M}(\sigma) = \mathbf{M}(\tau)$.

Definition 5.6 (Reliability):

- $\mathbf{M}$ *is reliable on* L*, if for any informant* I *for* L*, if* $\mathbf{M}$ *converges on* I *then* $W_{\mathbf{M}(I)} = L$.
- $\mathbf{M}$ *is reliable on* $\mathscr{L}$*, if* $\mathbf{M}$ *is reliable on each* $L \in \mathscr{L}$.
- $\mathbf{M}$ *is reliable, if* $\mathbf{M}$ *is reliable on* $\mathcal{E}$.

Definition 5.7 (Conservatism):
$\mathbf{M}$ *is conservative on* L*, if for all* $\sigma, \tau \in SEQ$ *such that* $\sigma \subsetneq \tau$*, if content*(τ) *is consistent with* $W_{\mathbf{M}(\sigma)}$*, then* $\mathbf{M}(\tau) = \mathbf{M}(\sigma)$.

All strategies listed above concern languages, we can use them to classify our scientist based on the language accepted by the DFA it conjectures.

A constraint has been omitted, we say $\mathbf{M}$ operating on texts is order-independent on L just in case for all texts T, T' for L, if $\mathbf{M}(T)$ is defined, then $\mathbf{M}(T) = \mathbf{M}(T')$. One would say we could define the same constraint for scientists operating on informants by replacing texts for informants in the definition.

However, as we have discussed in Section 4.5, if a computable scientist operating on informants identifies some class of languages then there is a computable scientist that identifies the canonical text for every language in that class. Plus there is an effective procedure to convert an informant into a canonical text, as such, it is equivalent to identifying the canonical text of a language.

The following constraints refer to functions, then we say a text T is for a map f if *content*$(T) = \{\langle x, f(x)\rangle : x \in \mathbb{N}\}$. Let $f[n] = \{\langle x, f(x)\rangle : x < n\}$ in ascending order of x.

Let $\mathcal{R}$ be the set of all recursive maps:

Definition 5.8 (Popperian):

- $\mathbf{M}$ *is Popperian on* f*, if and only if for all* $n \in \mathbb{N}$*,* $\phi_{\mathbf{M}(f[n])} \in \mathcal{R}$.
- $\mathbf{M}$ *is Popperian on* $\mathcal{S} \subset \mathcal{R}$*, if and only if* $\mathbf{M}$ *is Popperian on each* $f \in \mathcal{S}$.

- $\mathbf{M}$ *is Popperian if and only if* $\mathbf{M}$ *is Popperian on* $\mathcal{R}$.

Definition 5.9 (Decisive):

- $\mathbf{M}$ *is decisive on* f *if and only if for all* $l, m, n \in \mathbb{N}$ *with* $l < m < n$ *and* $\phi_{\mathbf{M}(f[l])} \neq \phi_{\mathbf{M}(f[m])}$, *then* $\phi_{\mathbf{M}(f[l])} \neq \phi_{\mathbf{M}(f[n])}$.
- $\mathbf{M}$ *is decisive on* $\mathcal{S} \subset \mathcal{R}$ *if and only if* $\mathbf{M}$ *is decisive on each* $f \in \mathcal{S}$.
- $\mathbf{M}$ *is decisive if and only if* $\mathbf{M}$ *is decisive on* $\mathcal{R}$.

Note that the Popperian constraint as it is does not impose any restriction on the type of convergence of $\mathbf{M}$, i.e, $\mathbf{M}$ on f can converge to a set of indexes, as explained in Section 4.5. Though, since we want a syntactic convergence, that is, convergence to a single index, we assume that $\mathbf{M}$ does exactly that in case it converges.

We, now, explain which strategies our scientist, name it **Reg**, belongs to. Let $\mathcal{REG}$ be the class of regular languages and, as **Reg** is only prepared to receive prefixes of an informant, we restrict every finite sequence σ, τ to just that. We say a fprefix σ of some informant is consistent with a DFA D if and only if σ is consistent with $L(D)$.

Reg is:

1. Always defined on $\mathcal{REG}$, as **Reg** is receiving prefixes from an informant I for some regular language L. Then there is some DFA D which recognizes L, thus every prefix will be consistent with D. This means that if no other DFA D' is consistent with the prefix of I, then D will be conjectured,
2. Consistent on $\mathcal{REG}$, since, by construction, it only looks for a DFA D such that σ is consistent with it,
3. Prudent, since it identifies $\mathcal{REG}$,
4. Set-driven, because the scientist returns the first DFA D such that σ is consistent with $L(D)$ and since *content*(σ) = *content*(τ), σ, τ are not consistent with any previous DFA,
5. Rearrangement-independent, for the same reason as above,
6. Not reliable on $\mathcal{REG}$, as it always conjectures a DFA which doesn't accept any word,
7. Conservative, because while σ is consistent with the conjectured DFA, it does not look for any other,
8. Popperian on the set of all informants for $\mathcal{REG}$ as the characteristic function we can define from a DFA is total:
$$f(x) = \begin{cases} 1 & \text{if } \delta(x, q_0) \in F \\ 0 & \text{otherwise} \end{cases}$$
9. Decisive on the set of all informants for $\mathcal{REG}$, since the enumeration does not come back to a DFA enumerated before.

6 Implementation of the Scientist Reg

The program was implemented in Python with the source code and library of examples available at `https://github.com/chengandre/Reg`. Alongside the Scientist and DFA classes, it was implemented a GUI for the application using the `tkinter` module, which is native in Python, to ease the use and demonstration of the application.

Other non-native modules were used:

- `Graphviz` is used to render the conjectured DFAs into png files. The Installer is available at https://graphviz.org/download/, though, we recommend that users who already have Python, to install the module through the following command:

  ```
  $ pip install graphviz
  ```

- `PIL` is used to open and manage images files rendered by `Graphviz` and can be installed through the following command:

  ```
  $ pip install pillow
  ```

- `Tkinter` is usually included with the standard Python distributions. However, if it is not available, the user can use the following command to install it:

  ```
  $ pip install tk
  ```

Even though a similar project named GOLD, available at https://github.com/gamatos/gold, was implemented before by Gabriel Matos, we have decided to take a different approach and start from scratch. Our implementation is simpler, though much slower, as only two classes were implemented:

- The class scientist does roughly what is specified in the Algorithm 7,
- The class dfa, as the name suggests, implements a deterministic finite automaton, while being able to update itself to the next DFA in the Reis' enumeration. Here, as opposed to GOLD, we have directly used the Algorithms 3 and 4 for the enumeration with just slight adjustments and correction compared to the ones specified in [11]. Also, we have gone with the classic DFA minimization Algorithm 1 that has a time complexity of $\mathcal{O}(kn^2)$, instead of Hopcrofts's algorithm available in [6], which is $\mathcal{O}(n\log(n))$, used in GOLD.

However, an interface was created which is intended to facilitate the use of the application, allowing the user to add words to be accepted or rejected on the fly, export the current session and load previous ones. These features are explained in the next Subsection 6.1.

In each session, words and images will be automatically stored in the `current_session` folder, which is created at launch in the same directory as the script file.

Note that, a Windows installer is available on GitHub, as Windows users might face problems installing Python and the modules since Windows might not know where are the installed modules located. The user can choose to use the installer, which will install the application and dependencies into a selected folder. After the installation, the user can run the application through the `App` shortcut, which redirects to `./lib/gui.exe`, this was made due to `gui.exe` needing the be in the same directory as all the dependencies. The directory `./lib` has a lot other `.exe` files including `gui.exe`, which would make finding the application harder.

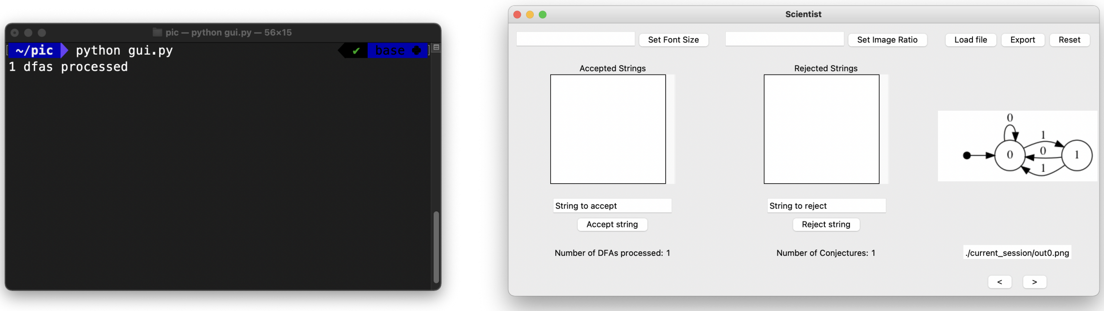
If the Windows installer method was chosen, the user need not have Python or any other modules to run the application, although to install it might require around 1GB of storage space, as the compiler only allows to include every module even ones which are not needed and deleting those would cause Windows Defender to give a false positive. If storage is a problem, after the installation, the folder `./lib/lib/mkl` can be deleted.

6.1 User Manual

After cloning the repository or installing the application, to use the application simply run `gui.py` using the following command or double-click the App shortcut:

```
$ python gui.py
```

The user will then be presented with the following windows, Figure 4:



```
~/pic python gui.py
1 dfas processed
```

Figure 4: Terminal (Left) and Application (Right) at launch

The use of the terminal is just to make sure that the application is still running and has not crashed, indicating how many DFAs were processed up until the moment while the scientist is updating its conjectures, since, in the meantime, the interface becomes unresponsive.

The interface allows, at the top side, to set a font size, which is equivalent to zooming in or out, as well as setting the ratio by which the images will be displayed. Alongside it, are the Loading, Exporting, and Resetting features.

The export feature allows the user to export the current session to a selected folder, after doing so, the images of each conjectured DFAs will be copied to that folder alongside a text file with all the words accepted and rejected, in order. Each line of the text file has a boolean indicating if a word should be accepted or not and, separated by a space, the word itself.

The user can load the text file exported from a previous session. The application will load the words one by one, as such the application is reprocessing those words. If some word was added to the opposite list, the user can reset the current session, which will revert back the application to its launch state, deleting any files in the `current_session` folder.

At this point, the user can choose to add words, manually, or load them from an exported text file, obviously after loading a text file, words can still be added and will be displayed in the respective list.

On the right side, an image of the current conjectured DFA is shown and the user can cycle through all the conjectures made by the scientist with the two arrow buttons in the bottom. The interface will also be updating the number of DFAs processed and conjectures taken after each word has been added. As mentioned before since the GUI becomes unresponsive while the scientist is updating its conjecture, the terminal will be updating the number every 100 000 DFAs processed.

Note that, to add the empty word the user can simply (1) click on the input box, (2) not type anything or the word "eps", and (3) hit Accept/Reject string. Both ways will result in the word "eps" being displayed, even if nothing was typed. Also, when exporting the current session, if the empty word was added at some point, then the word "eps", which means the empty word, will be added to the exported text file.

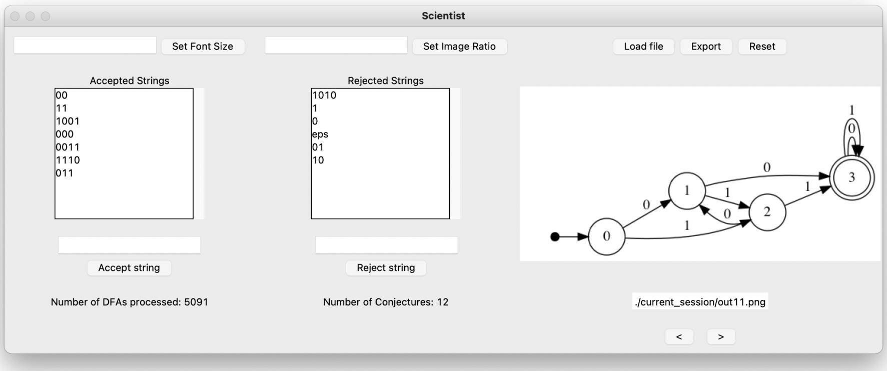
If, at some point, the user has tried to add a word not in $\Sigma^\star = \{0,1\}^*$, except "eps", or a word that has already been added, the application will ignore it and do nothing.

6.2 Examples

Alongside the source code comes a library of examples, 6 in total, which includes a text file and images exported from previous sessions. The user can load the `string.txt` file in each folder to check if the results match the images available in the folder. The user can choose to add the words shown in each figure manually, which should result in the same conjectures, as explained in Section 5. Each of the following figures shows what the interface should look like after loading the text files.

6.2.1 Word with two consecutive 0s or 1s

The text file, in the folder `./%00%or%11%`, contains a finite initial sequence of an informant for the regular language $L_1 = \{w \in \Sigma^\star : w \text{ has two at least consecutive 0s or 1s}\}$, with which the scientist can identify the minimal DFA that accepts L_1. After loading, the interface should look this Figure 5:



Accepted Strings

```
00
11
1001
000
0011
1110
011
```

Rejected Strings

```
1010
1
0
eps
01
10
```

Number of DFAs processed: 5091

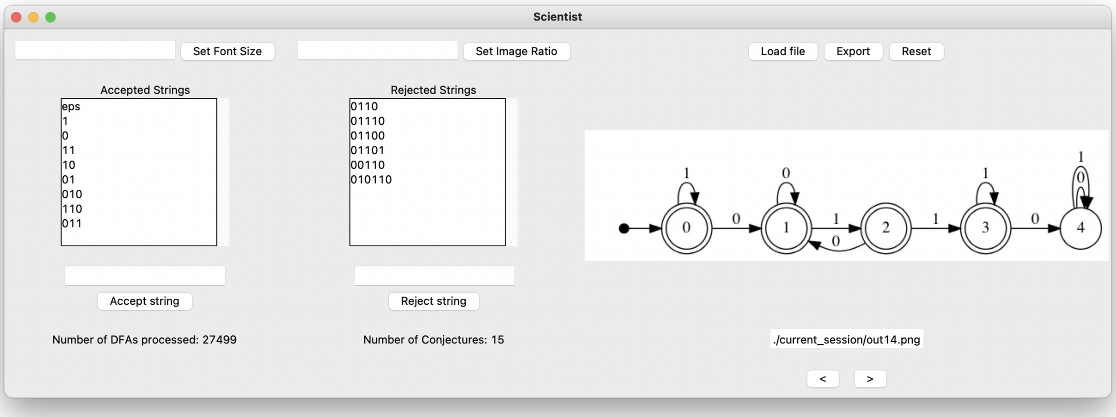
Number of Conjectures: 12

./current_session/out11.png

Figure 5: Interface after loading `./%00%or%11%/string.txt`

6.2.2 Word such that between two 0s there is at most one 1

The text file, in `./%0max(1)0%` allows the scientist to identify $L_2 = \{w \in \Sigma^\star :$ between two 0s there is at most one 1$\}$. After loading the file, it should look like the following figure 6:



Accepted Strings

```
eps
1
0
11
10
01
010
110
011
```

Rejected Strings

```
0110
01110
01100
01101
00110
010110
```

Number of DFAs processed: 27499

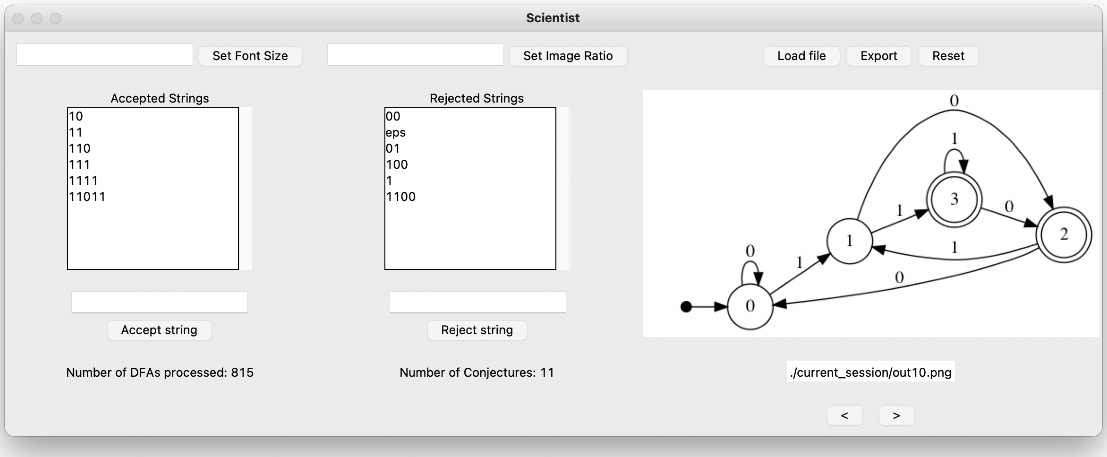
Number of Conjectures: 15

./current_session/out14.png

Figure 6: Interface after loading `./%0max(1)0%/string.txt`

6.2.3 Word that has a 1 in the penultimate position

After loading `./%1_/string.txt`, the DFA shown in Figure 7 accepts $L_3 = \{w \in \Sigma^\star :$ the penultimate symbol is 1$\}$.



Accepted Strings

```
10
11
110
111
1111
11011
```

Rejected Strings

```
00
eps
01
100
1
1100
```

Number of DFAs processed: 815

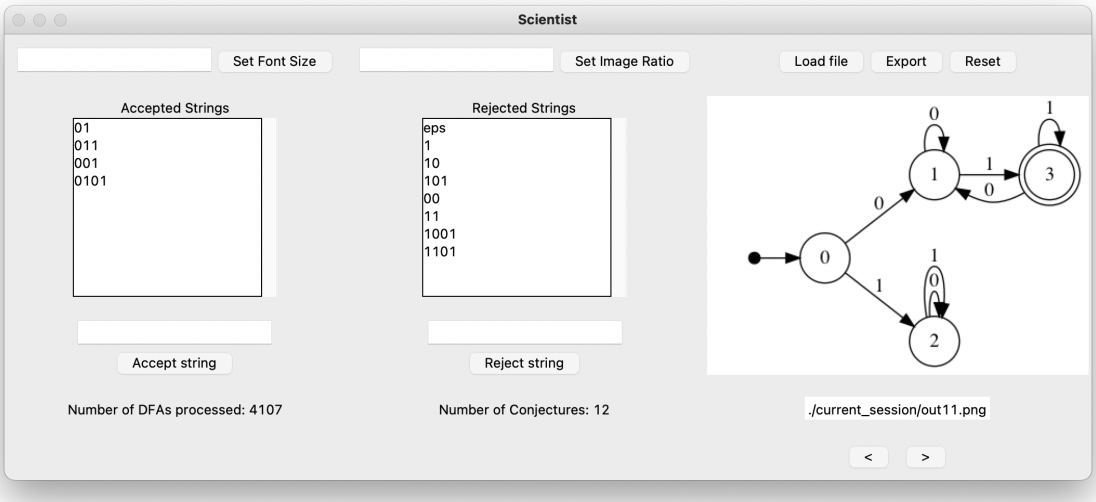
Number of Conjectures: 11

./current_session/out10.png

Figure 7: Interface after loading `./%1_/string.txt`

6.2.4 Word that starts with a 0 and ends with a 1

The DFA in the next Figure 8 accepts $L_4 = \{w \in \Sigma^\star : w \text{ starts with 0 and ends in a 1}\}$.



Accepted Strings

```
01
011
001
0101
```

Rejected Strings

```
eps
1
10
101
00
11
1001
1101
```

Number of DFAs processed: 4107

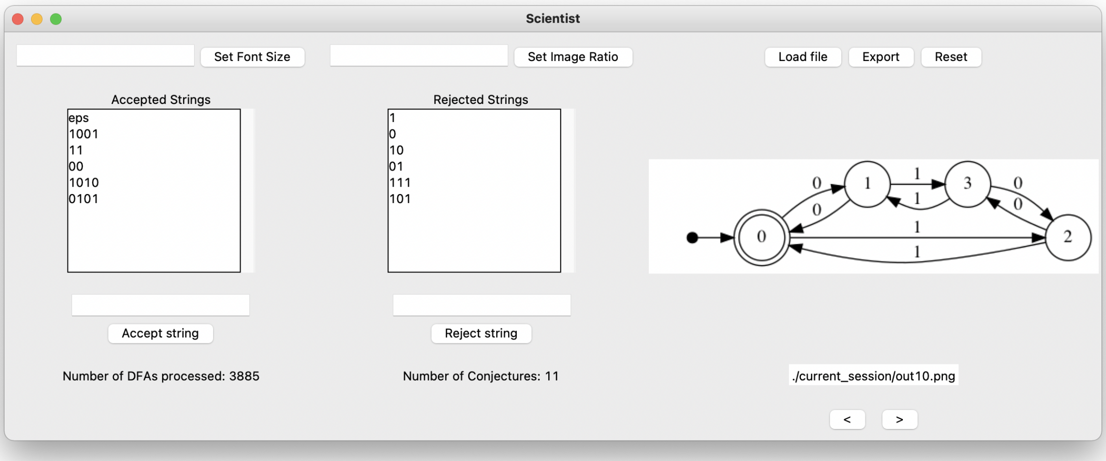
Number of Conjectures: 12

./current_session/out11.png

Figure 8: Interface after loading `./0%1/string.txt`

6.2.5 Word with even number of 0s and 1s

Two text files in two different folders are provided, `./even0and1_more_clues` and `even0and1_more_guesses`. Loading either text file, the scientist will be able to identify the DFA, however, the scientist provided the text file that has more words makes fewer conjectures and the other makes more conjectures but with fewer words provided. Figure 9 shows the result after loading `./even0and1_more_guesses/string.txt`:



Accepted Strings

```
eps
1001
11
00
1010
0101
```

Rejected Strings

```
1
0
10
01
111
101
```

Number of DFAs processed: 3885

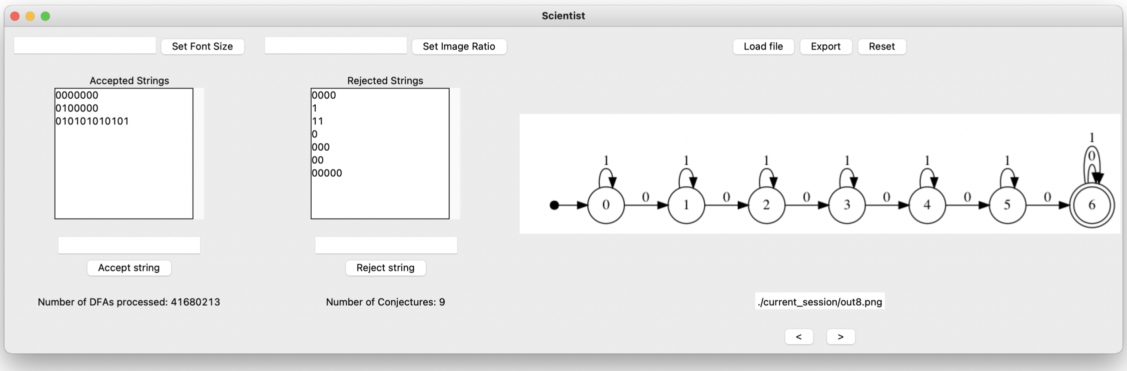
Number of Conjectures: 11

./current_session/out10.png

Figure 9: Interface after loading `./even0and1_more_guesses/string.txt`

6.2.6 Word with at least six 0s

This example is just to show that the scientist is capable of identifying DFAs with more states, though, takes a much longer time. The next DFA, in Figure 10, recognizes $L_6 = \{w \in \Sigma^\star : w \text{ has at least six 0s}\}$. The scientist took around 3 minutes to identify it with an Intel Core i9-9980H 8-Core CPU.



Accepted Strings

```
0000000
0100000
010101010101
```

Rejected Strings

```
0000
1
11
0
000
00
00000
```

Number of DFAs processed: 41680213

Number of Conjectures: 9

./current_session/out8.png

Figure 10: Interface after loading `./at_least_6_0s/string.txt`

6.3 Future Work

Our main goal was to make the application easy to use and to demonstrate, as such, not many optimizations were taken. In this aspect, one could swap the DFA minimization algorithm for Hopcroft's Algorithm, also Python is not known to be the fastest Programming Language, thus porting the application to C++ or Java may help.

Efforts were taken to make the application multiprocessing while the scientist was updating its conjectures, since obtaining the state in which the computation of a DFA halt can be done at the same time for all words added. But the overhead associated with creating parallel processes and allocating memory for them overweight the gains.

The usage of Threads was also an option as it doesn't have the necessity to allocate memory for each usage, though, due to Python's Global Interpreter Lock, only one thread can execute code at a time. However, if the application were to be ported to C, for example, threads can be helpful to speed up the application.

The interface even though it works, it does not look appealing and modern. One could try to improve it and modernize it.

While implementing the DFA render function, we tried a few different engines, one that made the DFA more balanced and squared, in general, was `neato`, but, at last, it was not used, as some DFAs would have its edge labels overlapped. If the user wishes to experiment with different engines the following line of code can be added to the `render` function in the `dfa` class:

```
graph.attr(layout='neato')
```

Only an installer for Windows is available, thus one can compile the scripts into an executable for Linux and macOS, as well as fix memory issues associated with the Windows Installer. No Linux or macOS installer was given, since installing Python and modules in these two operating systems is much simpler compared to Windows, and, rarely, give errors.

References

[1] Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135, 1980.

[2] Lenore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28(2):125–155, 1975.

[3] John Case. Algorithmic scientific inference within our computable expected reality. *International Journal of Unconventional Computing*, 8:193–206, 01 2012.

[4] François Denis. Learning regular languages from simple positive examples. *Machine Learning*, 44:37–66, 07 2001.

[5] E Mark Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[6] John Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In Zvi Kohavi and Azaria Paz, editors, *Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.

[7] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[8] Gabriel Matos. An exhaustive algorithm for minimum state automaton identification, 2017. Departamento de Matemática, Instituto Superior Técnico.

[9] Daniel N. Osherson, Sanjay Jain, James S. Royer, and Arun Sharma. *Systems That Learn*. The MIT Press, 2 edition, 1999.

[10] Daniel N. Osherson, Michael Stob, and Scott Weinstein. *Systems That Learn*. The MIT Press, 1 edition, 1990.

[11] Rogério Reis. *Autómatos finitos: manipulação, geração e contagem*. PhD thesis, Faculdade de Ciências, Universidade do Porto, 2007.

[12] A. Sernadas, C. Sernadas, J. Rasga, and J. Ramos. *A Mathematical Primer on Computability*. College Publications, 2018.

[13] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, 2 edition, 2005.

[14] Masaru Tomita. Learning of construction of finite automata from examples using hill-climbing : Rr: Regular set recognizer. 1982.