ENERGY CONSUMPTION OF PROBABILISTIC AND QUANTUM FINITE STATE VERIFIERS

by
Özdeniz Dolu
B.S., Computer Engineering, Boğaziçi University, 2021
B.S., Mathematics, Boğaziçi University, 2021

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2024

ACKNOWLEDGEMENTS

I thank Prof. A. C. Cem Say for introducing me to the field of theoretical computer science, for being a teacher and for being my advisor throughout the process of writing this thesis. I thank my peers and colleagues Nevzat Ersoy, Utkan Gezer and Fırat Kıyak for both social and intellectual support they have provided through this process. Throughout my master's studies, I was financially supported by TUBITAK through 2210/A National MSc/MA Scholarship Program.

ABSTRACT

ENERGY CONSUMPTION OF PROBABILISTIC AND QUANTUM FINITE STATE VERIFIERS

Irreversible steps in computation are associated with an erasure of information and thus, an increase in the entropy. In the case of a physical computer, this increase in entropy is associated with energy dissipation, although modern computers are nowhere near this lower bound and generate heat mostly due to other reasons. This work explores the reversibility properties of non-interactive verification systems with finite memory whose verifiers do quantum, deterministic or probabilistic computation. Definitions made in this work provide a framework in which questions about reversibility can be asked and answered. A method that eliminates some types of irreversibilities in probabilistic finite state verification systems is provided. Moreover, every type of irreversibility in such systems are conjectured to be eliminated as a conclusion.

ÖZET

OLASILIKSAL VE KUANTUM SONLU DURUMLU DENETMENLERİN ENERJİ TÜKETİMİ

Hesaplamadaki tersinebilir olmayan adımlar bilginin silinmesiyle ilişkilidir ve sonucunda entropi artışına neden olur. Fiziksel bilgisayarlarda bu entropi artışı aynı zamanda enerji tüketimiyle ilişkilidir. Modern bilgisayarlar, birçok farklı sebepten bu entropi artışı sebebiyle açığa çıkması gereken enerji alt sınırının çok daha üzerinde ısı açığa çıkarır. Bu çalışmada sonlu hafızaya sahip interaktif olmayan deterministik ya da kuantum denetim sistemlerinin tersinebilirlik özellikleri keşfedildi. Bu çalışmada yapılan tanımlar tersinebilirlik üzerine sorulan soruların cevaplanabileceği bir çerçeve sunar. Olasılıksal sonlu durumlu denetim sistemlerinde tersinebilirliği bozan belli durumlardan kurtulmayı sağlayan bir metod gösterildi. Bunun da ötesinde, sonuç olarak, aynı tür sistemlerde tersinebilirliği bozan tüm durumların giderilebileceği öne sürüldü.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS

C	Configuration
d	Head movement
L	Language
p	Proof string
Q	Set of states
q	State
r	Random sequence
R	Random tape alphabet / Irreversibility region
M	Machine
V	Verifier
x	Input string
α	Complex amplitude
Γ	Proof tape alphabet
γ	Proof tape symbol
δ	Transition function
ϵ	Error
ρ	Random tape symbol
Σ	Input tape alphabet
σ	Input tape symbol
Ω	Set of auxiliary states
$\times$	Cartesian product
$\otimes$	Tensor product
$\|\rangle$	Bra-ket notation
$\triangleleft$	Right-end marker
$\triangleright$	Left-end marker

LIST OF ACRONYMS/ABBREVIATIONS

1NFA	One-way Non-deterministic Finite Automata
1NRFA	One-way Non-deterministic Reversible Finite Automata
1NSNFA	One-way Non-Stop Non-deterministic Reversible Finite Automata
CONST	Constant
DFA	Deterministic Finite Automata
DFA-V	Deterministic Finite Automaton Verifier
MA	Merlin Arthur
NFA	One-way Non-deterministic Finite Automata
PFA	Probabilistic Finite Automata
PFA-V	Probabilistic Finite Automaton Verifier
REG	Set of Regular Languages
RFA	Reversible Finite Automata
RT	Real-time
QFA	Quantum Finite Automata
QFA-V	Quantum Finite Automaton Verifier

1. INTRODUCTION

Reversibility in computation, which is an interesting topic in the theoretical computer science in and of itself, has real physical implications in terms of the minimum amount of energy that needs to be dissipated by the computer while carrying out its computations. Landauer's Principle [1], states that logically irreversible steps in the computation has a thermodynamic cost associated with them. Every time some information is forgotten by the system, meaning its past has become ambiguous in some sense, entropy increase associated with this operation results in energy dissipation. This puts a lower-bound on the energy dissipation associated with a computer, depending on how irreversible its computations are designed to be. However, in [2], Bennett showed that given an arbitrary Turing Machine, it is possible to construct a three-tape Turing Machine that generates the same output but its computation is logically reversible while not leaving any "clutter" or unnecessary things written on its tape. Also, the space and time complexity associated with this simulation is not intolerable. This means that, at the level of universal computation of Turing Machines, reversible computation is not an insurmountable task. Reversible computation also has implications for quantum computation as unitary matrices have to be invertible. This fact leads to a direct connection with results about reversible computation and zero-error quantum computation such as in the [3, 4].

When we turn our attention to smaller, weaker machines, such as deterministic finite automata there are more clear distinctions. It is known that the set of languages recognized by reversible finite automata (rfa in short) is a smaller set than the set of regular languages. Moreover, there are hierarchies in REG such that any dfa that recognize a language in this hierarchy need to use increasing amount of "irreversibility" in its computation [4], meaning that it needs to forget more and more information as we go up in the hierarchy.

This work, mainly focuses on analyzing the reversibility properties of a particular type of proof system where an all-knowing Prover provides a membership proof for an input string to a Verifier who is a finite state automaton that is either deterministic or probabilistic. In the probabilistic case, the Prover is not allowed to know the results of the random coin tosses, this allows the Verifier to check the validity of the proof in a stronger manner. In Section 2, definitions of these systems are given along with the definition of the reversibility we are employing in this study. In Section 3, results are discussed. Section 3 focuses mainly on a proof system whose Verifier is only allowed to move right in the input tape, allowed to stay still or move right in the proof tape and only allowed to toss a constant number of coins on any input. Such verifiers are actually some of the weakest that are still able to verify non-regular languages and this makes them a point of interest. When reversibility is not taken into account, capabilities of such systems are explored in [5, 6]. In Section 3, it is shown that some type of irreversibilities in such systems can be eliminated using a general algorithm and it is also conjectured that all type of irreversibilities can be eliminated.

Proof systems and their corresponding verification procedures have real life implications as the world is getting ever more digital secure communication is getting ever more important. It is possible that a low-energy protocol, in a very real "physically lowest possible" sense, might prove to be helpful in the long run.

2. DEFINITIONS AND PRELIMINARIES

2.1. Verification Systems

Interactive proof systems are a family of abstract computational models that generally consist of a 2 way communication between a prover and a verifier. Both the prover and the verifier are generally assumed to be variants of Turing machines that can be subjected to constraints on their time and space complexities. Depending on how the system is defined, there can be more than one prover or more than one verifier. The verifier is tasked with checking the "membership" of an input string in light of its communication with the prover. Consequently, the prover is tasked with proving, through their communication, to the verifier that the given input string is a member of the language. Results such as IP = PSPACE [7] or MIP* = RE [8] have shown the extent of the language verification power of such systems.

Computational models that are investigated in this work are a particular family of non-interactive verification systems. Here, the keyword "non-interactive" signifies the fact that there is no ongoing communication between the prover and the verifier during the computation. Instead, before the computation of the verifier starts, the prover supplies the verifier with a single proof string in a proof tape after which there is no other interaction between the two parties. In the family of systems that are being considered in this work, provers are all powerful and the verifiers are constrained to be various finite state automata. These systems work in a similar manner to the MA protocol.

As for the naming convention throughout this document, verification systems will be characterized by the verifier. This is because in all cases, the provers are assumed to be all powerful machines that supply their proof before verification starts. Two main types of verifiers are considered: deterministic finite automata and probabilistic finite automata.

General notation we use for a verification system will be xfa-v(constraint 1, constraint 2, ...) where,

- x is the type of the computation that the verifier does. x is either d, p or q. They stand for deterministic, probabilistic and quantum respectively.
- There may be various categories of constraints imposed upon the verifier such as:
  - Input head movement. Potential values: rt-input, 1way-input, 2way-input
  - Proof head movement. Potential values: rt-proof, 1way-proof, 2way-proof
  - Access to random bits (for probabilistic machines). const-rand, poly-rand, log-rand
- When the name is uppercase, such as DFA-V(rt-input, 1way-proof), this refers to the class of languages that can be verified by the verifiers defined under given constraints.

To prevent any confusion: in the literature, when talking about tape head movement restrictions, the keyword 1way is sometimes used to mean that the tape head is only allowed to go right, but not allowed to stay in place. However, here, that restriction is called real-time ("rt" in short) and the restriction "1way" allows the tape head to stay in place.

2.1.1. Deterministic Verifier

A dfa-v V is a 6-tuple $\{Q, \Sigma, \Gamma, \delta, q_0, Q_H\}$ where,

(i) Q : A finite set of states.

(ii) Σ : Input alphabet where $\Sigma_{\triangleright\triangleleft}$ denotes the set $\Sigma \cup \{\triangleright, \triangleleft\}$ (input alphabet with left and right end markers).

(iii) Γ : Proof alphabet where $\Gamma_{\triangleright\triangleleft}$ denotes the set $\Gamma \cup \{\triangleright, \triangleleft\}$ (proof alphabet with left and right end markers).

(iv) δ : Transition function.

(v) q_0 : Initial state.

(vi) Q_H : Set of halting states (a subset of Q) where Q_H itself is the union of Q_A and Q_R (set of accepting and rejecting states respectively). Q_{NH} denotes the set of non-halting states (states that are in $Q \setminus Q_H$).

In its most general form, transition function δ of V is defined as

$$\delta : Q \times \Sigma_{\triangleright\triangleleft} \times \Gamma_{\triangleright\triangleleft} \to Q \times \{-1, 0, 1\} \times \{-1, 0, 1\}. \tag{2.1}$$

$\delta(q_s, \sigma, \gamma) = (q_t, d_x, d_p)$ if and only if, upon reading the input symbol σ and proof symbol γ, V will change its state from q_s to q_t and move its input and proof head in the direction d_x and d_p.

If we have rt-input or rt-proof constraint on the verifier V, we will omit using their corresponding head movement parameters as they will always be 1.

On input x and proof p, M halts upon reaching a state $q \in Q_H$. If $q \in Q_A$, we say M accepts x, otherwise (if $q \in Q_R$), we say M rejects x.

At the beginning of computation, V starts in the state q_0 and input and proof head is assumed to be at the left end marker. We assume both the input and proof string is extended with left and right end markers $\triangleright$ and $\triangleleft$. Computation is followed then, step by step, according to the transition function.

We say that a dfa-v V verifies a language L when

- for all $x \in L$, there exists a $p \in \Gamma^*$ such that V reaches an accepting state when supplied with x and p in its input and proof tapes respectively.
- For all $x \notin L$ and for all $p \in \Gamma^*$, V reaches a rejecting state when supplied with x and p in its input and proof tapes respectively.

2.1.2. Probabilistic Verifier

A pfa-v V is a 7-tuple $\{Q, \Sigma, \Gamma, R, \delta, q_0, Q_H\}$ where,

(i) Q : A finite set of states.

(ii) Σ : Input alphabet where $\Sigma_{\triangleright\triangleleft}$ denotes the set $\Sigma \cup \{\triangleright, \triangleleft\}$ (input alphabet with left and right end markers).

(iii) Γ : Proof alphabet where $\Gamma_{\triangleright\triangleleft}$ denotes the set $\Gamma \cup \{\triangleright, \triangleleft\}$ (proof alphabet with left and right end markers).

(iv) R : Random string alphabet. Unless stated otherwise, assumed to be $\{0, 1\}$.

(v) δ : Transition function.

(vi) q_0 : Initial state.

(vii) Q_H : Set of halting states (a subset of Q) where Q_H itself is the union of Q_A and Q_R (set of accepting and rejecting states respectively). Q_{NH} denotes the set of non-halting states (states that are in $Q \setminus Q_H$).

In its most general form, transition function δ of V is defined as

$$\delta : Q \times \Sigma \times \Gamma \times R \to Q \times \{-1, 0, 1\} \times \{-1, 0, 1\} \times \{0, 1\} \tag{2.2}$$

and $\delta(q_s, \sigma, \gamma, r) = (q_t, d_x, d_p, d_r)$ if and only if, upon reading the input symbol σ, proof symbol γ and random symbol r, M will change its state from q_s to q_t and move its input, proof and random tape heads in the directions d_x, d_p and d_r respectively.

Here, in every instance of computation, V is assumed to be supplied with an infinite random string generated uniformly from random alphabet. Although this is not the ordinary way to incorporate randomness, defining it in this manner will be useful later in discussions of reversibility. V is only allowed to stop or move forward in this tape (meaning that this tape always have 1way constraint). When randomness is considered as a resource, "moving right" on this tape will correspond to consuming a single coin. Contents of the random string are assumed to be hidden from the prover.

If we have rt-input or rt-proof constraint on the machine V, we will omit using their corresponding head movement parameters as they will always be 1.

On input x and proof p, V halts upon reaching a state $q \in Q_H$. If $q \in Q_A$, we say V accepts x, otherwise (if $q \in Q_R$), we say V rejects x.

At the beginning of computation, V starts in the state q_0 and input and proof head is assumed to be at the left end marker. We assume both the input and proof string is extended with left and right end markers $\triangleright$ and $\triangleleft$. Computation is followed then, step by step, according to the transition function.

We use one-sided error in our language verification criteria. We say that a pfa-v V verifies a language L with error $0 < \epsilon < 1$ when

- for all $x \in L$ there exists a $p \in \Gamma^*$ such that V reaches an accepting state for every random tape content, when supplied with x and p in its input and proof tapes respectively, with probability 1.
- For all $x \notin L$, for every random tape content and for all $p \in \Gamma^*$, V reaches a rejecting state when supplied with x and p in its input and proof tapes respectively, with probability at least $1 - \epsilon$.

2.1.3. Quantum Verifier

A qfa-v(1way-input, 1way-proof) M is a 7-tuple $\{Q, \Sigma, \Gamma, \delta, q_1, F, \Omega\}$ where,

- Q is a finite set of states.
- Σ is the input alphabet.
- Γ is the proof alphabet.
- δ is the transition function.
- q_1 is the initial state.
- F is the set of accepting states(a subset of Q).

- Ω is a finite set of auxiliary states($\Omega \cap Q = \emptyset$).

More about this notation and concepts can be summarized as follows:

- The symbol $C_{n,m}$ denotes the set of classical configurations of M for an input of length n and a proof of length m.
- A classical configuration (defined on input of length n and proof of length m) $c \in C_{n,m}$ where $|c\rangle = |q_i\rangle \otimes |k_x\rangle \otimes |k_p\rangle \otimes |\omega_j\rangle = |q_i, k_x, k_p, \omega_j\rangle$ is both a unit and a basis vector. Where $q_i \in Q$ is the state, $k_x \in \{0, 1, 2, ..., n\}$ is input head position, $k_p \in \{0, 1, 2, ..., m\}$ is proof head position and $w_j \in \Omega$ is the auxiliary state.
- For an input of length n and proof of length m, any superposition $|\Psi\rangle$ of M is of the form $|\Psi\rangle = \sum_{c \in C_{n,m}} \alpha_c |c\rangle$ where $\alpha_c \in \mathbb{C}$ and l_2 norm of this vector is equal to 1.
- The coefficients α_c are called amplitudes and they are complex numbers.

Transition function δ of M is defined as

$$\delta : Q \times \Sigma \times \Gamma \times \Omega \times \{0,1\} \times \{0,1\} \times Q \to \mathbb{C} \tag{2.3}$$

and the starting auxiliary state is not included because before every computational state, M measures its auxiliary state in its computational basis and then rotates it back to w_1. The values $\delta(q_s, \sigma, \gamma, \omega_j, d_x, d_p, q_t)$ correspond to the amplitudes with which, upon reading the input symbol σ and proof symbol γ, M will change its state from q_s to q_t, change its auxiliary state from w_1 to w_j and move its input and proof tape head in the directions d_x and d_p.

Another way to look at this is, for each input string x and proof string p, δ induces an operator $U_{x,p}$ on the set of valid configurations of M as

$$U_{x,p} |q_s, k_i, k_p, w_1\rangle = \sum_{\substack{q_t \in Q \\ \omega_j \in \Omega \\ d_x \in \{0,1\} \\ d_p \in \{0,1\}}} \delta(q_s, \sigma, \gamma, \omega_j, d_x, d_p, q_t) |q_t, k_i + d_x, k_p + d_p, w_j\rangle \tag{2.4}$$

and we can represent $U_{x,p}$ as a matrix for every input x and proof p. Since the auxiliary state is set to w_1 at the beginning of every step, only the first $|Q||x||p|$ columns of this

matrix is of interest. Rest can be filled to ensure unitarity. We will subdivide this matrix as $|Q||x||p|$ by $|Q||x||p|$ square blocks as

$$U_{x,p} = \left[\begin{array}{c|c} E_{x,p,1} & \cdots \\ \hline E_{x,p,2} & \cdots \\ \hline \vdots & \ddots \\ \hline E_{x,p,|\Omega|} & \cdots \end{array}\right] \tag{2.5}$$

and the machine M is said to be well formed if $U_{x,p}$ is unitary [9]. Since we can fill the remaining columns of the matrix in a way that keeps unitarity, this condition is equal to

$$\sum_{w_i \in \Omega} E^{\dagger}_{x,p,i} E_{x,p,i} = I. \tag{2.6}$$

If M is not well-formed, either there are two columns in $U_{x,c}$ (that correspond to C_i and C_j) that have non-zero entries in the same row or l_2 norm of a column is not equal to 1. It is easy to make sure that the well-formedness doesn't break due to the latter. So, let's focus on the former. This condition means that δ of M maps two distinct configurations to the same configuration with non-zero amplitude. This can happen in a unitary way but if we make sure that it never happens, we can be sure that well-formedness is preserved. It should be noted that, by not letting such superpositions, it might be possible we lose quantum advantage.

In only 1 computational step, M can move its input and proof head at most 1 to the right. Therefore, cases where two distinct configurations reach to the same configuration are limited. For a configuration C, there are 4 different possible 'past configurations' that can reach that configuration in 1 step for every state $q \in Q$ (auxiliary states are not of concern because we set them to ω_1 at every step.). General case can be described as

$$C_x = |q_x, k_{i_x}, k_{m_x}, \omega_1\rangle\,, \tag{2.7}$$

$$C_y = |q_y, k_{i_y}, k_{m_y}, \omega_1\rangle\,, \tag{2.8}$$

$$C_t = |q_t, k_{i_t}, k_{m_t}, \omega_{j_t}\rangle\,, \tag{2.9}$$

$$\delta(q_x, \sigma, \gamma, \omega_{j_t}, d_{x_x}, d_{m_x}, q_t) \neq 0, \tag{2.10}$$

$$\delta(q_y, \sigma, \gamma, \omega_{j_t}, d_{x_y}, d_{m_y}, q_t) \neq 0, \tag{2.11}$$

$$k_{i_x} + d_{x_x} = k_{i_t}, \tag{2.12}$$

$$k_{m_x} + d_{m_x} = k_{m_t}, \tag{2.13}$$

$$k_{i_y} + d_{x_y} = k_{i_t}, \tag{2.14}$$

$$k_{m_y} + d_{m_y} = k_{m_t} \ and \tag{2.15}$$

$$C_x \neq C_y. \tag{2.16}$$

Now, adopting a method described in [3], we can make sure that case described above can never happen. If for every $q \in Q$, we make sure that for any state to transition into q, only a particular head movement should be executed (formally, if there exists a function $D : Q \to \{0,1\} \times \{0,1\}$) then the case above can happen if and only if $C_x = C_y$. If we design our machine M with a function D like this, we can describe the transitions for each symbol pair (σ, γ) and auxiliary state in a simpler matrix $V_{\sigma,\gamma,\omega_j}$ and as long as such matrices are unitary, we can be sure that the machine M is well formed. Entries of $V_{\sigma,\gamma,\omega_j}$ are defined as(i^{th} row, j^{th} column)

$$V^{i,j}_{\sigma,\gamma,\omega_j} = \delta(q_j, \sigma, \gamma, \omega_j, D(q_j)[0], D(q_j)[1], q_i). \tag{2.17}$$

2.2. Reversibility

Definition of reversibility employed in this work can be considered as an extension to the one employed in the definition of a reversible finite automaton (rfa for short) and will only be made for the classical case since no results have been put forward for the quantum case. Let us start from the definition of (real-time) deterministic finite automaton (dfa for short). A dfa M is a 6-tuple $\{Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej}\}$ where,

(i) Q is the set of states.

(ii) Σ is the input alphabet

(iii) δ is the transition function.

(iv) q_0 is the initial states.

(v) Q_{acc} is the set of accepting states.

(vi) Q_{rej} is the set of rejecting states.

Transition function δ is defined as

$$\delta : Q \times \Sigma \to Q \tag{2.18}$$

where M moves right in the input tape and transitions into state $\delta(q, \sigma)$ upon reading the symbol σ in the case it is in state q. M starts its computation from initial state and at the first symbol of the input string. Upon reaching a state from Q_{acc} or Q_{rej}, M accepts or rejects respectively.

We say that M is a reversible finite automaton in the case where two transitions $\delta(q_1, \sigma) = q'$ and $\delta(q_2, \sigma) = q'$ are not allowed to exist if $q_1 \neq q_2$. In other words, no two transitions coming out of different states, reading the same symbol are allowed into the same state. To extend this definition into more complicated systems such as pfa-vs, it is helpful to consider configuration trees (using the yields relation) that are induced on a specific input, proof, random tape content. This definition of reversibility can be interpreted as no configuration having more than one predecessor in that tree [3].

Although this definition is straightforward in the case of rfas, when we introduce more tapes and more movement types to those tape heads, reversibility becomes more complicated to keep track of. This is simply due to there being many more ways to reach the same configuration from different predecessor configurations. Well-formedness conditions given for two-way quantum finite state automata in [3] is a good example for those complications, as unitarity conditions for the transition matrices also imply reversibility.

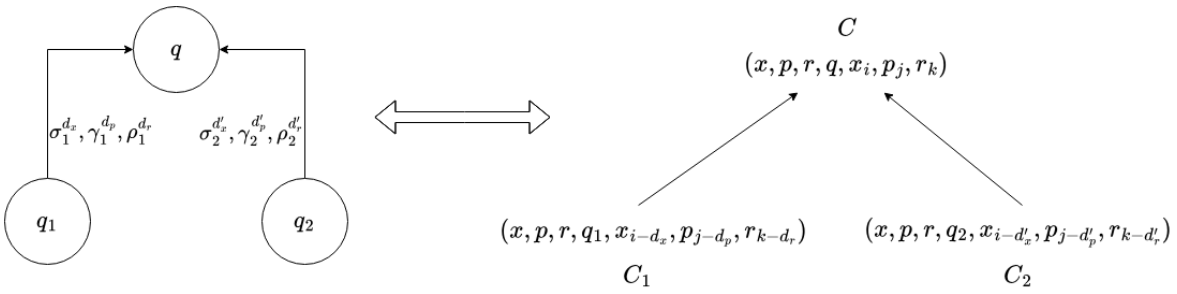

Figure 2.1. Two transitions and corresponding two configurations.

At any point in a computation, configuration of a classical verifier(pfa-v or dfa-v) V consists of its tape contents, tape head positions and its state. Let us consider the most general case. Let a pfa-v V be in the configuration $C = (x, p, r, q, x_i, p_j, r_k)$ where

- $x \in \Sigma^*$ is the input string.
- $p \in \Gamma^*$ is the proof string.
- r is an infinite sequence of random symbols from R.
- $q \in Q$ is the state
- i is the input head position where $x_i \in \Sigma$ is the i^{th} symbol of the input.
- j is the proof head position where $p_j \in \Gamma$ is the j^{th} symbol of the proof.
- k is the random head position where $r_k \in R$ is the k^{th} symbol of the random sequence.

Considering the transitions (into C)

$$\delta(q_1, \sigma_1, \gamma_1, \rho_1) = (q, d_x, d_p, d_r) \text{ and} \tag{2.19}$$

$$\delta(q_2, \sigma_2, \gamma_2, \rho_2) = (q, d'_x, d'_p, d'_r) \tag{2.20}$$

as well as considering the time reversal, these two transitions respectively imply the predecessor configurations

$$C_1 = (x, p, r, q_1, x_{i-d_x}, p_{j-d_p}, r_{k-d_r}) \text{ and} \tag{2.21}$$

$$C_2 = (x, p, r, q_2, x_{i-d'_x}, p_{j-d'_p}, r_{k-d'_r}). \tag{2.22}$$

Situation outlined here is summarized in Figure 2.1. Note that, in the left part of the figure, superscripts in the labels of the arrows denote the head movements corresponding to the related tape symbol. Here, if both C_1 and C_2 are valid predecessors to C, we say that these two transitions are introducing irreversibility. There are various head movement combinations to consider here.

If all of the head movements match (i.e. $d_x = d'_x$, $d_p = d'_p$ and $d_r = d'_r$), then both C_1 and C_2 are acting upon reading the same input, proof and random tape positions. If $q_1 = q_2$, then these two transitions are one and the same and there is no irreversibility as there is only one transition. If $q_1 \neq q_2$, both C_1 and C_2 are valid predecessor configurations to C (introducing irreversibility) only if $\gamma_1 = \gamma_2$, $\sigma_1 = \sigma_2$ and $\rho_1 = \rho_2$. This case is actually exactly the same case as in the irreversibility of dfas. Since in dfas, there is only one tape and only allowed movement is to the right, this is the only possible case of irreversibility that can happen.

If none of the head movements match, both C_1 and C_2 are acting upon different input, proof and random tape positions. Therefore, for every assortment of the values for $\gamma_1, \gamma_2, \sigma_1$, σ_2, ρ_1 and ρ_2, we can always imagine an input, proof and random tape triple (x, p, r) such that both C_1 and C_2 are valid predecessor configurations to C. This case is always irreversible.

If some of the head movements match and some don't, only the tapes whose head movements match are of concern. Without loss of generality, assume only $d_x = d'_x$ and other head movements do not match. In this case, C_1 and C_2 are acting upon reading the same input tape position but different proof and random tape positions. If $\sigma_1 \neq \sigma_2$, there is no way that both C_1 and C_2 are valid predecessors to C as a single position in the input tape cannot contain two different symbols. However, if $\sigma_1 = \sigma_2$, we can always imagine an input, proof and random tape triple (x, c, r) such that both C_1 and C_2 are valid predecessor configurations to C.

Since the definition of pfa-v is somehow similar to a 3-head deterministic finite automaton, one can easily see that above analysis can be extended to any number of tapes when considering the reversibility. Results of this analysis provide a simple way to detect if there exists any irreversibility in transition diagram of a pfa-v (or dfa-v). When considering any two transitions into the same state of a pfa-v or dfa-v

- if both of the transitions have different head movements on every tape, these two transitions always introduce irreversibility irrespective of the symbols they are reading.
- If the two transitions have matching head movements on k tapes where $k \leq n$ and n is the number of tapes, this situation introduces irreversibility only if in all of those k tapes, both transitions are also reading matching symbols. Symbols read in remaining tapes are of no consequence to irreversibility.

2.3. Related Work

In [10], Villagra and Yamakami have proven a theorem (Theorem 1 in [10]) about verification systems that have direct implications to the results in this work. The terminology employed for the verification systems that have been studied in [10] is that of Merlin-Arthur proof systems (MA systems for short). Here, to juxtapose with our terminology, Merlin acts in a similar manner to the Prover and Arthur is to the Verifier. In the case of "deterministic Merlin", Merlin deterministically calculates proof string p using a function that depends only on the input string and supplies it to Arthur. If Arthur is running a real-time deterministic finite automaton (dfa for short), this MA system is exactly equal to our verification system with dfa-v(rt-input, rt-proof). Furthermore, if Arthur is running a real-time reversible finite automaton (rfa for short), this case corresponds to our verification system with dfa-v(rt-input, rt-proof, reversible).

Deterministic and real-time nature of such MA systems results in a well-known characterisation of them as non-deterministic computation. Every non-deterministic

choice made by a real-time non-deterministic finite automaton (nfa for short), can be thought of as a different proof symbol supplied to Arthur, by Merlin, in a MA system with a deterministic Merlin and an Arthur running a dfa. Due to this characterization, languages that can be verified using such an MA system is exactly equal to REG since nfa's are equal in power to dfa's. If the Arthur is running an rfa, the language class associated with this MA system is called 1NRFA in [10]. Prefix of "1" stands for one-way and it is equal to the real-time constraint in the terminology employed in our work and prefix "N" (that stands for non-determinism) are due to characterization of proof systems with non-deterministic computation.

It is important to note that definition of dfa employed in [10] in the context of an MA system is very similar to dfa-v(rt-input, rt-proof). In both systems, whole proof is supplied at the beginning, dfa at hand is consuming an input and proof symbol pair at each step, and has the ability to halt upon entering one of the halting states. In the case of an MA system with Arthur running an rfa (the system associated with the language class 1NRFA), due to both tapes being real-time, every tape head movement in every transition is to the right. This implies that reversibility condition is met when no two transitions with $\delta(q, \sigma, \gamma) = \delta(q', \sigma', \gamma')$ are allowed to exist if $(\sigma, \gamma) = (\sigma', \gamma')$ and $q \neq q'$. Therefore dfa-v(rt-input, rt-proof, reversible) and an MA system with Arthur running an rfa are equivalent. This implies that 1NRFA = DFA-V(rt-input, rt-proof, reversible). This allows us to state the following theorem.

Theorem 2.3.1. *REG = DFA-V(rt-input, rt-proof, reversible)*

Proof. This has been implicitly proven by Theorem 1 in [10] which states that 1NRFA is equal to REG. As it has been discussed above, MA system associated with 1NRFA and dfa-v(rt-input, rt-proof, reversible) are equivalent. A sketch of their proof can be summarized as follows:

In $REG \subseteq$ DFA-V(rt-input, rt-proof, reversible) direction, given any dfa M, for every set of transitions into a state that introduces irreversibility, add a set of distinct

proof symbols that will be read at those transitions. For all the other transitions, a single placeholder proof symbol can be used. In this way, it is possible to eliminate irreversibilities by simply extending the proof alphabet by finitely many symbols.

In DFA-V(rt-input, rt-proof, reversible) $\subseteq$ *REG* direction, given any dfa-v(rt-input, rt-proof, reversible) V, one can construct an nfa M that can recognize the same language. At each step, M non-deterministically guesses the next proof symbol and applies the transitions according to the transition diagram of V. There exists a non-deterministic path to acceptance only if there exists a proof that makes V accept. $\square$

3. RESULTS

3.1. Reversible Verification of Regular Languages

Although it is not surprising that providing the dfas with a whole new tape helps with reversibility (as in the case of Theorem 2.3.1), there is a stronger result here. In [4], a class of languages L_j (defined on j-symbol alphabet) have been shown to have the property that every dfa that recognize L_j has to have a state that has $j+1$ incoming transitions reading the same symbol into that state. If we were to run a reversible simulation of any such dfa using a dfa-v(rt-input, rt-proof, reversible) in the same way as in the proof of Theorem 2.3.1, we would need to use a proof alphabet of size at least $j+1$. However, it is possible to verify any L_j using a binary proof alphabet and using a dfa-v(rt-input, rt-proof, reversible).

For $j = 1$, L_j consists of every unary string except the empty string. For $j > 1$, let $F(i) = (i \ mod \ j) + 1$, $\Sigma_j = \{\sigma_1, \sigma_2, ..., \sigma_j\}$ and we can define

$$L_j = \{w \mid w \text{ ends with } \sigma_i \sigma_{F(i)} \quad 1 \leq i \leq j\}. \tag{3.1}$$

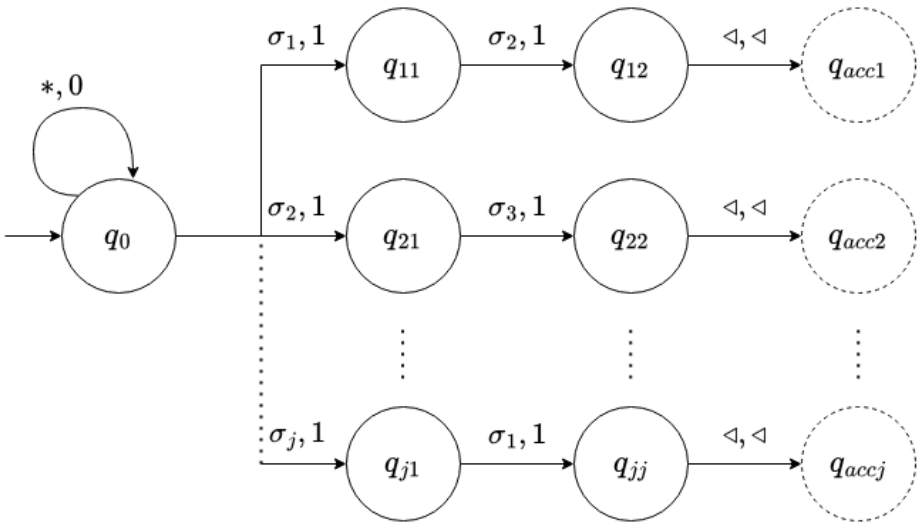

Figure 3.1. Generalized transition diagram for verification of L_j.

Theorem 3.1.1. *For every $j \geq 1$, there exists a dfa-v(rt-input, rt-proof, reversible) V_j that can verify L_j using a binary proof alphabet.*

Proof. Case of $j = 1$ is trivial as V_1 is able to accept upon reading any symbol in the input tape.

General transition diagram for the case of $j \geq 2$ is given in the Figure 3.1. For every member $x \in L_j$ of length $|x| \geq 2$, let the proof string p be $0^{|x|-2}1^2$. Initial state q_0 of V_j will self loop while 0 is written in the proof tape, for every input symbol. Upon reaching 1 in proof tape, V_j is at the point of last two elements of the input string. V_j branches into j different states (these branches do not share any states other than initial state) depending on the current input symbol σ_i and consumes the pair $(\sigma_i, 1)$ in two tapes. In every one of these j branches, V_j then expects to consume $(\sigma_{F(i)}, 1)$ in two tapes while transitioning into a new state for each branch. If these two steps were successful, V_j then expects to consume $(\triangleleft, \triangleleft)$ in both tapes and accepts. Every state of V_j has its own associated reject state where any unexpected input and proof symbol pair immediately gets rejected when the state in question comes across that pair.

Purpose of the proof string here is to signal the position of last two elements in the input. Prover has no way of deceiving V_j because V_j has the capacity to check whether the given proof is in the correct form or not. Since no state in this construction has two incoming transitions that read the same symbol pair, there are no irreversibilities. □

Although Theorem 3.1.1 is not a generalization for every irreversible regular language, it shows that hierarchies (in terms of the degree of the irreversibility, in some sense) such as the one in L_j might not translate to a linear increase in the size of the proof alphabet of dfa-v(rt-input, rt-proof, reversible) as it was is the case in the proof of Theorem 2.3.1. If we further allow the proof head of V_j to be a dfa-v(rt-input, 1way-proof, reversible), letting its proof head to be able to stay still, verification of L_j can be achieved only using a unary proof alphabet.

3.2. On the Properties of PFA-V

After seeing that the introduction of a proof tape lets the dfa to recognize every regular language reversibly, it is natural to ask the same question for non-regular languages. One potential target for this question is the class PFA-V(rt-input, 1way-proof, const-randomness). Reason being that this computational model, although having very tight computational limitations, have the capability to recognize non-regular languages. Extending the convention used in [5], we can define the abbreviations

$$\mathcal{V}(\text{rt-input}) := \text{PFA-V(rt-input, 1way-proof, const-randomness)}, \tag{3.2}$$

$$\mathcal{V}(\text{1way-input}) := \text{PFA-V(1way-input, 1way-proof, const-randomness)}, \tag{3.3}$$

$$\mathcal{V}_2(\text{rt-input}) := \mathcal{V}(\text{rt-input, binary-random-alphabet) } \textit{and} \tag{3.4}$$

$$\mathcal{V}_2(\text{1way-input}) := \mathcal{V}(\text{1way-input, binary-random-alphabet}). \tag{3.5}$$

Here, the class $\mathcal{V}_2$ is a restricted version of the class $\mathcal{V}$ where the associated pfa-vs are restricted to only have access to binary random alphabet (i.e. only able to branch with a probability of $\frac{1}{2}$). The class $\mathcal{V}_2$(rt-input), studied in [5, 6], have been shown to contain non-regular languages such as 0^n1^n and also the language of non-palindromes which is known to be context-free. In [5], an equivalence between $\mathcal{V}_2$(1way-input) and 1NFA($*$) and also an equivalence between $\mathcal{V}_2$(rt-input) and 1NSNFA($*$, $*$) (a subclass of 1NFA($*$)) have been proven.

It is helpful to describe language classes 1NFA($*$) and 1NSNFA($*$, $*$) for clarity. The class 1NFA($*$) contains all the languages that can be recognized by a one-way k-headed non-deterministic automata (1nfa(k) for short) for $k \in \mathbb{Z}^+$. 1nfa(k) is an extension to the typical definition of an nfa where the machine is given k independent input heads that are allowed to move right (or stand still) on the input tape at each step. The machine is allowed to non-deterministically branch out to multiple states at each step. If any one of the non-deterministic branches reaches an accept state, the machine is said to accept the input string.

The class 1NSNFA($*$, $*$) contains all the languages that can be recognized by a one-way k-head nonstop finite automaton with (up to) m nondeterministic choices per round (1nsnfa(k,m) for short) for $k, m \in \mathbb{Z}^+$. 1nsnfa(k, m) is a more constrained version of 1nfa(k) where the computation of the machine is composed of rounds. Every round, input heads take turns and move in a particular order. No two head moves at the same time. In its turn, an input head is moves at least one step to the right. Heads that reach the end of the input are let out of the iteration, while others keep the same order taking turns. In each round, the machine is allowed to make at most m non-deterministic choices. Other than this restriction on how its heads move and number of non-deterministic branchings allowed during these movements, 1nsnfa(k, m) behave like 1nfa(k). It is an unconventional and a particular restriction. However, under this restriction, a characterization with pfa-v(rt-input, 1way-proof, const-randomness) have been made possible.

In [5], the equality 1NFA($*$) = $\mathcal{V}_2$(1way-input) have been proven via simulations in both directions. In forward direction, given a 1nfa(k) M, a pfa-v(1way-input, 1way-proof, const-randomness) V with a binary proof alphabet is constructed. As a proof, V expects a complete step-by-step computation history of M as the proof string where each proof symbol contains information about what each input head of M is reading at that step and list of states that M is non-deterministically is in. At the beginning of the computation, V probabilistically branches into 2^m computations where $k \leq 2^m$ by reading m symbols from the random tape. Each branch is tasked with following the computation of one of the k heads in the computation history given by the proof string. If there exists an accepting computation history, every branch should lead to acceptance. If the Prover tries to mislead V into accepting a non-member, at least one branch will be able to reject by catching the mistake in the computation history. In the backwards direction, given a pfa-v(1way-input, 1way-proof, const-randomness) V with a binary proof alphabet that consumes at most m symbols from the random tape, a 1nfa(2^m) M is constructed. Since the computation of V is associated with 2^m deterministic computations, depending on the first m symbols of the random tape, each head of M is able to simulate one such deterministic computation. At each step

that V consumes a proof symbol, M non-deterministically tries every possible proof symbol. If there exists a valid proof for the input string, at least one non-deterministic branch should reach acceptance.

The equality $\mathcal{V}_2$(rt-input) = 1NSNFA($*$, $*$) is also proven in a similar manner. In forward direction, given a 1nsnfa(k, m) M, a pfa-v(rt-input, 1way-proof, const-randomness) V with a binary proof alphabet is constructed. V again probabilistically branches and simulates the input heads of M. However, V not being able to stand still on its input head restricts its ability to check the computation history provided by the Prover. All the restrictions put on the 1nfa(k) that makes it a 1nsnfa(k, m) makes the computation history of M be "checkable for misleading" by V. The complications that are associated with this rt-input constraint are discussed in detail in [5]. In the backwards direction, the simulation is much more straightforward. Given a pfa-v(rt-input, 1way-proof, const-randomness) V with a binary proof alphabet that consumes at most m symbols from the random tape, a 1nsnfa(2^m, 1) M is constructed. Since the computation of V is associated with 2^m deterministic computations, depending on the first m symbols of the random tape, each head of M is able to simulate one such deterministic computation. Since input head of V never stops, heads of M takes turns and moves one step to the right at each turn. For each round, they only need to make one non-deterministic choice because only a single proof symbol can be consumed by V in one step. At each step that V consumes a proof symbol, M non-deterministically tries every possible proof symbol and starts simulating each deterministic branch of V. If there exists a valid proof for the input string, at least one non-deterministic branch should reach acceptance by finding that proof string.

Theorem 3.2.1. $\mathcal{V}_2(\textit{rt-input}) = \mathcal{V}(\textit{rt-input})$

Proof. Since the other direction is trivial, only a proof for the inclusion $\mathcal{V}$(rt-input) $\subseteq$ $\mathcal{V}_2$(rt-input) direction is included.

Let V be a pfa-v(rt-input, 1way-proof, const-randomness) that reads at most k symbols from the random alphabet and whose random alphabet size is r. Then using the construction in the proof of 1NFA($*$) = $\mathcal{V}_2$(1way-input) [5], one can build a 1nfa(r^k) M that recognizes the language verified by V. However, due to V having a real-time constraint on its input head, M is a 1nsnfa(r^k, 1). Since $\mathcal{V}_2$(rt-input) = 1NSNFA($*$, $*$), there exists a pfa-v(rt-input, 1way-proof, const-randomness) V' that verifies the language recognized by M and has a binary random alphabet. □

Theorem 3.2.2. $\mathcal{V}_2$(*1way-input*) = $\mathcal{V}$(*1way-input*)

Proof. This proof is similar to that of Theorem 3.2.1.

Let V be a pfa-v(1way-input, 1way-proof, const-randomness) that reads at most k symbols from the random alphabet and whose random alphabet size is r. Then using the construction in the proof of 1NFA($*$) = $\mathcal{V}_2$(1way-input) [5], one can build a 1nfa(r^k) M that recognizes the language verified by V. Since $\mathcal{V}_2$(1way-input) = 1NFA($*$), there exists a pfa-v(1way-input, 1way-proof, const-randomness) V' that verifies the language recognized by M and has a binary random alphabet. □

These two theorems and the characterizations with respect to 1nfa classes provide a way to guarantee a canonical form for any pfa-v corresponding to a language in $\mathcal{V}$(rt-input) or $\mathcal{V}$(1way-input) due to the simulations provided for each directions of these equalities. Given any one of the verifiers from one of these classes, running the simulation forward and backwards should create verifiers with particular properties in their transition diagrams. This “canonical” quality might prove to be useful for some research. In the next section, we will also provide a construction that proves (Lemma 3.3.1) for any language in any one of these classes, there exists a corresponding pfa-v that consumes a single random tape symbol at the first step of the computation and never again, at the cost of a larger random alphabet size.

3.3. On the Reversibility Properties of PFA-V

This section primarily focuses on the reversibility properties of $\mathcal{V}$(rt-input). We will see that some types of irreversibilities that exist in the computations of pfa-v(rt-input, 1way-proof, const-randomness) can be eliminated by extending the proof alphabet. This method of elimination can be considered as an extension to the method used in the proof of Theorem 2.3.1. Following lemma will be required in later discussions of this method.

Lemma 3.3.1. *Let V be a pfa-v(rt-input, 1way-proof, const-randomness) that verifies the language L, moves in the random tape at most t times on any input and has a random tape alphabet of size m. Then, there exists a pfa-v(rt-input, 1way-proof, const-randomness) V' such that*

- *V' verifies L.*
- *V' has a random tape alphabet of size m^t.*
- *On every input, V' moves in the random tape alphabet once and only in the first transition.*

Proof. Let r_t be the string comprised of first t symbols of the random tape of V. Under any fixed r_t, the computation of V is completely deterministic and equivalent to one of dfa-v(rt-input, 1way-proof). Since there are m^t different possible r_t values, computation of V is associated with m^t possibly different deterministic computations. Random tape alphabet of V' has a different symbol for each possible r_t value. Construct V' as follows:

(i) Initial state of V' has m^t outgoing transitions associated with each random tape symbol r_t, all of which are moving right on the random tape.

(ii) Each one of these m^t paths lead to a clone of the sub-graph of the transition diagram of V that follows the deterministic computation associated with the random string r_t.

□

In the proof given above, constructed machine V' has the property that, regions of the transition diagram of V' associated with each random symbol r_t, do not share any state except the initial state.

It will be helpful to define a notation on classifying the transitions of a pfa-v with regard to their head movements. If we are given the transition

$$\delta(q, \sigma, \gamma, \rho) = (q', d_x, d_p, d_r), \tag{3.6}$$

all three values for d_x, d_p and d_r are in the set $\{0, 1\}$, since we are in the confines of $\mathcal{V}$(rt-input). We say that a transition is of type (1, 0, 0) when it is the case that $(d_x, d_p, d_r) = (1, 0, 0)$ and will use similar notation in all other triplets.

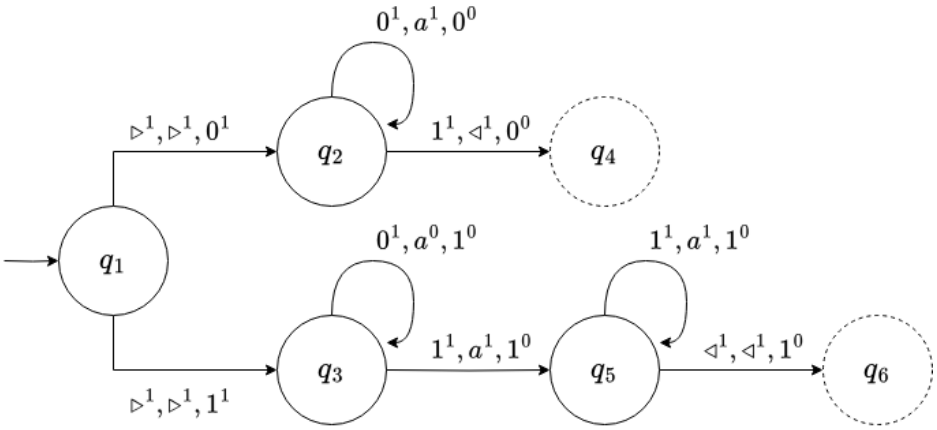

Figure 3.2. Irreversible transition diagram for L_{01}.

Before getting into the details of this method of eliminating irreversibilities, it will be helpful to consider a simpler case. Let the language $L_{01} = \{0^n 1^n \quad \text{where} \quad n \geq 1\}$ be defined on binary alphabet. The assumption $n \geq 1$ is made in order to keep the diagram simple. In Figure 3.2, a transition diagram for a pfa-v(rt-input, 1way-proof, const-randomness) that verifies this language is shown. The 0 and 1 symbols used in superscript denote the head movements on their corresponding tapes where 1 means a head movement and 0 means no head movement. The order of symbols are input,

proof and random tape. The states with dashed lines are accept states. For every state, there exist a hidden reject state. For every state, all the transitions that are not shown goes into the reject state corresponding to that state. Here, expected proof for a string in the form 0^n1^n is a^n. Machine only uses a single random bit. If it is 0, machine compares the number of 0's in the input tape to number of a's in proof tape, otherwise if it is 1, it compares the number of 1's in the input tape to number of a's in the proof tape. Since the prover does not have access to the contents of the random tape, it has to supply correct number of a's, otherwise one of the two deterministic branches will detect the inequality and reject.

Here, we can see that the only existing irreversibility in this diagram is at the incoming transitions to the state q_5 where we have two transitions of type (1, 1, 0) that read the same symbols on every tape. However, via a simple application of the process of eliminating the irreversibilities that will be explained in the proof of Theorem 3.3.2, it is possible to verify the same language reversibly. This modification is shown in Figure 3.3. In the figure, some of the unnecessary transitions added by the process is omitted to save space and increase readability. The expected proof for this new modified machine is $a_1 a_2^{(n-1)}$.

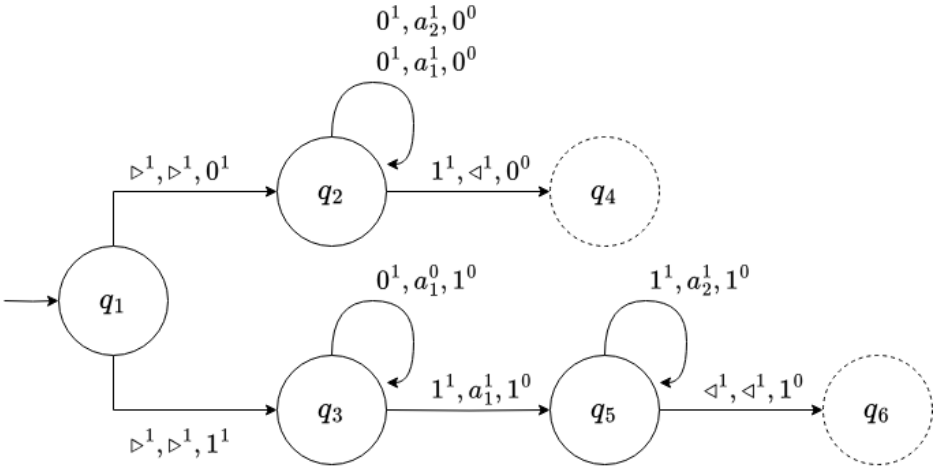

Figure 3.3. Reversible transition diagram for L_{01}.

It is possible to generalize this method that has been used in the example of L_{01} to get rid of every possible irreversibility arising from (1, 1, 0) transitions. This method is stated as the following theorem.

Theorem 3.3.2. *Let V_{init} be a pfa-v(rt-input, 1way-proof, const-randomness) that verifies the language L, moves in the random tape at most t times on any input and has a random tape alphabet of size n. Then, there exists a pfa-v(rt-input, 1way-proof, const-randomness) V' such that*

- *V' verifies L.*
- *V' has a random tape alphabet of size n^t.*
- *V' has a proof tape alphabet whose size is greater than or equal to that of V's.*
- *V' does not have any irreversibility associated with (1, 1, 0) type transitions.*

Proof. Construct V' as follows: Modify V_{init} as described in the proof of Lemma 3.3.1. Label the resulting machine V. Verifier V' will be constructed from V.

It will be helpful to introduce the concept of "irreversibility region" for the remainder of this proof. An irreversibility region R is a sub-graph of the transition diagram of a verifier V where

(i) region R consists of a single target state q, a set of enumerated source states q_i where $i \in \{1, 2, ..., k\}$ and a list of transitions connecting the source states to the target state.

(ii) Region R consists of every and only the transitions in V that read the same set of three symbols in all tapes, is of type (1, 1, 0) and from a source state to the target state. Meaning that these transitions are every transition of the form $\delta(q_i, \sigma, \gamma, \rho) = (q, 1, 1, 0)$ where $i \in \{1, 2, ..., k\}$.

(iii) The number $k \geq 1$ (size of the set of source states) is the largest possible number with respect to the target state q and the symbols (σ, γ, ρ).

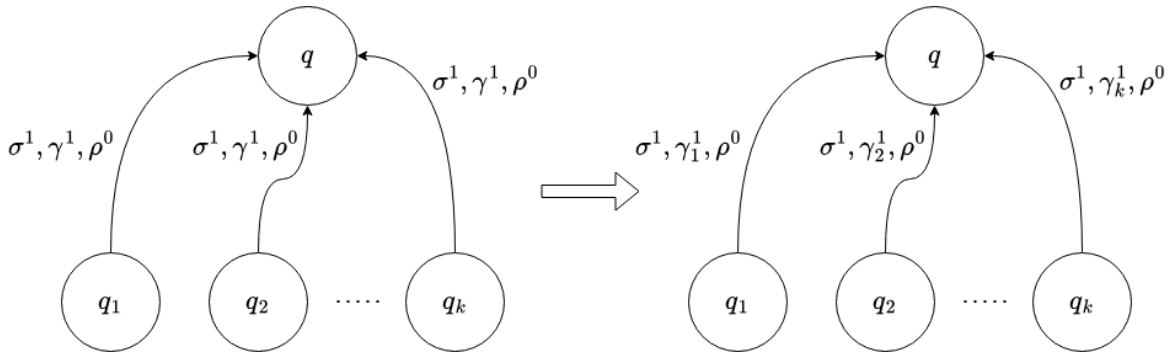

Figure 3.4. Irreversibility elimination in a pfa-v.

By these properties, an irreversibility region R is uniquely defined by its target state q and the symbols (σ, γ, ρ) that is being read by its transitions. An example of an irreversibility region R is shown in the left side of the Figure 3.4. Definition of an irreversibility region does not have to be reduced to only (1, 1, 0) type transitions. However, for the purposes of this proof, this definition will suffice.

Figure 3.4 shows the application of the method (which will be presented in this proof) only for a single irreversibility region. The figure only aims to show the gist of the idea behind the method. Construction of V' from V will be more complicated than that.

Since irreversibility regions are defined uniquely by target states and a triplet of symbols, there are finitely many of them on any given verifier. Moreover, they can be categorized by only the proof symbols that are being read. For a proof symbol γ, let m be the total number of unique irreversibility regions in the transition diagram of V, such that for $1 \leq i \leq m$, region R_i has the property that

- there exists a state q_i (the target state) that has k_i distinct incoming transitions of the type (1, 1, 0).
- Each one of those k_i transitions reads the same triplet of symbols $(\sigma_i, \gamma, \rho_i)$.

Notice that these m regions are all of the irreversibility regions associated with the proof symbol γ. For every region R_i, fix an ordering $\{q_{i,1}, q_{i,2}, ..., q_{i,k_i}\}$ on the source

states of the transitions. We can modify pfa-v(rt-input, 1way-proof, const-randomness) V' by

(i) removing γ from the proof alphabet of V. Add $\prod_{i=1}^{m} k_i$ new symbols. Let these symbols be indexed using the notation $\gamma[a_1, a_2, ..., a_m]$ where $a_i \in \{1, 2, ..., k_i\}$ for $i \in \{1, 2, ..., m\}$.

(ii) Replace all the transitions (not only the ones in irreversibility regions) that read the symbol γ in the proof tape with $\prod_{i=1}^{m} k_i$ new duplicate transitions that read every possible $\gamma[...]$ in the proof tape, everything else remaining the same.

(iii) For $1 \leq i \leq m$ and $1 \leq j \leq k_i$, remove $\delta(q_{i,j}, \sigma_i, \gamma[a_1, a_2, ..., a_m], \rho_i) = (q_i, 1, 1, 0)$ if $a_i \neq j$.

(iv) For every removed transition in previous step, add a new "rejecting" transition reading the same symbols and from the same source state as the removed transition, to a reject state that is unique to this source state. If this source state does not already have a unique reject state, add a new rejecting state for this purpose.

For every region R_i, the third step in the construction above ensures that no two states q_{i,j_1} and q_{i,j_2} will be able to transition into q_i while reading the same proof symbol. This is because the proof symbol $\gamma[a_1, a_2, ..., a_m]$ read by these two states will always differ at the index a_i. Recall that the region R_i is contains the largest set of transitions that introduce a irreversibility corresponding to a target state q_i and a fixed triplet of symbols. After the modification, this target state q_i does not have any transition of the type (1, 1, 0) whose source states read the "same triplet of symbols", irreversibility is eliminated. A simple example of this is shown in Figure 3.4.

Therefore, this modification, at the cost of adding new symbols to the proof alphabet, eliminates the irreversibilities in all of the m regions separately but all at once. Therefore, it destroys all of the m irreversibility regions.

This process will be applied to the irreversibility regions associated with every other proof symbol separately. This can be done independently because these regions

differ at the proof symbol. Therefore, none of them share any transitions and all the modification is being made on the transitions. When it is applied for every proof symbol, there are no regions left with (1, 1, 0) type irreversibilities in the transition diagram of V'.

The fact that all the irreversibilities arising from (1, 1, 0) type transitions are eliminated from V does not imply that V' is able to verify the same language L as V. In order to show that, we have to prove two statements.

- If $x \in L$ then there exists a proof p' such that when supplied with the proof p', V' accepts with probability 1.
- There exists $0 < \epsilon < 1$ such that for all $x \notin L$ and proof p', V' rejects with probability at least $1 - \epsilon$.

Start by proving the second statement. These statements are true in the case of V. Therefore, the question reduces to the following: After our modifications to V, are we allowing some new proofs that might lead V' to accept the non-members of L with probability 1? To this end, it is important to focus on how we are modifying the behavior of V while eliminating the irreversibility region. For the sake of simplicity, assume that we applied the elimination procedure only to the m irreversibility regions associated with the proof symbol γ. At this point, behavior of V' only differs from that of V in the computational steps where V previously read the symbol γ because we only modified the transitions that contain γ in the proof tape. Step (ii) in our procedure guarantees that the behavior of V' in every one of the newly added proof symbols $\gamma[...]$ mirrors exactly that of V except for the transitions in the m irreversibility regions. Therefore, outside of these m regions, symbol $\gamma[...]$ is a redundant copy of symbol γ. Inside the m regions, step (iii) in our procedure removes some of the transitions associated with the newly added proof symbols. This deletion results in a behavior difference between two machines for some of the proof symbols. However, inside these m regions, every case where behavior of V' differs from that of V results in a rejection by V' because of the deletion. To summarize, behavior of V' upon reading any one of

the newly added proof symbols $\gamma[...]$ attempts to mirror the behavior of V upon reading γ except for a few cases where in those cases where V' fails to mirror, V' rejects. This means that it is impossible for this procedure to increase acceptance rate of V. Notice that this procedure is applied for the irreversibility regions associated with every proof symbol independently and this argumentation holds for resulting final machine after the modification is applied for every symbol. Therefore, if V rejects a non-member $x \notin L$ with probability at least $1 - \epsilon_V$ on any given proof string, V' rejects the same input string with probability at least $1 - \epsilon_{V'}$ on any given proof string where $1 - \epsilon_V \leq 1 - \epsilon_{V'}$.

In order to prove the first statement, we will construct a proof string p' for V' by modifying a valid proof string p of V, for a member input string $x \in L$. This argument will follow a similar schema to the previous one in that construction will be shown only for the case where the elimination procedure is only applied to the m irreversible regions associated with a single proof symbol γ. However, since the procedure is independent for every proof symbol, it is possible to extrapolate for the case where procedure is applied to regions associated with every possible proof symbol.

Since V' mirrors the behavior of V on its newly added $\gamma[...]$ proof symbols, replace every occurrence of γ in p with an arbitrary selection of one of the $\gamma[...]$ symbols. As it stands, it is possible that V' would accept input proof pair (x, p') with probability 1. However, as we have established previously, when supplied with one of the newly added proof symbols, there are certain cases where V' rejects (inside the irreversibility regions), instead of the mirroring the behavior of V. If this distinction did not exist, V' would accept (x, p') whenever V would accept (x, p). If it is the case that V' might possibly reject the pair (x, p'), rejection has to happen at a particular proof tape position that contains one of the $\gamma[...]$ symbols. We can say this for sure because for every other symbol unrelated to γ, V' mirrors V. And we know that V' tries to mirror V for every $\gamma[...]$ except for the computations in one of the m irreversibility regions, in which case it is possible that V' might reject, instead of mirroring. Therefore, if V' rejects (x, p') with probability greater than 0, this rejection has to happen in a region that previously was one of the m irreversible regions and at the point of reading a

particular proof tape location that contains one of the newly added $\gamma[...]$ symbols. Let us refer to these locations in the proof string "the problematic locations". Problematic locations of p' (for input x) have the following properties.

- There may be multiple problematic locations in p'.
- Each problematic location in p' contains one of the newly added $\gamma[...]$ symbols.
- Each problematic location is associated with a particular transition in "at least" one of the m irreversible regions in V associated with the proof symbol γ.
- For each problematic location, there exists a $\gamma[...]$ symbol where upon replacing the symbol in the problematic location with this symbol, this location ceases to be problematic.

If we are able to prove the fourth property, then we can make sure that V' mirrors the behavior of V and the proof is completed.

The specification "at least" made in the third property is due to the existence of random tape. Since we modified V_{init} using Lemma 3.3.1 to construct V, V has the property that it only moves once in the random tape at the beginning of its computation, it has n^t random tape symbols and associated with each one of these n^t random tape symbols, computation of V has n^t deterministic branches that only share the initial state of V and no other state. Therefore, each one of the m irreversibility regions associated with the proof symbol γ belongs to one of these n^t branches, consequentially is associated with one of the random tape symbols that belong to the deterministic branch. In order for a location in p' to be problematic, V' needs to be in an region that was previously an irreversibility region (in V) while reading that proof tape location. However, it is possible that V' reaches two different irreversibility regions with two different random tape symbols, at the same proof tape location. Therefore, problematic locations can be associated with more than one irreversibility regions.

Assume that a problematic location in p' is associated with only the i^{th} irreversibility region R_i in V and at that location the symbol $\gamma[a_1, a_2, ..., a_m]$ is written.

This means that there exists a random tape symbol ρ such that, when supplied with (x, p', ρ), V' rejects while reading the proof symbol at the problematic location. Note that this rejection has to happen from one of the source states of irreversibility region R_i whose target state is q_i. Under a fixed random tape symbol ρ, computation of V' is deterministic, therefore, this rejection must be happening at one particular source state $q_{i,j}$ where $j \in \{1, 2, ..., k_i\}$. However, we know that, due to the step (iii) of our elimination procedure, the transition $\delta(q_{i,j}, \sigma_i, \gamma[a_1, a_2, ..., a_m], \rho) = (q_i, 1, 1, 0)$ is not removed if $a_i = j$. This means that if we replace the $\gamma[...]$ symbol at this problematic location of p' with a $\gamma[...]$ symbol whose i^{th} index is j, we know that V' does not reject and proceeds to mirror the behavior of V at this location.

We have shown that in the case where a problematic location in p' is associated with only a single irreversibility region, we have a solution. In the more general case, if a problematic location in p' is associated with more than one irreversible region in V, same logic presented in the previous paragraph can be extended. However, before presenting that extension, it is important to note that in this general case every irreversibility region associated with this problematic location should be associated with a different random tape symbol. To prove this by contradiction, assume otherwise, that a problematic location in p' is associated with two different irreversibility regions R_{i_1} and R_{i_2} and further assume that these two regions are associated with the same random tape symbol ρ. Then, V' rejects (x, p', ρ) while reading a $\gamma[...]$ symbol at the problematic location in p'. However, since ρ is fixed, computation of V' is deterministic but then, it is impossible for V' to reach two different irreversibility regions because under a deterministic computation V' is only allowed to execute a single transition in an irreversibility region at a particular proof tape location. This is because the transitions in irreversibility regions are of type (1, 1, 0). Therefore, if V' is executing one at a particular proof tape location, it is impossible to execute another one at the same location (under a fixed random tape symbol). Then, this problematic location cannot be associated with two different irreversibility regions.

Now let us present the most general case for solving the problematic locations in p'. Let a problematic location in p' be associated with l different irreversibility regions (out of m regions) associated with the proof symbol γ. As we have concluded in the previous paragraph, each one of these l regions should be associated with a different random tape symbol ρ_z. Let these regions be enumerated by R_{i_z} where $z \in \{1, 2, ..., l\}$ and $i_z \in \{1, 2, ..., m\}$. Then, V' rejects (x, p', ρ_z) upon reading a $\gamma[...]$ symbol at the problematic location in p'. This rejection happens from one of the source states of region R_{i_z} and since under a fixed ρ_z, the computation of V' is completely deterministic, it is always the same source state. Without loss of generality, assume this source state is q_{i_z,j_z} where $j_z \in \{1, 2, ..., k_{i_z}\}$. Step (iii) of the elimination procedure ensures that the transition $\delta(q_{i_z,j_z}, \sigma_{i_z}, \gamma[a_1, a_2, ..., a_m], \rho_z) = (q_{i_z}, 1, 1, 0)$ exists if $a_{i_z} = j_z$. Then, if we select a $\gamma[...]$ symbol whose ${i_z}^{th}$ index is j_z for $z \in \{1, 2, ..., l\}$ and put it in place of the $\gamma[...]$ symbol written in the problematic location of p', we can make sure that each one of these l irreversible regions associated with this problematic location successfully transitions into the target state and mirrors the behavior of V upon reading γ.

To summarize the process above, given the verifier V_{init}, we start by modifying it using Lemma 3.3.1 and we get V. We then determine all the irreversibility regions associated with a proof symbol and apply the elimination procedure. After the elimination, it is possible that the proof constructed by arbitrary selection of newly added proof symbols have problematic locations. However, as we have established, every problematic location has a particular proof symbol that solves the problem. Since the prover is all knowing, existence of such solution for every member is enough. We have established that this modification cannot make V' accept an input string where V' previously be able to reject. Since we can apply this procedure and get a modified verifier that can verify the same language as V every time, we can apply this procedure for the irreversibility regions associated with different proof symbols one by one. When we exhaust every proof symbol, final modified machine V' cannot have any irreversibility associated with (1, 1, 0) type transitions and still able to verify the same language as L. □

We now provide some examples of languages in the class $\mathcal{V}$(rt-input, reversible).

Theorem 3.3.3. *Following languages are in $\mathcal{V}$(rt-input, reversible). Assume every one of them is defined on binary alphabet.*

(i) $L_{01} = \{0^n 1^n \quad where \quad n \geq 1\}$

(ii) $L_{twin} = \{\omega \# \omega \quad where \quad \omega \in \Sigma^* \quad and \quad \# \notin \omega\}$

(iii) L_{nonpal}*, the language consisting of the strings that are not palindromes.*

Proof. For all the figures, superscripts in transition labels correspond to head movements associated with symbols read. The order of symbols on the label are input, proof and random tape. Every state has a unique hidden reject state. Every valid transition that are not shown in the figure goes into reject state associated with the corresponding state. The special symbol Σ is used to represent multiple labels at once. It should be interpreted as "any symbol from Σ is allowed here and there is a transition as such".

(i) Proof is shown in Figure 3.3 and explained above.

(ii) Proof is shown in Figure A.1. Expected proof string for the member inputs is ω. Depending on the first symbol in the random tape alphabet, the verifier branches into two. First branch is tasked with matching the part of the input string until # symbol with the proof string. Second branch is tasked with matching the part of the input string after # symbol with the proof string. If both matches succeed, verifier accepts with the probability 1. This is exactly the case of valid input. If the input is not a member, no proof can mislead both branches at the same time. Therefore, the verifier rejects with the probability of at least $\frac{1}{2}$.

(iii) Proof is shown in Figure A.2. This is a reversible implementation of the verifier described in [5]. Members of the language of non-palindromes are in the form $x\sigma_1 y\sigma_2 z$ where $x, y, z \in \Sigma^*$, $\sigma_1, \sigma_2 \in \Sigma$, $\sigma_1 \neq \sigma_2$ and $|x| = |z|$. Expected proof is in the form $0^{|x|}10^{|y|}$. The verifier branches into two by consuming a random tape symbol. First branch takes two steps in the input for every 0 in the proof string until it reaches 1 in the proof string. It takes two more steps while consuming

1 in the proof tape and continues to consume remaining 0's in the proof tape at the same speed at the input. If the proof is in the correct form, this branch should reach the both end markers at the same time because for member inputs the equality $|x\sigma_1 y\sigma_2 z| = 2|x| + |y| + 2$ should hold. Second branch assumes the proof is in the correct form and starts reading both the input and the proof at the same speed. At the point it reads 1 in the proof tape, it records the value it has read in the input tape (σ_1 for member inputs) in its finite memory. At the point it reads the end marker in the proof tape, the symbol it is reading in the input tape (σ_2 for member inputs) should be different from the one it kept in its finite memory. Accepts if the two are different. A valid proof for a member string leads to the acception by both branches, thus probability 1. If the input is not a member at least one branch rejects, thus the verifier rejects with a probability at least $\frac{1}{2}$.

□

4. CONCLUSION

Introducing probabilistic behaviour in terms of a random tape helps analyzing irreversibility by reducing randomness to a tape head movement. Definitions and analysis made in this work provide a simple way (for human eyes) to just look at a transition diagram of a multi-head automaton and recognize the irreversible transitions by a couple of simple checks. Not to forget that this is only a particular definition of reversibility, out of many.

It is no surprise that introduction of a proof tape, a whole new memory apparatus in some sense, seems to increase the ability to run a computation reversibly as we have seen in the case of regular languages. However, to take this one step further, the statement "$\mathcal{V}$(rt-input) = $\mathcal{V}$(rt-input, reversible)" is supplied as a conjecture. Although a mathematically rigorous proof have not been supplied in support of this statement, there are a couple of reasons for this conclusion. The main reason is that, in the process of studying these computational models, a lot of work has been put into finding a language that "forces" irreversibility, with no success. Even though the fact that a counter example has not been found out of infinite possibilities does not imply that it does not exist, insights and intuitions developed in this process leads one to believe that this conjecture is true.

A proof tape whose alphabet we can extend as much as we want provides a strong tool for eliminating irreversibilities. As we have shown in the previous section, irreversibilities arising from moving on both input and proof tapes at the same time can be eliminated. pfa-v(rt-input, 1way-proof, const-randomness) can only move real-time in input tape. As it has been shown in Section 3.1, these machines can already verify regular languages reversibly. In order to verify non-regular languages, they need both a proof tape that they can choose not to move on (and use this power at least once, otherwise they would be rt-proof and would not be able to recognize non-regular languages) and also a random tape whose contents are hidden from the prover. When

the machine is moving on the proof tape, we know we can eliminate irreversibilities. Only when the machine stands still on the proof tape that we don't know how to deal with irreversibilities (in a general sense). Since we can always modify the machine such that it does not move on the random tape except for the first step, sub-computations that do not move in the proof tape can be executed using a dfa (all the transitions on these subcomputations are of type (1, 0, 0)). This means that those sub-computations correspond to substrings in the input string that are regular expressions. This, coupled with the fact that these machines can verify regular languages reversibly, leads one to believe that the proof tape have always enough space to eliminate irreversibilities for pfa-v(rt-input, 1way-proof, const-randomness) although no such general method have been found.

In [3] a method for reversibly simulating the computation of a dfa using a 2rfa have been provided. It might be possible to apply a similar technique between pfa-v(rt-input, 1way-proof, const-randomness) and pfa-v(2way-input, 1way-proof, const-randomness). More specifically, irreversibilities arising from (1, 0, 0) type transitions are exactly like the transitions of a dfa, therefore same method as in [3] might potentially eliminate those, at the cost of having a 2way input head. However, in the case of verification systems, having a 2way input head might already be a very powerful improvement, to the point where the reversible simulation is almost trivial.

REFERENCES

1. Landauer, R., "Irreversibility and Heat Generation in the Computing Process", *IBM Journal of Research and Development*, Vol. 5, No. 3, pp. 183–191, 1961.

2. Bennett, C. H., "Logical Reversibility of Computation", *IBM Journal of Research and Development*, Vol. 17, No. 6, pp. 525–532, 1973.

3. Kondacs, A. and J. Watrous, "On the Power of Quantum Finite State Automata", *Annual Symposium on Foundations of Computer Science*, Florida, USA, pp. 66–75, 1997.

4. Yılmaz, Ö., F. Kıyak, M. Üngör and A. C. C. Say, "Energy Complexity of Regular Language Recognition", *Implementation and Application of Automata*, Rouen, France, pp. 200–211, 2022.

5. Gezer, M. U., Ö. Dolu, N. Ersoy and A. C. Say, "Real-Time, Constant-Space, Constant-Randomness Verifiers", *Theoretical Computer Science*, Vol. 976, No. 5, pp. 114–155, 2023.

6. Say, A. C. C. and A. Yakaryılmaz, "Finite State Verifiers with Constant Randomness", *How the World Computes*, Heidelberg, Germany, pp. 646–654, 2012.

7. Shamir, A., "IP = PSPACE", *Journal of the ACM (JACM)*, Vol. 39, No. 4, p. 869–877, 1992.

8. Ji, Z., A. Natarajan, T. Vidick, J. Wright and H. Yuen, "MIP* = RE", *Communications of the ACM*, Vol. 64, No. 11, p. 131–138, 2021.

9. Yakaryilmaz, A., R. Freivalds, A. C. C. Say and R. Agadzanyan, "Quantum Computation with Write-Only Memory", *Natural Computing*, Vol. 11, No. 1, pp. 81–94, 2012.

10. Villagra, M. and T. Yamakami, “Quantum and Reversible Verification of Proofs Using Constant Memory Space”, *Theory and Practice of Natural Computing*, Granada, Spain, pp. 144–156, 2014.

APPENDIX A: TRANSITION DIAGRAMS

In this appendix, two figures have been supplied in a larger form in order to improve the readability of the complicated transitions. These figures have been referred in the proof of Theorem 3.3.3.

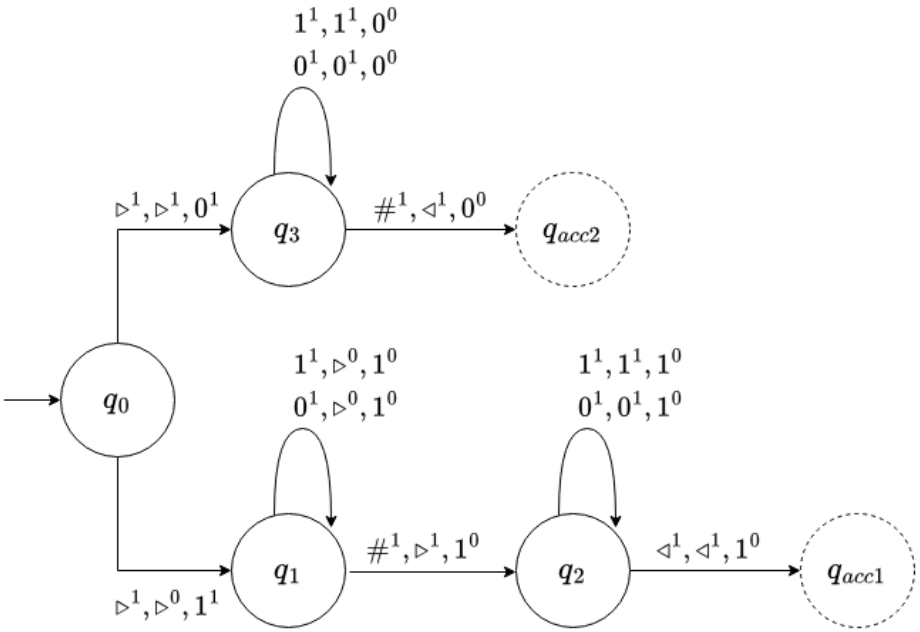

Figure A.1. Transition diagram for L_{twin}.

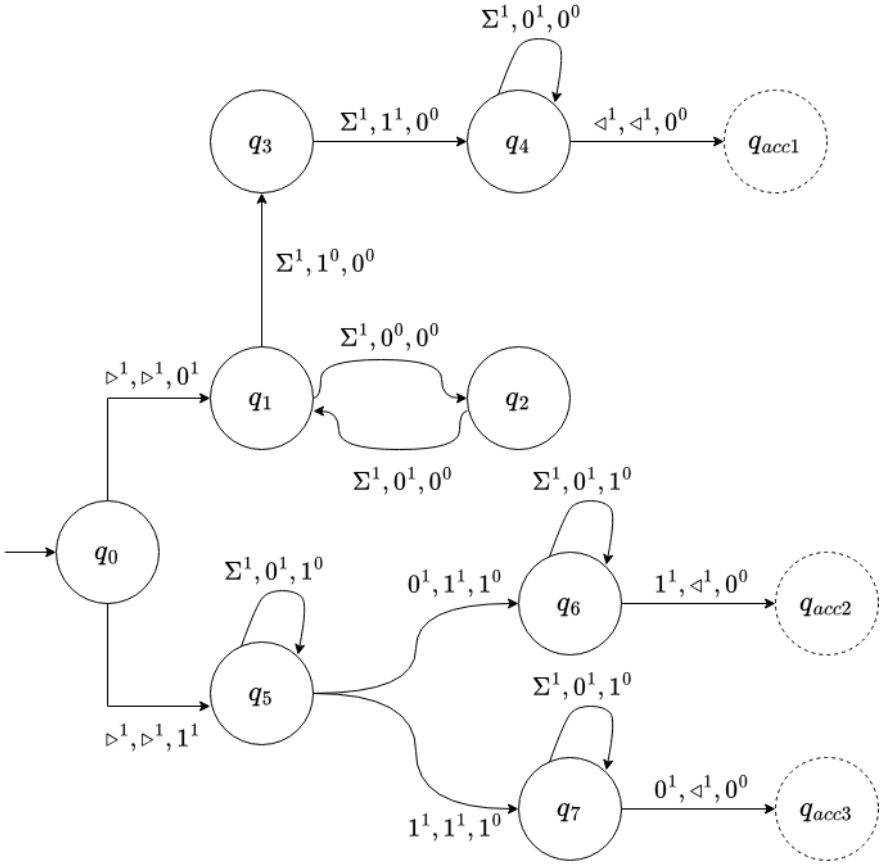

Figure A.2. Transition diagram for L_{nonpal}.