Bachelor's Thesis Computing Science

Control Improvisation for infinite regular languages and feature-based parameters

Kevin van de Glind
s1052379

February 5, 2023

First supervisor/assessor:
Dr. Sebastian Junges

Second assessor:
Dr. Jurriaan Rot

Radboud University

Abstract

In this thesis, we will extend the Control Improvisation paradigm [4] by including a method to create improvisers which, given a infinite regular language, is able to generate infinitely many different words. Furthermore, we propose a feature-based framework, similar to [1] for two different parameters of the CI problem, which are dependent on a property of the word (this could for example be a function based on the length of a word). For these problems we give necessary and sufficient conditions concerning the feasibility of the problem and we propose an algorithm to create improvisers for a restricted family of piecewise constant functions.

Contents

Chapter 1

Introduction

The Control Improvisation problem is to construct an improviser: a probabilistic algorithm which generates words. The words which can be generated are given by means of a language specification, which is called the hard language specification in the Control Improvisation paradigm. To ensure there is enough randomness between the words an improviser creates, two additional parameters are included in the problem, such that the probability of such a word being chosen will lay in a certain interval, given by these two parameters. Lastly, one is allowed to make some families of words, given by a second language constraint, more favourable by requiring that the words generated by the improviser must be, for at least a certain fraction, part of this second family. Therefore, given these parameters and language constraints, the problem is to decide whether we can make such a distribution, and if it is possible, to create such an improviser [4].

One of the applications of Control Improvisation is fuzz testing, where we want to generate inputs which adhere to some constraints to test software [2]. These constraints could be given by language specifications in the Control Improvisation paradigm. Another application lays in the domain of robotic planning, of which an illustrative and motivating example for the use of the Control Improvisation paradigm is given in [6]. If we were to consider a security robot that follows a fixed path visiting specified landmarks, one could abuse the knowledge of the path of the robot to visit these landmarks whilst knowing that the robot would definitely not be there; adding randomness to the route of the robot will make the path of the robot less predictable and therefore it may become more difficult to time your visit of the landmark at exact times that the security robot is not present there. Adding this randomness can once again be done by using different parameters and language specifications in the Control Improvisation paradigm.

The problem was initially presented in [4]. In this paper, and in many of the generalizations and adaptations of the original Control Improvisation problem (such as [5], [6]), only improvisers over words with a length in a given finite interval are considered. Therefore, these papers were considering different constructions and generalizations of making improvisers over finitely many different words. The natural question may come to mind as to how one can generalize this to infinite languages.

In this paper, we will address this question, with language constraints to be given by regular languages, by using two different constructions:

1. We will consider problems without an additional language constraint, therefore only having a hard constraint;

2. We will include an additional language constraint, being the soft constraint in the Control Improvisation paradigm, using the construction given in 1.

Furthermore, we aim to give infinitely many words a non-zero probability, since otherwise we would be back to the original improviser problem which considers a finite subset of the original language constraint.

Originally, as presented in [4], the parameters which define the interval, in which the probability of a word being generated by the improviser lays, are constant. However, one can imagine many use cases in which very short (or very long) words are undesirable. One such case could be the generation of passwords which have to adhere to certain constraints. In this case, one may not want to get passwords with a relatively small length. Neither is it desirable to get infinitely long passwords. Therefore, we will consider the upper and lower bound for the probability of a word being dependent on the length of this word in this case. This example gives us reason to examine CI problems with upper and lower bound, for a word to be generated, to be dependent on any property of this word. Necessary and sufficient conditions for the existence of improvisers for such problems, where parameters are dependent on a property of the word, will be given for both the finite and the infinite case. Furthermore, a greedy approach will be presented to construct improvisers for finite problems of this kind. Some interesting examples of problems which turn out to be non-feasible will be given as well. These proofs show that the necessary conditions are violated, by means of showing that some series diverges. Lastly, when only a finite number of different intervals are attained, we will give the construction of an improviser.

This thesis is structured as follows: Chapter 2 contains relevant preliminary knowledge of standard automata theory and a formal introduction to the Control Improvisation problem. Chapter 3 contains the new findings on the Control Improvisation problem for constant and feature-based parameters, in that exact order.

Chapter 2

Preliminaries

In this chapter, the basic automaton theory used in this thesis will be presented. Furthermore, we give an introduction to the Control Improvisation problem and the results presented in [4].

2.1 Notation & basic automata theory

Throughout this paper, we will adhere to the standard automata notation as used in, for example, [8]. A Deterministic Finite Automaton D is a tuple $D := (Q, \Sigma, \delta, q_0, F)$, where:

- Q := The set of states in the automaton
- Σ := The set of all possible input symbols, also called the alphabet of the automaton. The alphabet is assumed to be finite.
- δ := The function $Q \times \Sigma \to Q$ which, given an input symbol and a state returns a state. This is the transition function.
- q_0 := The initial state of the DFA.
- F := The set of final states.

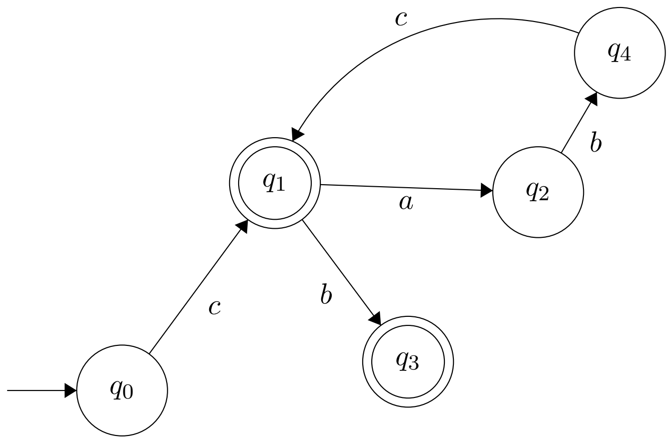
Furthermore, we define Σ^* to be the language of all words which can be made with alphabet Σ. And for a DFA D, L(D) to be all of the words accepted by D, also known as the language of D. This is formally defined as follows:

Definition 2.1.1. *Given a DFA $(Q, \Sigma, \delta, q_0, F)$, define $\delta^* : Q \times \Sigma^* \to Q$ by*

$$\delta^*(q,w) = \begin{cases} q \textit{ if } w = \lambda \\ \delta^*(\delta(q,a), w') \textit{ if } w = aw' \textit{ for some } a \in \Sigma \textit{ and } w' \in \Sigma^* \end{cases}$$

The set $L(D) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$ is called the language *of D. Furthermore, we say that D accepts a word w if $w \in L(D)$.*

Example 2.1.2. *Consider a DFA D which accepts all words given by the regular expression $c(abc)^*(1+b)$. That is, $L(D)$ is the set of words starting with a c, having arbitrarily many abc subwords after and these can either end with such a abc subword or with a b. Then D is given by the following schema:*



And we have that:

- $Q := \{q_0, ..., q_4, sink\}$
- $\Sigma := \{a, b, c\}$
- *δ returns the destination of the transition, given a state and the input symbol of an outgoing transition of said state. If a state is given in combination with a transition which is not visually represented in the visual representation, it will return the sink-state sink.*
- $F := \{q_1, q_3\}$

Note that, in the schematic representation of a DFA, we will only include the transitions which have a path to a final state for readability. And we will implicitly assume that any transition not drawn is present but goes towards a sink state.

Lastly, we define the length function and the product of two DFAs. Define

$$|\cdot| : \Sigma^* \to \mathbb{N}, |w| = \begin{cases} 0 & \textit{If } w = \lambda \\ 1 + |w'| & \textit{Otherwise, when } w = aw' \textit{ for some } a \in \Sigma \textit{ and } w' \in \Sigma^*. \end{cases}$$

be the function which gives the length of a word. Furthermore, we denote the product construction of two DFAs as follows: Let D and E be two DFAs, then the product $D \cap E$ is defined such that $L(D \cap E) := L(D) \cap L(E)$ with the formal definition given in Definition 2.1.3.

Definition 2.1.3. *Let $D = (Q_D, \Sigma, \delta_D, q_0^D, F_D)$,$E = (Q_E, \Sigma, \delta_E, q_0^E, F_E)$ be two DFAs, then $D \cap E = (Q_{D\cap E}, \Sigma, \delta_{D\cap E}, q_0^{D\cap E}, F_{D\cap E})$ is defined as follows:*

- $Q_{D\cap E} = \{(q_D, q_E) \mid q_D \in Q_D, q_E \in Q_E\}$
- $\delta_{D\cap E} : Q_{D\cap E} \times \Sigma \to Q_{D\cap E}, \quad \delta_{D\cap E}((q_0, q_1), a) = (\delta_D(q_0, a), \delta_E(q_1, a))$
- $q_0^{D\cap E} = (q_0^D, q_0^E)$
- $F_{D\cap E} = \{(q_D, q_E) \mid q_D \in F_D, q_E \in F_E\}$

2.2 Control Improvisation

Control Improvisation is a problem which given a language constraint aims to construct an improviser, an algorithm which generates words from this language specification. In this section the basic definitions and some relevant theorems to our contribution will be given.

Both the problem formalization and feasibility results shown below were introduced and proved in [4]. The CI problem is defined as follows:

Definition 2.2.1. *A CI problem instance is a tuple $C = (\mathcal{H}, \mathcal{S}, m, n, \epsilon, \lambda, \rho)$ with $\epsilon, \lambda, \rho \in [0;1]$, $\mathcal{H}, \mathcal{S}$ DFAs (called the hard- and soft language specification/constraint respectively), and $m, n \in \mathbb{N}$, we define*
$I := \{w \in L(\mathcal{H}) \mid m \leq |w| \leq n\}$ and $A := \{w \in L(\mathcal{S}) \mid w \in I\}$ to be the set of improvisations and the set of admissible improvisations. Furthermore, we call $D : \Sigma^* \to [0;1]$ an improvising distribution if it adheres to the following conditions:

1. *$P(w \in I \mid w \leftarrow D) = 1$*
2. *$P(w \in A \mid w \leftarrow D) \geq 1 - \epsilon$*
3. *$\forall w \in I, \lambda \leq D(w) \leq \rho$*

First of all, we note that the notation $w \leftarrow D$ has the following interpretation: "w is generated by improviser D".

By having requirements 1-3 in Definition 2.2.1 we make sure that an improvising distribution always generates words which are in the set of improvisations, which is ensured in requirement 1). Furthermore, requirement 2) ensures that a word, generated by the improviser, has at least a probability of $1 - \epsilon$ to be accepted by the soft constraint, which gives us the ability to prefer certain families of words. Lastly, the choice of λ, ρ gives us the ability to assign a certain amount of randomness in the improvising distribution by requirement 3). That is, by having relatively high values for λ, ρ we can ensure that a word has a higher probability of being generated, whilst a lower value of λ, ρ would enforce a lower probability of a given word to be generated. Furthermore, in favour of readability we will write $P(w)$ when we mean $P(w \leftarrow D)$.

Definition 2.2.2. *We call C feasible if there exists an improvising distribution D for C.*

We will use the notation '-' when no soft constraint is specified, or the soft constraint is trivial, which is the case when the language of the soft constraint is a superset of the set of words accepted by the hard constraint. Furthermore, we write $CI(\mathcal{H}, \mathcal{S})$ for the family of CI problems with hard language specification of type $\mathcal{H}$ and soft constraint of type $\mathcal{S}$, and similarly as above we use the notation $CI(\mathcal{H}, -)$ when we want to consider the family of CI problems with a trivial soft constraint (e.g. $(\mathcal{H}, -, 0, 50, \frac{1}{2}, \frac{1}{2}, \frac{7}{8}) \in CI(DFA, -)$).

In our case, $\mathcal{H}$ and $\mathcal{S}$ will be DFA's. But it has to be noted that Definition 2.2.1 can be extended to include other languages specifications, such as context free languages.

Thus, our task is to find out whether C is feasible, and if this is the case, to construct an improviser. An algorithm to create such a distribution is called an improvisation scheme, and it is defined as follows:

Definition 2.2.3. *Let $\mathcal{P}$ be a family of CI problems. Then we call an algorithm $\mathcal{S}$ an improvisation scheme for $\mathcal{P}$ if it has the following property:*

$$\text{For any } \mathcal{C} \in \mathcal{P}\text{: } \mathcal{S}(\mathcal{C}) = \begin{cases} \text{An improviser for } \mathcal{C}, \text{ if } \mathcal{C} \text{ is feasible} \\ \perp \text{ Otherwise} \end{cases}$$

Note that any constructive proof for an improviser of a family of CI problems $\mathcal{P}$ is a proof for the existence of an improvisation scheme for $\mathcal{P}$. Furthermore, if it does not require us to iterate over infinitely many words, such a constructive proof will be an improvisation scheme for $\mathcal{P}$.

Remark 2.2.4. *Note that ($\mathcal{H}$,$\mathcal{S}$,m,n,0,λ,ρ) with $\lambda > \rho$ will never be feasible. And neither is any such problem with $m > n$.*

In this thesis, we will use a result presented in the work of Fremont et al [4] about Control Improvisation over a finite language, which is shown below.

Theorem 2.2.5. *For any $C = (\mathcal{H}, \mathcal{S}, m, n, \epsilon, \lambda, \rho)$, the following are equivalent:*

1. *C is feasible.*

2. *The following inequalities hold:*

   (a) $\frac{1}{\rho} \leq |I| \leq \frac{1}{\lambda}$

   (b) $(1-\epsilon)\frac{1}{\rho} \leq |A|$

   (c) $|I| - |A| \leq \frac{\epsilon}{\lambda}$

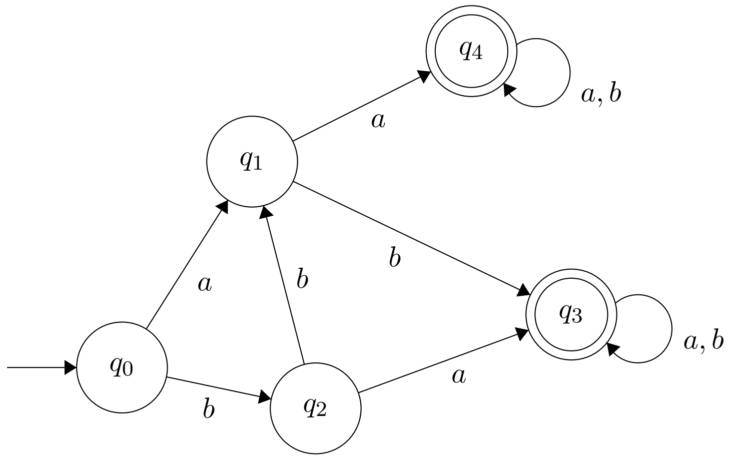
3. *There is an improviser of C.*

Theorem 2.2.5 gives us necessary and sufficient conditions to check whether or not a CI problem over a finite language is feasible.

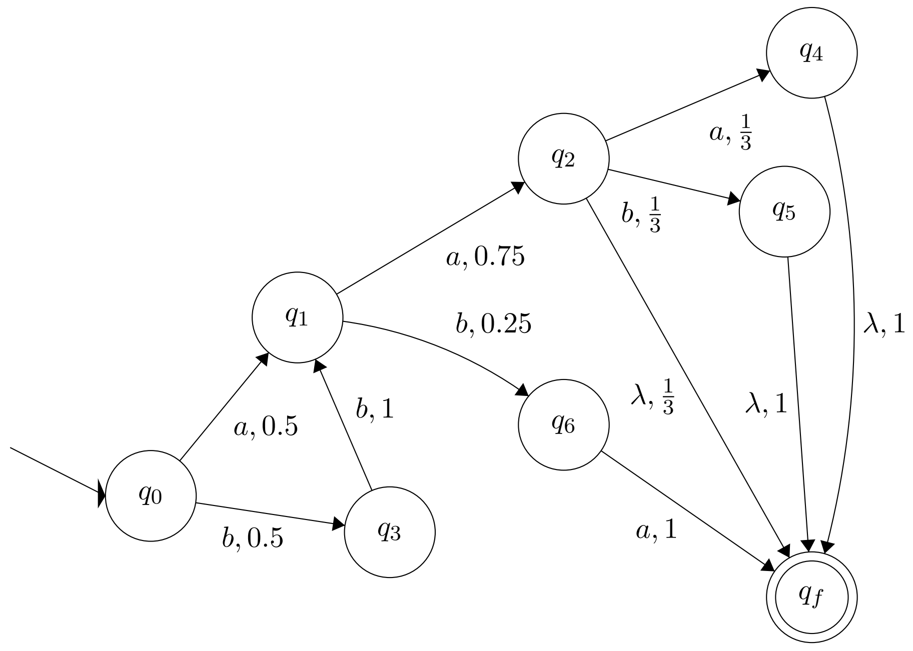
We would like to be able to generate words accepted by a DFA D, with $L(D)$ finite, such that all words in the language of D have an equal probability of being generated without the need to compute every word. The theorem which enables us to do so is presented in [7] and shown below in Theorem 2.2.6.

Theorem 2.2.6. *There exists an algorithm that, given a DFA D with L(D) finite, gives a uniform sample over L(D).*

Example 2.2.7. *Consider* $(D, -, 0, 4, \epsilon, 0.1, 0.5)$ *with D given by the following DFA:*



Then $I = \{aa, bba, aba, aaa, aab, bbba, bbab, bbaa\}$ *and therefore, by Theorem 2.2.5 we know that the problem is feasible, since* $2 \leq |I| = 8 \leq 10$. *Furthermore, using Theorem 2.2.6 we get the following improvising distribution:*



Intuitively, for every word generated by D, D will start in q_0 *and choose to take a certain edge, and therefore adding that letter to the word, with its assigned probability until it reaches the final state. Using this improvising distribution, we get for example that* $P(bbba) = \frac{1}{8}$ *and* $P(aa) = \frac{1}{8}$*. A quick inspection reveals that every word which adheres to the length bounds and is accepted by the DFA indeed has an equal probability of being generated by the improviser. Furthermore, we can clearly see that any word which was not accepted by our hard constraint, has probability 0 of being generated by the algorithm (E.g.* $P(a) = 0$*).*

Chapter 3

Research

In this section the new research of this thesis will be presented. First of all, we consider some general remarks on equivalent forms of a DFA and a characterization on when regular languages are infinite in section 3.1. After which we will focus on CI problems with infinitely large hard language specifications and λ, ρ as real values in section 3.2. Lastly, we will consider $\lambda, \rho : \Sigma^* \to [0;1]$ as functions of words in section 3.3 for problems with both finite and infinitely large language specifications. An overview of all conditions and improvisation schemes which have been proven throughout sections 3.2 and 3.3 can be found in section 3.4.

3.1 Generalizations and remarks

3.1.1 Characterization of infinite regular languages

First of all, we introduce the notion of a pruned DFA:

Definition 3.1.1. *Given a DFA D we call D' the pruned equivalent of D, if the following hold:*

1. *Every state in D' is in D.*
2. *Every transition in D' is in D.*
3. *There exists a path from q_0 to every state q in D'.*
4. *For every state q in D', either $q \in F$ or there exists a $q_f \in F$ such that there is a path from q to q_f.*

Note that pruning a DFA, which essentially means removing states and transitions which do not adhere to conditions (3) and (4), will not always result in a DFA. However, the key property of the pruned DFA which is used for the constructions of improvising distributions is the fact that there exists at most 1 transition for any character from a given state. Therefore, without loss of generality, we assume that every DFA in the upcoming sections is represented as its pruned equivalent from Lemma 3.1.3 onwards.

In order to get a characterization of the finiteness of regular languages we use Lemma 3.1.2 to get an equivalence for the existence of the maximum length of a word accepted by such a DFA and the existence of a loop in its pruned equivalent.

Lemma 3.1.2. *Given a DFA D.* $\max_{w \in L(D)}(|w|)$ *exists if and only if there is no cycle in its pruned equivalent D'.*

Proof. '$\rightarrow$' Assume, towards a contradiction, that the pruned DFA contains a cycle. Thus there is some word, which has a path through the cycle and is accepted. In that case, this cycle can be repeated an arbitrary amount of times and still reach a final state. Therefore, there is an unbounded word in the language.
'$\leftarrow$' This can be immediately seen by the fact that the length of a word is bounded by the number of edges it can traverse, which in the case of a DFA without cycle is $|Q| - 1$. □

Now that we have necessary and sufficient conditions for the existence of a maximum length of a word accepted by a DFA given by Lemma 3.1.2, we can easily see that for any DFA D, $L(D)$ is infinite if and only if $\max_{w \in L(D)}(|w|)$ does not exist, as this implies that there is a loop in the pruned DFA and therefore $L(D)$ is infinite.

3.1.2 General remarks on Control Improvisation problems over infinite regular languages

For a problem in CI(DFA, -) we use Lemma 3.1.3 to put further constraints on the feasibility of this problem, since this Lemma gives us that many Control Improvisation problems are not feasible.

Lemma 3.1.3. *Let D be a DFA with $L(D)$ infinite. If $C := (D, \mathcal{S}, m, +\infty, \epsilon, \lambda, \rho)$ is feasible, then $\lambda = 0$*

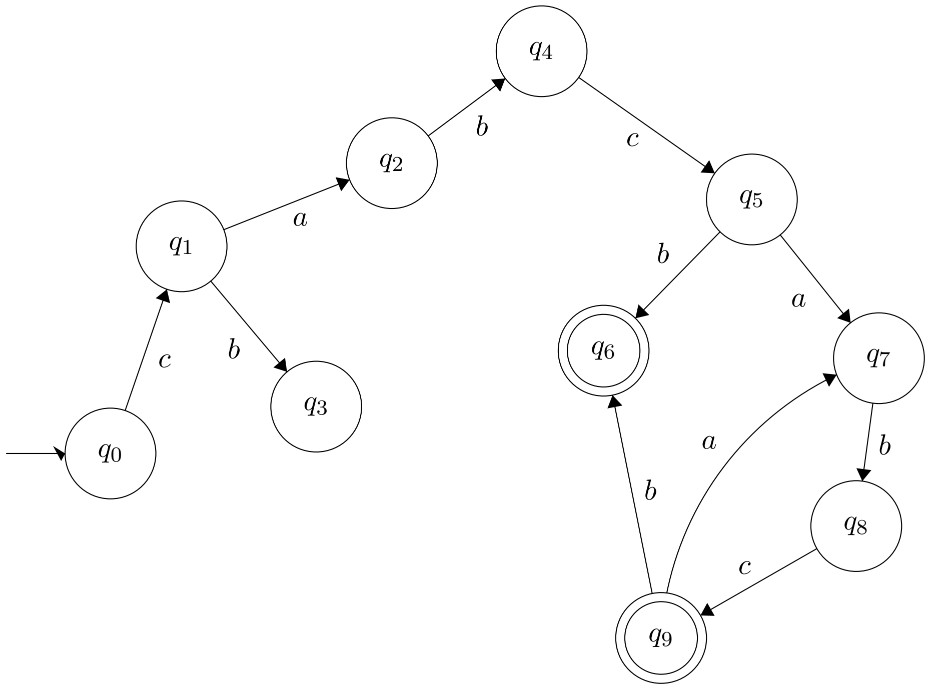
Proof. Suppose $\lambda > 0$ and assume towards contradiction that C is feasible, thus there exists some function $P : \Sigma^* \to [0; 1]$ with $P(w) \geq \lambda$ for all $w \in L(D)$, then $1 = \sum_{w \in L(D)} P(w) \geq \sum_{w \in L(D)} \lambda > 1$. Which clearly is a contradiction, thus $0 \leq \lambda \leq 0$ and we can conclude that $\lambda = 0$. □

Lemma 3.1.3 is even stronger than initially expected, as we will see in Theorem 3.2.2 that the condition $\lambda = 0$ will be the necessary and sufficient condition for the feasibility for any Control Improvisation problem in the class CI(DFA,-) with infinitely large hard constraint.

Lemma 3.1.3 already narrows the number of Control Improvisation problems we need to inspect. However, in order to construct improvisation schemes for these problems, we would like to limit the problems for which we need to construct improvisers. That is, we would like to remove some of the parameters in the Control Improvisation problem by incorporating them in the hard language constraint, such that any improviser for the changed Control Improvisation problem is an improviser for the original one as well.

In order to do this, we note that for any DFA D, we can make a DFA D' such that only words with lengths in certain bounds are accepted. This can be done by traversing every possible path in D for the first m steps, and make these in separate non-final states. After which, we can use D again. An example of which can be seen in Example 3.1.4

Example 3.1.4. *From D (as defined in Example 2.2.7), we could make D', which only accepts words of length greater than 4. D' will then be:*



From these previous results we can conclude that any problem $(D, D', m, +\infty, \epsilon, 0, \rho) \in CI(DFA, DFA)$ is feasible if and only if the equivalent problem $(\hat{D}, \hat{D}', 0, +\infty, \epsilon, 0, \rho)$ is feasible. Furthermore, since the set of improvisations and the set of admissible improvisations are the same for both problems, any improvising distribution for $(\hat{D}, \hat{D}', 0, +\infty, \epsilon, 0, \rho)$ will also be an improvising distribution for $(D, D', m, +\infty, \epsilon, 0, \rho) \in CI(DFA, DFA)$. Therefore, we only consider the construction of improvisers for Control Improvisation problems with trivial length bounds ($[m, n] = [0, +\infty]$) in order to find an improvisation scheme for all Control Improvisation problems in the class CI(DFA,-) with infinite hard constraint.

3.2 Constant ρ,λ

In this section, we will give algorithms to decide the Control Improvisation problem for Control Improvisation instances in the class CI(DFA,DFA) with an infinitely large set of improvisations. We will do this in two steps. First of all, we give an algorithm for the class of CI problems without a soft constraint. After this, a construction to decide improvising problems with a soft constraint will be given, this will use the previous result.

3.2.1 Deciding CI(DFA, -)

To make the construction of an improviser for CI(DFA,-) work, we first need to introduce a Lemma to transform DFA's into equivalent NFA's using a similar method as used in Theorem 2.2.6.

Lemma 3.2.1. *For any DFA D with $|F| = n$, there exists a NFA D' with $|F| = 1$ such that L(D) = L(D')*

Proof. To do this, we add an extra state q_f. Now that we have that state, we add a transition, for every $q \in F$, (q, q_f, λ). Now we can set $F = \{q_f\}$. Lastly, we prune D', with the foresight that this construction will be used in order to construct improvisers in Theorem 3.2.2. This obviously gives the same language. □

Theorem 3.2.2 gives an improvisation scheme for CI(DFA,-) for DFA's accepting infinitely many words. It is important to note that all such problems will be feasible as long as $\lambda = 0$.

Theorem 3.2.2. *There exists an improvisation scheme for*
$C := (D, -, 0, +\infty, \epsilon, 0, \rho) \in CI(DFA, -)$ *with L(D) infinite.*

Proof. First of all, we convert D to D' as given by Lemma 3.2.1. Let N_{q_i} be the number of outgoing transitions from $q_i \in Q$ and S be the set of transitions which are part of a cycle in D. Now, for every $q_i \in Q$, the probabilities to its outgoing transitions are assigned using the following construction:

1. Let M_{q_i} be the set of outgoing transitions from q_i with the property that after taking this transition, there is no path to a final state $t_1, ..., t_n$ with $t_i \in S$ for some $i \in [n]$.
2. For any $m \in M_{q_i}$, set $P(m) = \min(\rho; \frac{1}{N_{q_i}})$ if $|M_{q_i}| \neq N_{q_i}$. Otherwise set $P(m) = \frac{1}{|M_{q_i}|} = \frac{1}{N_{q_i}}$.
3. For any $t \notin M_{q_i}$, set $P(t) = \frac{1 - \sum_{m \in M_{q_i}} P(m)}{N_{q_i} - |M_{q_i}|}$.

Note that this construction guarantees that the sum of all outgoing edges of a state is exactly equal to 1. Furthermore, the probability of any word is at most ρ, since the path of any word must go through a transition, after which there is no path with a transition part of a loop, at least once. Since otherwise it would imply that every transition is part of a loop. However, this is in contradiction with the construction of D', since $(q, q_f, \hat{\lambda})$ is not part of a loop (because q_f does not have any outgoing transitions).

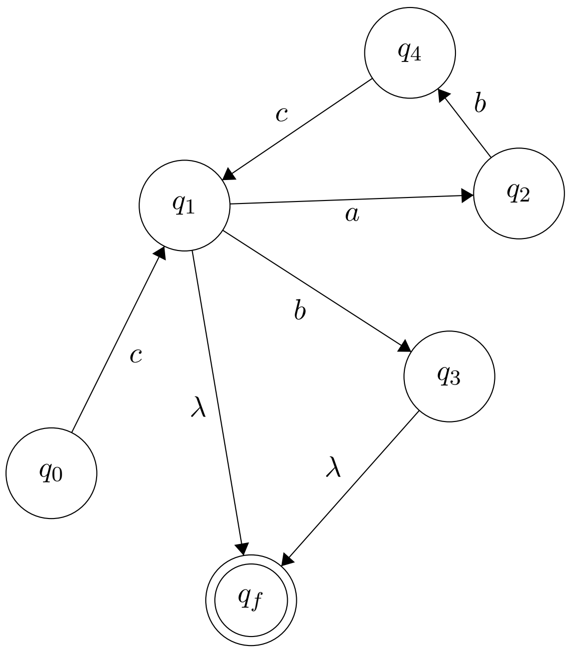
Lastly, it can not be the case that $|M_{q_i}| = N_{q_i}$ for all i, since this would imply that there are no cycles in D'. Which gives a contradiction by Lemma 3.1.2. Therefore, we can conclude that this is a correct improvising distribution for C. $\square$

The construction in Theorem 3.2.2 gives an inverse correlation between the expected length of a word and the value of ρ. Since by construction a walk through the model is more likely to choose the path which does not go through a loop again if the value of ρ is high (close to 1). Furthermore, we know that, if $\rho \geq \frac{1}{2}$, the construction above would generate words equivalently as if one were to perform a random walk through the pruned DFA.

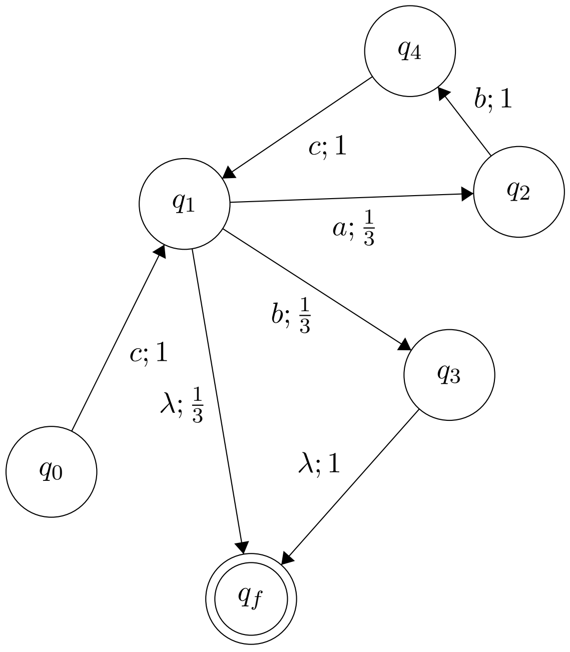
Furthermore, note that there are many possible constructions for an improvisation scheme. E.g. we could set $P(m) = \min(\frac{\rho}{|M_{q_i}|}; \frac{1}{N_{q_i}})$ for any $m \in M_{q_i}$ if $|M_{q_i}| \neq N_{q_i}$. This construction would in general lead to a higher likelihood of larger words to be generated in comparison to the method of Theorem 3.2.2. Furthermore, in this case the expected length of a word can be bounded from below by $\frac{1}{\rho}$, which is a claim that in general does not hold for the construction of Theorem 3.2.2.

In Example 3.2.3 we see a case where Theorem 3.2.2 is applied and for every state, the outgoing transitions have equal probability of being taken.

Example 3.2.3. *To demonstrate Theorem 3.2.2, we consider* $(D, -, 0, +\infty, 0, 0.1, 0.5) \in CI(DFA, -)$ *with D defined similarly as in Example 2.2.7. Lemma 3.2.1 gives us the following NFA.*



Using the same sets and values as in Theorem 3.2.2, we get the following: $N_{q_0} = N_{q_3} = N_{q_2} = N_{q_4} = 1, N_{q_1} = 3$ *and* $M_{q_i} = \emptyset$ *for all states except for* q_1 *and* q_3*, where we have that* $M_{q_1} = \{(q_1, q_f, 1), (q_1, q_3, b)\}, M_{q_3} = \{(q_3, q_f, 1)\}$ *From this we derive the improvising distribution D, using the construction as given in the proof:*



Using this result, we get, for example, that $P(c) = \frac{1}{3}$ *and* $P(cabcabcabcabcabc) = \frac{1}{3^6}$

Theorem 3.2.2 gives a construction for an improviser of any CI problem $C := (D, -, 0, +\infty, \epsilon, 0, \rho) \in CI(DFA, -)$ with L(D) infinite. Taking section 3.1 into account, which tells us that for any CI problem $(D, -, m, +\infty, \epsilon, 0, \rho)$ there exists an equivalent problem $(D', -, 0, +\infty, \epsilon, 0, \rho)$, we get that Theorem 3.2.2 gives a construction for an improviser of any CI problem $(D, -, m, +\infty, \epsilon, 0, \rho)$. Furthermore, combining the knowledge of Theorem 3.2.2 and Lemma 3.1.3, we get that $(D, -, m, +\infty, \epsilon, \lambda, \rho) \in CI(DFA, -)$ with L(D) infinitely large is feasible if and only if $\lambda = 0$.

Other than this result, Theorem 3.2.2 provides us with a construction to make an improvisation scheme for problems in CI(DFA,DFA), as can be seen in the next section.

3.2.2 Deciding CI(DFA,DFA)

In Theorem 3.2.4 a necessary and sufficient condition will be presented for the feasibility of problems in CI(DFA,DFA), these are problems which do have both a hard constraint and possibly, but not necessarily, a soft constraint. Therefore, this class is a superset of CI(DFA,-), for which we already found an improvisation scheme in Theorem 3.2.2. This condition is the feasibility of a different CI problem, which does not have a soft language constraint, due to which, by Theorem 3.2.2, we know that this condition can only fail if the hard constraint of this second problem is finite.

Theorem 3.2.4. *Assume that L(D) is infinite, then $(D, D', 0, +\infty, \epsilon, 0, \rho) \in CI(DFA, DFA)$ is feasible if and only if $(D \cap D', -, 0, +\infty, \epsilon, 0, \frac{\rho}{1-\epsilon})$ is feasible.*

Proof. '$\rightarrow$' Suppose that $(D, D', 0, +\infty, \epsilon, 0, \rho)$ is feasible. Furthermore, assume, towards a contradiction, that $(D \cap D', -, 0, +\infty, \epsilon, 0, \frac{\rho}{1-\epsilon})$ is not feasible, then $L(D \cap D')$ is finite. And this implies that $\frac{1-\epsilon}{\rho} > |L(D \cap D')|$ as given by Theorem 2.2.5, which implies that
$\sum_{w \in L(D \cap D')} P(w) \leq |L(D \cap D')|\rho < 1 - \epsilon$ and this is a clear contradiction.
'$\leftarrow$' Suppose that $(D \cap D', -, 0, +\infty, \epsilon, 0, \frac{\rho}{1-\epsilon})$ is feasible. We make a case distinction for the size of $L(D \backslash D')$:

- If $L(D \backslash D')$ is infinite: Then $(D \backslash D', 0, +\infty, \epsilon, 0, \frac{\rho}{\epsilon})$ is feasible and we can define the following improvising distribution for $(D, D', 0, +\infty, \epsilon, 0, \rho)$:

$$P(w) = \begin{cases} (1-\epsilon)\bar{P}(w) & \textit{if } w \in L(D \cap D') \\ \epsilon \hat{P}(w) & \textit{otherwise.} \end{cases}$$

  Where $\bar{P}, \hat{P}$ are derived from the improvising distributions received from respectively $(D \cap D', -, 0, +\infty, \epsilon, 0, \frac{\rho}{1-\epsilon})$ and $(D \backslash D', -, 0, +\infty, \epsilon, 0, \frac{\rho}{\epsilon})$. This gives a correct improvising distribution, since:

1. $\sum_{w \in L(D)} P(w) = \sum_{w \in L(D \cap D')} P(w) + \sum_{w \in L(D \backslash D')} P(w)$
   $= (1-\epsilon) \sum_{w \in L(D \cap D')} \bar{P}(w) + \epsilon \sum_{w \in L(D \backslash D')} \hat{P}(w) = 1 - \epsilon + \epsilon = 1$
2. $\sum_{w \in L(D \cap D')} P(w) = (1-\epsilon) \sum_{w \in L(D \cap D')} \bar{P}(w) = 1 - \epsilon$
3. (a) For any $w \in L(D \cap D')$: $P(w) = (1-\epsilon)\bar{P}(w) \leq (1-\epsilon)\frac{\rho}{1-\epsilon} = \rho$
   (b) For any $w \in L(D \backslash D')$: $P(w) = \epsilon \hat{P}(w) \leq \epsilon \frac{\rho}{\epsilon} = \rho$

- If $L(D \backslash D')$ is finite: Then that implies that $L(D \cap D')$ infinite, since $L(D \cap D') \cup L(D \backslash D') = L(D)$ which is infinite by assumption. If $|L(D \backslash D')|\rho \geq \epsilon$ we can construct the following distribution:

$$P(w) = \begin{cases} (1-\epsilon)\bar{P}(w) & \textit{if } w \in L(D \cap D') \\ \epsilon \frac{1}{|L(D \backslash D')|} & \textit{otherwise.} \end{cases}$$

Which is a correct improvising distribution, since:

1. $\sum_{w \in L(D)} P(w) = \sum_{w \in L(D \cap D')} P(w) + \sum_{w \in L(D \backslash D')} P(w)$
   $= 1 - \epsilon + \epsilon \sum_{w \in L(D \backslash D')} \frac{1}{|L(D \cap D')|} = 1$
2. $\sum_{w \in L(D \cap D')} P(w) = (1-\epsilon) \sum_{w \in L(D \cap D')} \bar{P}(w) = 1 - \epsilon$
3. (a) For any $w \in L(D \cap D')$: $P(w) = (1-\epsilon)\bar{P}(w) \leq (1-\epsilon)\frac{\rho}{1-\epsilon} = \rho$
   (b) For any $w \in L(D \backslash D')$: $P(w) \leq \epsilon \frac{\rho}{\epsilon} = \rho$

NB $\epsilon \frac{1}{|L(D \backslash D')|}$ can be interpreted as: "With probability ϵ, generate a word which is accepted by the DFA $D \backslash D'$ uniformly". To generate this word, we can use Theorem 2.2.6, since $L(D \backslash D')$ is finite.

Otherwise, if $|L(D \backslash D')|\rho < \epsilon$ we create a new CI problem $(D \cap D', -, 0, +\infty, \epsilon, 0, \frac{\rho}{1-|L(D \backslash D')|\rho})$, which is feasible since, as mentioned previously, $|L(D \cap D')|$ is infinite. Now we can create the following distribution:

$$P(w) = \begin{cases} (1 - |L(D \backslash D')|\rho)\bar{P}(w) & \textit{if } w \in L(D \cap D') \\ \rho \quad \textit{otherwise.} \end{cases}$$

Which is a correct improvising distribution, since:

1. $\sum_{w \in L(D)} P(w)$
   $= \sum_{w \in L(D \cap D')} P(w) + \sum_{w \in L(D \backslash D')} P(w)$
   $= 1 - |L(D \backslash D')|\rho + \sum_{w \in L(D \backslash D')} \rho$
   $= 1 - |L(D \backslash D')|\rho + |L(D \backslash D')|\rho = 1$
2. $\sum_{w \in L(D \cap D')} P(w) = (1 - |L(D \backslash D')|\rho) \sum_{w \in L(D \cap D')} \bar{P}(w)$
   $= 1 - |L(D \backslash D')|\rho \geq 1 - \epsilon$
3. (a) For any $w \in L(D \cap D')$: $P(w) = (1 - |L(D \backslash D')|\rho)\bar{P}(w) \leq \rho$

(b) For any $w \in L(D \backslash D')$: $P(w) = \rho$

NB $\rho = |L(D \backslash D')| \rho \frac{1}{|L(D \backslash D')|}$ and therefore that part of the function definition can be interpreted as: "With probability $|L(D \backslash D')|\rho$, generate a word which is accepted by the DFA $D \backslash D'$ uniformly". To generate this word, we can use Theorem 2.2.6, since $L(D \backslash D')$ is finite.

And using the constructions above, we have shown that there always exists an improvising distribution for $(D, D', 0, +\infty, \epsilon, 0, \rho)$, thus we can conclude that it is feasible. $\square$

With a similar argument as used in section 3.2.1, Theorem 3.2.4 gives that any CI problem $(D, D', m, +\infty, \epsilon, \lambda, \rho) \in CI(DFA, DFA)$ is feasible if and only if $\lambda = 0$ and $(D \cap D', -, 0, +\infty, \epsilon, \lambda, \frac{\rho}{1-\epsilon})$ is feasible. Furthermore, if a CI problem is feasible, Theorem 3.2.4 gives the construction of an improviser. Therefore, Theorem 3.2.4 in combination with the conditions above form an improvisation scheme.

3.3 λ,ρ as functions

In this section we will consider $\rho, \lambda : \Sigma^* \to [0; 1]$ as a function based on a property of the word. This could, for example, be the number of occurrences of a certain substring in a word or the length of a word. In order to increase readability in the upcoming sections, we introduce the following notation:

Definition 3.3.1. *We denote the set of words with length n accepted by the DFA D by $L_n(D) := L(D) \cap \Sigma^n = \{w \in L(D) \mid |w| = n\}$.*

Unfortunately, but expectedly, we need stronger assumptions to make conclusions about the feasibility of a problem, when considering ρ, λ as functions, in comparison to Theorem 2.2.5.

3.3.1 Deciding CI(DFA,-)

In this section, the necessary and sufficient condition will be given for the feasibility of problems in the class of CI(DFA,-) with λ, ρ as functions. Furthermore, when the set of improvisations is finite, we will show the construction of an improvisation scheme. Lastly, for the infinite case we make several observations with respect to the feasibility of some families of Control Improvisation instances as well as improvisation schemes for others.

Theorem 3.3.2. *Given $C := (D, -, 0, +\infty, \lambda, \rho, \epsilon) \in CI(DFA, -)$ with $\lambda, \rho : \Sigma^* \to [0; 1]$. Then C is feasible if and only if the following inequalities hold:*

1. $\forall w \in L(D) : \lambda(w) \leq \rho(w)$

2. $\sum_{w\in L(D)} \lambda(w) \leq 1$

3. $\sum_{w\in L(D)} \rho(w) \geq 1$

Proof. '$\rightarrow$' Assume, towards a contradiction, that $\sum_{w\in L(D)} \lambda(w) > 1$, then for any function P with the property that $P(w) \geq \lambda(w)$ we have that $1 < \sum_{w\in L(D)} \lambda(w) \leq \sum_{w\in L(D)} P(w)$. Thus any such function would not be an improvising distribution. Therefore, we can conclude that C is not feasible, which gives the required contradiction.

Concerning the other inequality, we once again assume towards a contradiction that $\sum_{w\in L(D)} \rho(w) < 1$, then for any function P with the property that $P(w) \leq \rho(w)$, we have that $1 > \sum_{w\in L(D)} \rho(w) \geq \sum_{w\in L(D)} P(w)$. Thus any such function would not be an improvising distribution either. Therefore, we can conclude that C is not feasible, which gives the required contradiction.

Lastly, if we were to assume towards a contradiction that there exists a word w in L(D) such that $\lambda(w) > \rho(w)$ we have by definition that the problem can not be feasible. And we can conclude that the inequalities must hold.
'$\leftarrow$' Suppose that the inequalities above hold. Then we have the following algorithm to get an improvising distribution, using a similar approach as in the greedy strategy used in [6]:

1. For every $w \in L(D)$, set $P(w) = \lambda(w)$
2. Iterate over the words w_0 until $\sum_{w\in L(D)} P(w) = 1$:
   Set $P(w_0) \leftarrow \min(\rho(w_0), 1 - \sum_{w\in L(D)\setminus\{w_0\}} P(w))$

This will definitely terminate in the finite case, since there must exist values such that the sum of probabilities is equal to 1. Furthermore, it describes a well constructed improvising distribution in the infinite case. To conclude, we proved existence of an improvising distribution and this implies that C is feasible. $\square$

Note that Theorem 3.3.2 does not depend on L(D) being finite and thus holds for infinite languages as well. However, the construction of the improvising distribution is not an algorithm for infinite languages, since it requires, in the worst case, to iterate over all lengths (which means that it would not terminate in certain cases). This worst case could occur when, for example, $\lambda = 0$ and $\rho(w) = \frac{2^{-(|w|+1)}}{|L_{|w|}(D)|}$ such that $1 = \sum_{k\geq 0} |L_k(D)|\rho(k)$. Therefore, this construction only proves existence when L(D) is infinite.

Example 3.3.3. *Let us consider an example of the non-feasibility of a large family of problems where λ is given by a function which may be of interest. Suppose that D is a DFA with $L(D)$ infinite.*

Then, $(D, -, 0, +\infty, \lambda(w) := \frac{c}{|w|}, \rho, \epsilon) \in CI(DFA, -)$, for any $c > 0$, can not be feasible. By Lemma 3.1.2 there must be a cycle in the DFA, thus there exists some word w without using the cycle, which is accepted by D and passes through a state which has the cycle, lets call the length of this cycle k. Therefore, for any function $P : \Sigma^* \to [0; 1]$ which has the property that $P(w) \geq \lambda(w)$ we have that $\sum_{w \in L(D)} P(w) \geq \sum_{n \geq 1} \frac{c}{|w|+nk}$
$\geq c \sum_{n \geq 1} \frac{1}{n(|w|+k)} = \frac{c}{|w|+k} \sum_{n \geq 1} \frac{1}{n}$. And this is a divergent series, therefore the problem can not be feasible.

Furthermore, as a direct corollary, one can see that
$(D, -, 0, +\infty, \lambda \in \Omega(\frac{1}{|w|}), \rho, \epsilon) \in CI(DFA, -)$ can not be feasible either.

To get some actual improvising distributions for infinite languages, we will consider ρ, λ to be in certain families of functions. First of all, let us cover an easy corollary heavily inspired on Theorem 3.2.2.

Lemma 3.3.4. *There exists an algorithm to decide*
$(D, -, 0, +\infty, 0, \rho, \epsilon) \in CI(DFA, -)$ with $\rho(w) \geq c > 0$ for all $w \in \Sigma^*$.

Proof. By theorem 3.2.2, the CI problem $(D, -, 0, +\infty, 0, c, \epsilon)$ is feasible and the improviser acquired from this problem is an improviser for $(D, -, 0, +\infty, 0, \rho, \epsilon)$ as well. $\square$

However, this still does not give us any freedom over the choice of λ. Therefore, we consider a different family of functions in Lemma 3.3.5.

Lemma 3.3.5. *Given a DFA D, assume $L(D)$ is infinite and*
$\lambda, \rho : \Sigma^* \to [0; 1]$. Then, if there exists an $a \in (0; 1)$ and a bijection
$f : L(D) \to \mathbb{N}$ such that:

$$\lambda(w) \leq a^{f(w)}(1-a),\ \rho(w) \geq a^{f(w)}(1-a) \text{ for all } w \in L(D)$$

Then $(D, -, 0, +\infty, \lambda, \rho, \epsilon) \in CI(DFA, -)$ is feasible.

Proof. Let a, f be as required. Then, define $P(w) = a^{f(w)}(1-a)$, this gives a correct improvising distribution, since it is by construction greater/ lesser than or equal to respectively $\lambda(w), \rho(w)$ and
$\sum_{w \in L(D)} P(w) = \sum_{w \in L(D)} a^{f(w)}(1-a) = \sum_{n \geq 0} a^n(1-a) = \frac{1-a}{1-a} = 1$ $\square$

Note that a bijection $f : L(D) \to \mathbb{N}$ always exists, since, by assumption of Lemma 3.3.5 L(D) is infinite. However, one has to note that, if there exists a bijective function for which Lemma 3.3.5 holds, this does not necessarily mean that it works for any bijective function f, as can be seen in the following example.

Example 3.3.6. *Consider $(D, -, 0, +\infty, \lambda, \rho, \epsilon) \in CI(DFA, -)$ with $L(D) = L(a^*)$ and λ, ρ defined as follows:*

$$\rho(w) = \begin{cases} \frac{1}{20} \text{ If } |w| = 0 \\ 1 \text{ Otherwise} \end{cases} \quad \lambda(w) = \begin{cases} \frac{1}{5} \text{ If } |w| = 1 \\ 0 \text{ Otherwise} \end{cases}$$

In this case, when the function $w \mapsto |w|$ is used (which is clearly bijective), there is no $a \in (0;1)$ such that ρ majorizes, and λ is majorized by, $a^n(1-a)$. However, this problem is clearly feasible and using the following bijection:

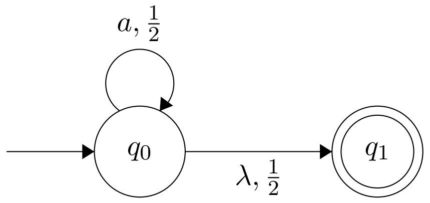
$$f = \begin{cases} \lambda \mapsto 1 \\ a \mapsto 0 \\ a^k \mapsto k \end{cases}$$

We get that, for example, $a = 10^{-3}$ would suffice.

Furthermore, note that the constraints in Lemma 3.3.5 are not necessary conditions, as one can easily construct a counterexample for this case. E.g. let $\lambda \equiv 0$ and $\rho(w) = 1$ for all $w \in L(D)$ with $|w| \in \{0;1\}$ and $\rho(w) = \frac{1}{2}^{|w|^2+1}$ otherwise. Note that this problem is feasible in very many cases, for example when the hard constraint is given by $L(D) = L(a^*)$, in which case we can set $P(a) = 1$ and $P(w) = 0$ for all other words. But there does not exist a value for a and a bijection f such that ρ majorizes $a^{f(w)}(1-a)$, since ρ is strictly decreasing and therefore there can only exist a bijection f for which ρ majorizes $a^{f(w)}(1-a)$, if ρ majorizes $a^{|w|}(1-a)$. However, this is not the case since a has to be chosen independently of $|w|$ and ρ decreases in the limit significantly faster than $a^{|w|}(1-a)$.

Using Lemma 3.3.5 we can create improvisation schemes for these families of functions. However, this does require us to know, or to be able to compute, a value for a and a bijective function beforehand. On the other hand, we could consider a different example, where λ, ρ are based on the length of words.

Example 3.3.7. *Let us consider $L(D) = L(a^* + b^*)$ and the following CI problem $(D, -, 0, +\infty, \epsilon, \frac{1}{2^{|w|+9}}, \frac{1}{2^{|w|+1}})$. Then we can create the following improviser: Generate a word w and its length $|w| = n$ in the following improviser (of an arbitrary problem $(a^*, -, 0, +\infty, \epsilon, 0, \frac{1}{2})$):*



Now we can sample uniformly from $L_n(D)$, which gives an improvising distribution. Since for any word, the probability of being sampled is $P(w) = \frac{1}{2^{|w|+1}} \frac{1}{|L_n(D)|} = \frac{1}{2^{|w|+2}}$. Lastly, we also have that: $\sum_{w \in L(D)} P(w) = \sum_{n \geq 0} \sum_{w \in L_n(D)} P(w) = \sum_{n \geq 0} \frac{1}{2^{n+1}} = 1$. And therefore we can conclude that this is indeed a correct improvising distribution.

3.3.2 Deciding CI(DFA,DFA)

Similarly to Theorem 3.3.2 we need stronger necessary assumptions to guarantee the feasibility of instances in the class CI(DFA,DFA) when we consider ρ, λ as functions, this will be displayed in the proof below:

Theorem 3.3.8. *Let $\lambda, \rho : \Sigma^* \to [0;1]$,*
then $(D, D', 0, +\infty, \epsilon, \lambda, \rho) =: C \in CI(DFA, DFA)$ is feasible if and only if the following inequalities hold:

1. $\forall w \in L(D) : \lambda(w) \leq \rho(w)$
2. $\sum_{w \in L(D)} \lambda(w) \leq 1$
3. $\sum_{w \in L(D)} \rho(w) \geq 1$
4. $\sum_{w \in L(D \cap D')} \rho(w) \geq 1 - \epsilon$
5. $\sum_{w \in L(D \backslash D')} \lambda(w) \leq \epsilon$

Proof. '$\rightarrow$' Suppose that C is feasible, then there exists an improvising distribution P and we have that:

1. Inequality (1) holds by definition of the feasibility of the CI problem.
2. $\sum_{w \in L(D)} \lambda(w) \leq \sum_{w \in L(D)} P(w) = 1$
3. $1 = \sum_{w \in L(D)} P(w) \leq \sum_{w \in L(D)} \rho(w)$
4. $1 - \epsilon \leq \sum_{w \in L(D \cap D')} P(w)$
   $\leq \sum_{w \in L(D \cap D')} \rho(w)$
5. $\sum_{w \in L(D \backslash D')} \lambda(w)$
   $\leq \sum_{w \in L(D \backslash D')} P(w)$
   $= \sum_{w \in L(D)} P(w) - \sum_{w \in L(D \cap D')} P(w)$
   $\leq \sum_{w \in L(D)} P(w) - (1 - \epsilon) = 1 - (1 - \epsilon) = \epsilon$

'$\leftarrow$' Suppose that the inequalities above hold. First of all, we consider the case that $\sum_{w \in L(D \backslash D')} \rho(w) < \epsilon$. For any $w \in L(D \backslash D')$ we set $P(w) = \rho(w)$. Since $L(D \cap D') \subset L(D)$ we get that:
$\sum_{w \in L(D \cap D')} \lambda(w) \leq \sum_{w \in L(D)} \lambda(w) \leq 1$.

Furthermore, since $\sum_{w \in L(D)} \rho(w) \geq 1$ we know that there exists some assignment of probabilities to words $w \in L(D \cap D')$ such that $\sum_{w \in L(D)} P(w) = 1$, $\lambda(w) \leq P(w) \leq \rho(w)$ for all w in $L(D)$ and $\sum_{w \in L(D \cap D')} P(w) = 1 - \sum_{w \in L(D \backslash D')} P(w) = 1 - \sum_{w \in L(D \backslash D')} \rho(w) \geq 1 - \epsilon$. Therefore P gives the probabilities assigned to generating any word and describes our improvising distribution.

In case that $\sum_{w \in L(D \backslash D')} \rho(w) \geq \epsilon$, we once again have a function P such that $\lambda(w) \leq P(w) \leq \rho(w)$ for all w in $L(D \cap D')$, and in this case

$\sum_{w \in L(D \cap D')} P(w) \geq 1 - \epsilon$ and $\sum_{w \in L(D \cap D')} P(w) \leq 1$ hold (where we aim for $\sum_{w \in L(D \cap D')} P(w) = 1 - \epsilon$, which is possible if $\sum_{w \in L(D \cap D')} \lambda(w) \leq 1 - \epsilon$, otherwise we set $P(w) = \lambda(w)$), since we still have that
$\sum_{w \in L(D \cap D')} \lambda(w) \leq \sum_{w \in L(D)} \lambda(w) \leq 1$.

Lastly, we make another case distinction over $\sum_{w \in L(D \cap D')} \lambda(w)$:

- In case that $\sum_{w \in L(D \cap D')} \lambda(w) \leq 1 - \epsilon$ (Therefore making $\sum_{w \in L(D \cap D')} P(w) = 1 - \epsilon$), the assumption $\sum_{w \in L(D \backslash D')} \lambda(w) \leq \epsilon$ can be used to assign probabilities to words w in $L(D \backslash D')$ with the property:
$\sum_{w \in L(D \backslash D')} P(w) = \epsilon, \lambda(w) \leq P(w) \leq \rho(w)$. Finally we have that $\sum_{w \in L(D)} P(w) = \sum_{w \in L(D \cap D')} P(w) + \sum_{w \in L(D \backslash D')} P(w) = 1 - \epsilon + \epsilon = 1$. Thus in this case we get a correct improvising distribution.

- Otherwise, we have that $\sum_{w \in L(D \cap D')} \lambda(w) > 1 - \epsilon$ (Therefore defining $P(w) = \lambda(w)$ for all w in $L(D \cap D')$), we get that:
$\sum_{w \in L(D \backslash D')} \lambda(w)$
$= \sum_{w \in L(D)} \lambda(w) - \sum_{w \in L(D \cap D')} \lambda(w)$
$= \sum_{w \in L(D)} \lambda(w) - \sum_{w \in L(D \cap D')} P(w)$
$\leq 1 - \sum_{w \in L(D \cap D')} P(w)$
It has already been shown that $\sum_{w \in L(D \backslash D')} \rho(w) \geq \epsilon \geq 1 - \sum_{w \in L(D \cap D')} P(w)$.
Thus we can assign probabilities to all words w in $L(D \backslash D')$ such that $\lambda(w) \leq P(w) \leq \rho(w)$ and $\sum_{w \in L(D \backslash D')} P(w) = 1 - \sum_{w \in L(D \cap D')} P(w)$.
Thus this once again gives a correct improvising distribution.

To conclude, for every case we were able to prove the existence of an improviser. Therefore, we can conclude that the problem is feasible. □

Note that Theorem 3.3.8 gives necessary and sufficient conditions for the feasibility of problems in class CI(DFA,DFA) when we consider λ, ρ as functions. However, we do not give a construction for such an improviser for every CI instance, as this could require us to iterate over infinitely many words. Furthermore, similar to Theorem 3.3.2 these conditions do not depend on the language of the hard constraint to be infinite. Even stronger, in case that the language of the hard constraint is finite, the proof above gives us a method to assign probabilities to every word of being generated by an improvising distribution (with a similar algorithm as given in Theorem 3.3.2), therefore this proof provides an, although very inefficient, improvisation scheme for these cases.

3.3.3 λ, ρ as finite regular piecewise constant functions

In this section, we will consider λ, ρ to be part of a restricted family of piecewise constant functions, which we call fRPC as given in Definition 3.3.9.

Definition 3.3.9. *Let Y be a non-empty set. Let $f : \Sigma^* \to Y$ for some alphabet Σ. We say that f is finite regular piecewise constant (fRPC(Y)) if* *there exists a finite partition $P = \{L(D_1), ..., L(D_{k+1})\}$ of Σ^* such that for any $L(D_i) \in P$ we have that $L(D_i)$ is a regular language and $|f(L(D_i))| = 1$.*

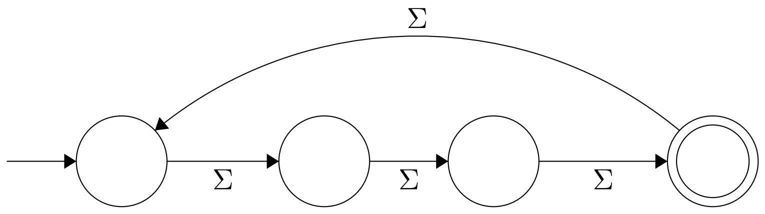
Therefore, if λ, ρ are fRPC([0;1]), then there exist partitions P_λ, P_ρ such that λ and ρ attain exactly one value in any of the elements of their respective partitions. Furthermore, we can create another finite partition, which is a refinement of P_λ, P_ρ, as follows:

$$P = \bigcup_{L(D_i) \in P_\lambda,\ L(D_j) \in P_\rho} \{L(D_i) \cap L(D_j)\}$$

Note that this partition is once again finite, since both P_λ, P_ρ are finite, and every element of P is a regular language, since regular languages are closed under intersection. Furthermore, by construction of P, for any $L(D_i) \in P$ we have that $|\lambda(L(D_i))| = |\rho(L(D_i))| = 1$.

In order to increase readability we write, with abuse of notation, $D_i \in P$ to reference the DFA which generates $L(D_i)$. Furthermore, we write λ_i, ρ_i for the value which is attained by respectively λ and ρ for all words $w \in L(D_i)$.

Remark 3.3.10. *An example of such a partition of Σ^* is given by two regular languages, where one of which contains all words with length $|w| \equiv 3$ mod 4. This language $\{w \in \Sigma^* \mid \exists k \in \mathbb{N},\ |w| = 3 + 4k\}$ is given by the following DFA:*



Therefore, the set containing this regular language as well as its complement is a finite partition of Σ^*.

Lemma 3.3.11 gives an improvisation scheme for the class of problems CI(DFA,-) with λ, ρ fRPC([0;1]). Furthermore, we only cover the instances where the language of the hard constraint is infinite, since we have already provided an improvisation scheme for the finite case in Theorem 3.3.2.

Lemma 3.3.11. *Let λ, ρ be fRPC([0;1]) and $L(D)$ infinite, then there exists an improvisation scheme for $(D, -, 0, +\infty, \epsilon, \lambda, \rho) \in CI(DFA, -)$, under the assumption that we know the partition P.*

Proof. Consider a control improvisation problem $(D, -, 0, +\infty, \epsilon, \lambda, \rho) \in CI(DFA, -)$ with λ, ρ fRPC([0;1]) and partition P. Let $P'_n, P'_\infty \subset P$ such that $L(D \cap D_i)$ is finite, but not zero, for every element in $D_i \in P'_n$ and infinitely large for every $D_i \in P'_\infty$. Note that P'_∞

is not empty, since this would imply that L(D) would be finite, which is a contradiction with our assumption.

As an initial check, for every i with $D_i \in P'_\infty$, λ_i must be equal to zero, which can be easily seen with a similar argument as used in Lemma 3.1.3. Furthermore, the following inequality must hold:
$\sum_{D_i \in P'_n} |L(D \cap D_i)|\lambda_i \leq 1$, otherwise we can once again conclude that the problem is not feasible (with a similar argument as mentioned in Theorem 3.3.2). If the inequality does hold however, we can assign $P(w) = \lambda_i$ for all $w \in \bigcup_{D_i \in P'_n} L(D \cap D_i)$. If $\sum_{D_i \in P'_n} |L(D \cap D_i)|\lambda_i = 1$ we are done and could set the probability of all other words to 0. Moreover, we can not hope to give infinitely many words a non-zero probability. However, if $\sum_{D_i \in P'_n} |L(D \cap D_i)|\lambda_i < 1$, we set, for $w \in L(D \cap D_i)$ with $D_i \in P'_\infty$, $P(w) = \frac{1}{|P'_\infty|}(1 - \sum_{D_i \in P'_n} |L(D \cap D_i)|\lambda_i)\hat{P}(w)$ where $\hat{P}$ is derived from the improvising distribution of the problem $(D \cap D_i, -, 0, +\infty, \epsilon, 0, \rho_i)$. This is a correct construction, since we know by Theorem 3.2.2 that this problem will be feasible. Moreover, the same Theorem gives us a construction for such a distribution. □

Once again we can extend the improvisation scheme to include a soft language constraint, this is done in Theorem 3.3.12.

Theorem 3.3.12. *Let λ, ρ be fRPC([0;1]) and L(D) infinite, then there exists an improvisation scheme for $(D, D', 0, +\infty, \epsilon, \lambda, \rho) \in CI(DFA, DFA)$, under the assumption that we know the partition P.*

Proof. Consider a control improvisation problem $(D, D', 0, +\infty, \epsilon, \lambda, \rho) \in CI(DFA, DFA)$ where, once again, λ, ρ are fRPC([0;1]) with partition P and L(D) infinite. Now we define the following subsets of P:

$$L(D_i) \in P_n^{D'} \leftrightarrow 0 < |L(D \cap D') \cap L(D_i)| < +\infty$$

$$L(D_i) \in P_n \leftrightarrow 0 < |L(D \backslash D') \cap L(D_i)| < +\infty$$

$$L(D_i) \in P_\infty^{D'} \leftrightarrow 0 < |L(D \cap D') \cap L(D_i)| = +\infty$$

$$L(D_i) \in P_\infty \leftrightarrow 0 < |L(D \backslash D') \cap L(D_i)| = +\infty$$

Note that these sets are disjoint and the union of the sets once again is equal to L(D).

Once again, if the problem were to be feasible, then, for every i such that $D_i \in P_\infty \cup P_\infty^{D'}$, λ_i must be equal to zero, which can be easily seem with a similar argument as used in Lemma 3.1.3. Furthermore, the following inequality must hold $\sum_{D_i \in P_n} |L(D \backslash D' \cap D_i)|\lambda_i + \sum_{D_i \in P_n^{D'}} |L(D \cap D' \cap D_i)|\lambda_i \leq 1$, otherwise we can once again conclude that the problem is not feasible, since this would imply that the minimum probability that we have to assign to a subset of words is already greater than 1. Furthermore, we must require that $\sum_{D_i \in P_n} |L(D \backslash D' \cap D_i)|\lambda_i \leq \epsilon$, such that we are guaranteed to have enough room to give the admissible words a minimum total probability of $1-\epsilon$.

Lastly, if $P_\infty^{D'} = \emptyset$ then we must have that $\sum_{D_i \in P_n^{D'}} |L(D \cap D' \cap D_i)|\rho_i \geq 1-\epsilon$ to satisfy the constraint on the soft specification of the CI problem. If all of the inequalities above hold, then we can conclude that the problem is feasible and we create the following improvising distribution:

- If $P_\infty^{D'} = \emptyset$, then $P_\infty \neq \emptyset$ since L(D) is infinitely large by assumption. Therefore, for any $w \in L(D \cap D' \cap D_i)$ with $D_i \in P_n^{D'}$ we set $P(w) = \max(\min(\rho_i, (1-\epsilon)\frac{1}{|P_n^{D'}|\sum_{D_i \in P_n^{D'}} |L(D \cap D' \cap D_i)|}), \lambda_i)$ such that $\sum_{w \in L(D \cap D')} P(w) \geq 1-\epsilon$. Furthermore, we define $P(w) = \lambda_i$ for any w in $L(D \backslash D' \cap D_i)$ with $D_i \in P_n$.

  Lastly, for any $w \in L(D \backslash D' \cap D_i)$ with $D_i \in P_\infty$, $P(w) = \frac{1}{|P_\infty|}(1 - \sum_{w \in L(D \cap D' \cap D_i): D_i \in P_n^{D'}} P(w) - \sum_{w \in L(D \backslash D' \cap D_i): D_i \in P_n} P(w))\hat{P}(w)$ where $\hat{P}(w)$ is constructed by the CI problem $(D \backslash D' \cap D_i, 0, +\infty, \epsilon, 0, \rho_i)$ which is feasible and can be constructed by Theorem 3.2.2.

- If $P_\infty = \emptyset$, then $P_\infty^{D'} \neq \emptyset$ since L(D) is infinitely large by assumption. Therefore, for any $w \in L(D \cap D' \cap D_i)$ with $D_i \in P_n^{D'}$ we set $P(w) = \lambda_i$, we do the same for any $w \in L(D \backslash D' \cap D_i)$ with $D_i \in P_n$. Lastly, for any $w \in L(D \cap D' \cap D_i)$ with $D_i \in P_\infty^{D'}$ we set $P(w) = \frac{1}{|P_\infty^{D'}|}(1 - \sum_{w \in L(D \cap D' \cap D_i): D_i \in P_n^{D'}} P(w) - \sum_{w \in L(D \backslash D' \cap D_i): D_i \in P_n} P(w))\hat{P}(w)$ where $\hat{P}(w)$ is constructed by the CI problem $(D \cap D' \cap D_i, 0, +\infty, \epsilon, 0, \rho_i)$ which is feasible, and we can construct an improvising distribution, by Theorem 3.2.2.

- If $P_\infty \neq \emptyset$ and also $P_\infty^{D'} \neq \emptyset$, then, once again define, for any $w \in L(D \cap D' \cap D_i)$ with $D_i \in P_n^{D'}$, $P(w) = \lambda_i$, we do the same for any $w \in L(D \backslash D' \cap D_i)$ with $D_i \in P_n$. Furthermore to get a non-zero probability for infinitely many other words, we use the following construction: For any $w \in L(D \cap D' \cap D_i)$ with $D_i \in P_\infty^{D'}$ we set $P(w) = \frac{1}{|P_\infty^{D'}|}(\max(0, 1-\epsilon - \sum_{w \in L(D \cap D' \cap D_i): D_i \in P_n^{D'}} P(w)))\hat{P}(w)$ where $\hat{P}(w)$ is constructed by the CI problem $(D \cap D' \cap D_i, 0, +\infty, \epsilon, 0, \rho_i)$. This ensures that $1 \geq \sum_{w \in L(D \cap D')} P(w) \geq 1-\epsilon$.

  Lastly, we define similarly, for any $w \in L(D \backslash D' \cap D_i)$ with $D_i \in P_\infty$, $P(w) = \frac{1}{|P_\infty|}(1 - \sum_{w \in L(D \cap D')} P(w) - \sum_{w \in L(D \backslash D' \cap D_i): D_i \in P_n} P(w))\hat{P}(w)$ where $\hat{P}(w)$ is constructed by the CI problem $(D \backslash D' \cap D_i, 0, +\infty, \epsilon, 0, \rho_i)$.

Since this process will correctly give an improvising distribution, when a problem is feasible, we can conclude that this is a correct improvisation scheme. $\square$

Unlike Theorem 3.3.8, Theorem 3.3.12 has a constructive proof for which we do not have to explicitly generate infinitely many words and therefore gives an improvisation scheme.

Furthermore, note that the condition that we know P is not a very strict condition, since λ, ρ could very well be given with their respective partitions, with which we can construct the refinement of their partitions using the method above.

3.4 Overview

Throughout this thesis, we have considered many different classes of CI problems, for some of which we were able to find constructive proofs, and therefore actual improvisation schemes. Whilst for others we have only shown the necessary and sufficient conditions for the feasibility of such problems. These results are summarized in the following two tables. Note that Theorem 2.2.5 has been proven in the research of Fremont et al [4].

$L(D)$ finite			
λ, ρ:	CI Class	Feasibility conditions	Improvisation scheme
λ, ρ real valued	CI(DFA,-)	Theorem 2.2.5	Theorem 2.2.5
	CI(DFA,DFA)	Theorem 2.2.5	Theorem 2.2.5
λ, ρ fRPC([0;1])	CI(DFA,-)	Theorem 3.3.2	Theorem 3.3.2
	CI(DFA,DFA)	Theorem 3.3.8	Theorem 3.3.8
$\lambda, \rho : \Sigma^* \to [0;1]$	CI(DFA,-)	Theorem 3.3.2	Theorem 3.3.2
	CI(DFA,DFA)	Theorem 3.3.8	Theorem 3.3.8

$L(D)$ infinite			
λ, ρ:	CI Class	Feasibility conditions	Improvisation scheme
λ, ρ real valued	CI(DFA,-)	Lemma 3.1.3	Theorem 3.2.2
	CI(DFA,DFA)	Theorem 3.2.4	Theorem 3.2.4
λ, ρ fRPC([0;1])	CI(DFA,-)	Lemma 3.3.11	Lemma 3.3.11
	CI(DFA,DFA)	Theorem 3.3.12	Theorem 3.3.12
$\lambda, \rho : \Sigma^* \to [0;1]$	CI(DFA,-)	Theorem 3.3.2	Open problem
	CI(DFA,DFA)	Theorem 3.3.8	Open problem

Chapter 4

Related Work

Control improvisation was first proposed as a concept to generate random sequences of music by Donze et al [3]. This idea was formalized by Fremont et al [4], where a construction for an improviser and feasibility conditions for finite problems were introduced (See preliminaries 2 for the relevant results for this research). A multitude of adaptations and generalizations have been made based on this research. Such as the research by Gittis et al [6], where assigning costs for words was introduced. Furthermore, the research on Reactive Control Improvisation by Fremont et al [5] modeled two-player games in the Control Improvisation paradigm. All of these have in common that they are focused around finite subsets of possibly infinitely large languages. Therefore, Theorem 3.2.4 extends the theory as defined in [4] and has opportunities to be applied in the different generalizations as well.

Furthermore, the consideration of λ, ρ as functions based on properties of words in a language is an extension of research done by Boneschanscher [1], where feature constraint functions $f_c : \Sigma^* \to \mathbb{Q} \cap [0; 1]$ were introduced. Once again, finite sets of words were considered in this case, which gets extended to infinite regular languages in this research. Furthermore, we extend the theory presented by both giving necessary and sufficient requirements for feasibility of such problems and considering the lower and upper bound as functions based on a feature of the word instead of only the upper bound.

Chapter 5

Conclusions

5.1 Findings

The main contributions of this thesis are given in Theorems 3.2.4, 3.3.12 and 3.3.8 where we propose a construction, whilst considering infinite languages, for an improvisation scheme when λ, ρ are constant or of class fRPC([0;1]) and give necessary and sufficient conditions for the feasibility of CI problems when $\lambda, \rho : \Sigma^* \to [0; 1]$ respectively. Furthermore, when we consider λ, ρ as functions based on some property of the words in the language, we made several observations concerning the feasibility and constructions of improvisers for different families of functions. However, the conditions in this case may be very challenging to check.

5.2 Future work

For future research, it would be interesting to find an improvisation scheme for any CI problem over regular languages with λ, ρ as functions based on some property of the words in the language. However, we suspect that this would be quite challenging and therefore one could focus on a different family of feature based parameters, such as parameters described by unary weighted automata and/or absorbing markov chains. These models could possibly be described as functions using their Jordan Normal Form decomposition. However, the check for feasibility may not be as easy as one hopes for. That is, Theorem 3.3.2 and Theorem 3.3.8 can not be easily checked, since the series limit is not known in general.

Secondly, one could be interested in finding (more) efficient improvisation schemes for the CI problems, as in this thesis we did not investigate the complexity of our proposed improvisation schemes. There is still a big improvement to be made as it is for example clearly undesirable to generate all words in cases as Theorem 3.3.8 and 3.3.2, even in the finite case.

Bibliography

[1] Stefan Boneschanscher. *Feature-based Randomness Constraints in Control Improvisation*. Bachelor's thesis, Radboud University, 2022.

[2] Chen Chen, Baojiang Cui, Jinxin Ma, Runpu Wu, Jianchao Guo, and Wenqian Liu. A systematic review of fuzzing techniques. *Computers & Security*, 75:118–137, 2018.

[3] Alexandre Donzé, Rafael Valle, Ilge Akkaya, Sophie Libkind, Sanjit A Seshia, and David Wessel. *Machine improvisation with formal specifications*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 2014.

[4] Daniel J Fremont, Alexandre Donzé, and Sanjit A Seshia. Control improvisation. 2017.

[5] Daniel J Fremont and Sanjit A Seshia. Reactive control improvisation. In *International conference on computer aided verification*, pages 307–326. Springer, 2018.

[6] Andreas Gittis, Eric Vin, and Daniel J Fremont. Randomized synthesis for diversity and cost constraints with control improvisation. In *Computer Aided Verification: 34th International Conference, CAV 2022, Haifa, Israel, August 7–10, 2022, Proceedings, Part II*, pages 526–546. Springer, 2022.

[7] Timothy Hickey and Jacques Cohen. Uniform random generation of strings in a context-free language. *SIAM Journal on Computing*, 12(4):645–655, 1983.

[8] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Introduction to Automata Theory, Languages, and Computation*, chapter 3.2. Pearson, 2001.