NP-Complete Ranges of Counting Functions of Nondeterministic Finite Automata*

Stefan Karl
Institute for Informatics
Julius Maximilian University Würzburg

December 12, 2009

Abstract

The complexity of counting with polynomial time Turing machines has been well studied[1]. But the concept of counting can not only be applied to Turing machines, but also to other nondeterministic devices as e.g. nondeterministic finite automata. One method to get a decision class from counting NFAs is to consider the ranges of their counting functions. A first survey on these ranges has been given by Rich and Slutzki [RS88] who proved a lower bound (regular languages) and an upper bound (nondeterministic linear space) for their complexities. In this report we show that there are NP-complete ranges of counting NFAs, thus significantly improving the previous lower bound from [RS88]. A similiar result for probabilistic finite automata is proven.

1 Introduction

For any nondeterministic device M, such as a Turing machine or an automaton, we can count the number of paths that halt in accepting states, getting the counting function $\#M$ which maps every possible input to a natural number.

There are various methods to define decision classes on the basis of the counting functions given by nondeterministic machine models. Using the counting functions of NP machines as oracles for a deterministic polynomial time machine, for example, is known to yield fairly powerful complexity classes ([Tod91]). Alternatively, we can obtain a decision problems from counting functions by considering their ranges (i.e. their codomains).

In this paper we will focus on the class range($\#$ NFA) of ranges of counting functions of nondetermistic finite automata which were investigated by Craig A. Rich and Giora Slutzki in 1988 ([RS88]). In this paper they proved constructively that for every regular language there is a NFA whose counting function has a range equal to this language, and that there are ranges which are context sensitive but not context free ($0^n1^n0^n$). By imposing a simple cap on the entries of the vector which holds the number of paths in

*Extended abstract of the author's Bachelor Thesis

[1] An up-to-date summary can be found in [AB09]

each state during the simulation of a counting NFA on a Turing machine, they showed that all ranges of NFAs can be decided in nondeterministic linear space.

In the following sections we will prove that there are NFAs whose counting functions have NP-complete ranges, by defining and validating such a range which is essentially equivalent to the subset sum problem. This shows that counting yields sets far more complex than $0^n1^n0^n$ on a model as simple as a finite automaton and implies a new lower bound for the time complexity of range(# NFA).

In a short corollary, we transfer this result to probabilistic finite automata by showing that there are PFAs for which the set of all acceptance probabilities is NP-complete.

2 Preliminaries

2.1 Definitions and Notations

In this report we will partially adopt the notation from [RS88]. Let $\mathbb{B}_1 = 1\{0,1\}^* \cup \{\varepsilon\}$ be the language of all binary strings without leading zeroes. The function $\text{bin} : \mathbb{N} \to \mathbb{B}_1$ maps a natural number n to its binary representation without leading zeroes. Note that $\text{bin}(0) = \varepsilon$ (empty word) and bin is a bijection. The inverse function is $\text{bin}^{-1} : \mathbb{B}_1 \to \mathbb{N}$. We extend these definitions to sets and functions by defining $\text{bin}(N) = \{\text{bin}(n) \mid n \in N\}$, $\text{bin}(f)(x) = \text{bin}(f(x))$, $\text{bin}^{-1}(B) = \{\text{bin}^{-1}(b) \mid b \in B\}$ and $\text{bin}^{-1}(f')(x) = \text{bin}^{-1}(f'(x))$ for $N \subseteq \mathbb{N}$, $B \subseteq \mathbb{B}_1$, $f : A \to \mathbb{N}$ and $f' : A \to \mathbb{B}_1$ (A is an arbitrary set). At one point we will also use the ordinary binary representation bin' for which $\text{bin}'(0) = 0$.

A *nondeterministic finite automaton* (*NFA*) is a 5-tuple $M = (S, \Sigma, \delta, I, F)$, where S is the finite set of *states*, Σ is the finite *input alphabet*, $\delta : S \times \Sigma \to \mathcal{P}(S)$ is the *transition function*, $I \subseteq S$ are the *initial states* and $F \subseteq S$ is the set of *final states*. To obtain a class of functions we define recursively a *counting function* $\#\delta : S \times \Sigma^* \to \mathbb{N}$ by

$$\#\delta(q, \varepsilon) = \begin{cases} 1 & \text{if } q \in F \\ 0 & \text{else} \end{cases} \tag{2.1}$$

$$\#\delta(q, \sigma x) = \sum_{p \in \delta(q,\sigma)} \#\delta(p, x) \tag{2.2}$$

and a counting function $\#M : \Sigma^* \to \mathbb{N}$ for the entire NFA by

$$\#M(x) = \sum_{q \in I} \#\delta(q, x) \tag{2.3}$$

Informally, $\#\delta(q, x)$ is the number of accepting paths of the NFA if the computation starts in state q and the automaton reads the input x, and $\#M(x)$ is the number of accepting paths of M starting from the initial states I and reading the input x. For languages $L \subseteq \Sigma^*$ and a NFA with the input alphabet Σ we define $\#\delta(q, L) = \{\#\delta(q, x) \mid x \in L\}$ and $\#M(L) = \{\#M(x) \mid x \in L\}$. For an NFA M, $\text{range}(\#M) = \#M(\Sigma^*)$ is the range of $\#M$. Finally, $\#\,\text{NFA} = \{\#M \mid M \text{ is an NFA}\}$ is the class of all counting functions of NFAs and $\text{range}(\#\,\text{NFA}) = \{\text{range}(\#M) \mid M \text{ is an NFA}\}$ is the class of all their ranges.

2.2 Linear Automata

To facilitate the formal verification of NFAs in the following sections, we use a well-known extension of NFAs which can greatly reduce the number of states in some cases.

A *linear automaton* (LA) is a 5-tuple $M = (S, \Sigma, \delta, I, F)$, where S is the finite set of *states*, Σ is the finite *input alphabet*, $\delta : S \times \Sigma \times S \to \mathbb{N}$ is the *transition function*, $I \subseteq S$ are the *initial states* and $F \subseteq S$ is the set of *final states*. Informally, δ maps a triple (q, σ, p) of a current state q, an input symbol σ and a subsequent state p to the number of paths leading from q to p when reading σ. Again, we recursively define a *counting function* $\#\delta : S \times \Sigma^* \to \mathbb{N}$ by

$$\#\delta(q, \varepsilon) = \begin{cases} 1 & \text{if } q \in F \\ 0 & \text{else} \end{cases} \tag{2.4}$$

$$\#\delta(q, \sigma x) = \sum_{p \in S} (\delta(q, \sigma, p) \cdot \#\delta(p, x)) \tag{2.5}$$

and an according counting function $\#M : \Sigma^* \to \mathbb{N}$ for the entire LA by

$$\#M(x) = \sum_{q \in I} \#\delta(q, x) \tag{2.6}$$

Additionally, we define the *reverse counting function* $\xi : S \times \Sigma^* \to \mathbb{N}$ by

$$\xi(q, \varepsilon) = \begin{cases} 1 & \text{if } q \in I \\ 0 & \text{else} \end{cases} \tag{2.7}$$

$$\xi(q, x\sigma) = \sum_{p \in S} (\delta(p, \sigma, q) \cdot \xi(p, x)) \tag{2.8}$$

Informally, $\xi(q, x)$ is the number of paths in state q when M has read the input x starting from the initial states I. Since for all inputs $x = \sigma_n \sigma_{n-1} ... \sigma_1$ it can be easily shown that $\sum_{p_0 \in F} \xi(p_0, x) = \sum_{p_n \in I} \#\delta(p_n, x)$ we can alternatively obtain $\#M(x)$ for an automaton M by determining $\sum_{s \in F} \xi(s, x)$, which will turn out to be more intuitive than the original definition. The definitions of $\#\delta(q, L)$, $\#M(L)$, $\#\,\mathrm{LA}$, $\mathrm{range}(\#M)$, $\#\,\mathrm{LA}$, $\mathrm{range}(\#\,\mathrm{LA})$ for a LAs are analogous to those of NFAs.[2]

Lemma 1. *LAs and NFAs are equivalent, that is, for every NFA $M' = (S', \Sigma', \delta', I', F')$ there is a LA $M = (S, \Sigma, \delta, I, F)$ with $\#M = \#M'$ and vice versa.*

Proof. Trivially, we get a equivalent LA M for a given NFA M' by defining $S = S'$, $\Sigma = \Sigma'$, $\delta(q, \sigma, p) = \begin{cases} 1 & \text{if } p \in \delta'(q, \sigma) \\ 0 & \text{else} \end{cases}$, $I = I'$ and $F = F'$.

[2] We do not use the designation LA as it is usually used for various classes of decision problems for linear automata.

For a given LA M consider the LA $M^\diamond = (S^\diamond, \Sigma^\diamond, \delta^\diamond, I^\diamond, F^\diamond)$ defined by

$$S^\diamond = \bigcup_{s \in S} \bigcup_{1 \leq m \leq \max_{t \in S, \sigma \in \Sigma} \delta(t,\sigma,s)} \{s_m\} \qquad \text{(where the } s_i \text{ are new states)} \quad (2.9)$$

$$\Sigma^\diamond = \Sigma \quad (2.10)$$

$$\delta^\diamond(q_i, \sigma, p_j) = \begin{cases} 1 & \text{if } j \leq \delta(q, \sigma, p) \\ 0 & \text{else} \end{cases} \qquad \text{(for all } q_i, p_j \in \Sigma^\diamond) \quad (2.11)$$

$$q_i \in I^\diamond \leftrightarrow (q \in I \wedge i = 1) \qquad \text{(for all } q_i \in \Sigma^\diamond) \quad (2.12)$$

$$q_i \in F^\diamond \leftrightarrow q \in F \qquad \text{(for all } q_i \in \Sigma^\diamond) \quad (2.13)$$

By a short induction on the length of the input x one can prove that $\#\delta^\diamond(q_i, x) = \#\delta(q, x)$ holds for any $q_i \in S^\diamond$. Since $\forall(q_i, \sigma, p_j) \in S \times \Sigma \times S : (\delta^\diamond(q_i, \sigma, p_j) \leq 1)$ the NFA M' with $S' = S^\diamond$, $\Sigma' = \Sigma^\diamond$, $p_j \in \delta(q_i, \sigma) \leftrightarrow \delta^\diamond(q_i, \sigma, p_j) = 1$, $I' = I^\diamond$ and $F' = F^\diamond$ is equivalent to $M^\diamond$. □

2.3 Graphical Representation of LAs

A LA can be straightforwardly represented by a classic automaton graph where we label the edge from state s_i to state s_j with $\delta(s_i, \sigma_1, s_j) * \sigma_1, \delta(s_i, \sigma_2, s_j) * \sigma_2, \ldots, \delta(s_i, \sigma_n, s_j) * \sigma_n$ for all σ_i for which $\delta(s_i, \sigma_i, s_j) \geq 1$. If $\delta(s_i, \sigma_i, s_j) = 1$ we omit the multiplicity. A simple example is given in Figure 1. As usual, initial states are marked by a arrow without a source, and final states are depicted as double circle.

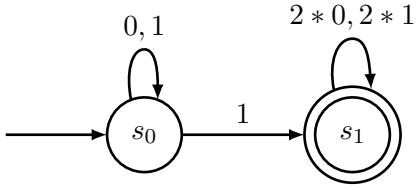

Figure 1: A LA for which $\text{bin}(\#M(x))$ equals the input x without leading zeroes.

3 Basic Properties of NFAs

3.1 Auxiliary Functions

We prove that three auxiliary functions we will need for Lemma 12 are in #NFA.

Lemma 2. *All constant functions, i.e. all functions $f_c : \Sigma^* \to \mathbb{N}$ for some alphabet Σ that satisfy $\forall(x \in \Sigma^*) : (f_c(x) = c)$ for some $c \in \mathbb{N}$, are in $\#\,\text{NFA}$.*

Proof. For the LA $M = (S, \Sigma, \delta, I, F)$ where $S = I = F = \bigcup_{1 \leq i \leq c} \{s_i\}$ and $\delta(s_i, \sigma, s_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases}$ it evidently holds that $\forall i : \xi(s_i, x) = 1$, and thus $\#M(x) = \sum_{s \in F} \xi(s, x) = c \cdot 1 = c$. □

Lemma 3. *For $a_1, ..., a_m \in \{0,1\}$ and $x_1, ..., x_m \in \mathbb{N}$ there is a NFA M for which $\#M(\#a_1 \operatorname{bin}(x_1)...\#a_m \operatorname{bin}(x_m)) = \sum_{i=1}^{m} a_i x_i$.*

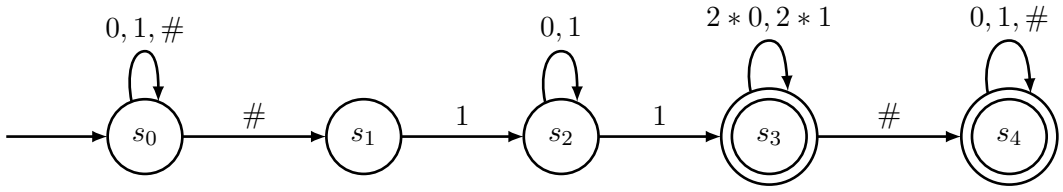

Figure 2: A LA for the index set function.

Proof. Consider the linear automaton M given in Figure 2. We only discuss inputs of the form $\#a_1 \operatorname{bin}(x_1)...\#a_m \operatorname{bin}(x_m)$. Obviously, we have $\xi(s_0, x) = 1$ for all $x \in \{0,1,\#\}^*$. Now we prove by induction on m the claims $\xi(s_1, \#...a_m \operatorname{bin}(x_m)) = 0$, $\xi(s_2, \#...a_m \operatorname{bin}(x_m)) = a_m$, $\xi(s_3, \#...a_m \operatorname{bin}(x_m)) = a_m x_m$ and $\xi(s_4, \#...a_m \operatorname{bin}(x_m)) = \sum_{i=0}^{m-1} a_i x_i$ where $a_0 = x_0 = 0$. For $m = 0$ it is

$$\xi(s_1, \varepsilon) = 0 \tag{3.1}$$
$$\xi(s_2, \varepsilon) = 0 = a_0 \tag{3.2}$$
$$\xi(s_3, \varepsilon) = 0 = a_0 x_0 \tag{3.3}$$
$$\xi(s_4, \varepsilon) = 0 = \sum_{i=0}^{-1} a_i x_i \tag{3.4}$$

For $(m+1) > 1$ we get for the symbol $\#$

$$\xi(s_1, \#... \operatorname{bin}(x_m)\#) = \xi(s_0, \#...\#a_m \operatorname{bin}(x_m)) = 1 \tag{3.5}$$
$$\xi(s_2, \#... \operatorname{bin}(x_m)\#) = 0 \tag{3.6}$$
$$\xi(s_3, \#... \operatorname{bin}(x_m)\#) = 0 \tag{3.7}$$
$$\xi(s_4, \#... \operatorname{bin}(x_m)\#) = \xi(s_4, \#...\#a_m \operatorname{bin}(x_m)) + \xi(s_3, \#...\#a_m \operatorname{bin}(x_m)) \tag{3.8}$$
$$\stackrel{\text{ih}}{=} a_m x_m + \sum_{i=0}^{m-1} a_i x_i = \sum_{i=0}^{m} a_i x_i \tag{3.9}$$

We add a_{m+1}

$$\xi(s_1, \#...\#a_{m+1}) = 0 \tag{3.10}$$
$$\xi(s_2, \#...\#a_{m+1}) = a_{m+1} \cdot \xi(s_1, \#... \operatorname{bin}(x_m)\#) + \xi(s_2, \#... \operatorname{bin}(x_m)\#) = a_{m+1} \tag{3.11}$$
$$\xi(s_3, \#...\#a_{m+1}) = 2 \cdot \xi(s_3, \#... \operatorname{bin}(x_m)\#) + a_{m+1} \cdot \xi(s_2, \#... \operatorname{bin}(x_m)\#) = 0 \tag{3.12}$$
$$\xi(s_4, \#...\#a_{m+1}) = \xi(s_4, \#... \operatorname{bin}(x_m)\#) = \sum_{i=0}^{m} a_i x_i \tag{3.13}$$

Now we can use a subinduction on $|\operatorname{bin}(x_{m+1})|$ to prove that $\xi(s_1, \#...\#a_{m+1}\operatorname{bin}(x_{m+1})) = 0$, $\xi(s_2, \#...\#a_{m+1}\operatorname{bin}(x_{m+1})) = a_{m+1}$, $\xi(s_3, \#...\#a_{m+1}\operatorname{bin}(x_{m+1})) = a_{m+1}x_{m+1}$ and $\xi(s_4, \#...\#a_{m+1}\operatorname{bin}(x_{m+1})) = \sum_{i=1}^{m} a_i x_i$.

For $\operatorname{bin}(x_{m+1}) = \varepsilon$ we get

$$\xi(s_1, \#...a_{m+1}) = 0 \tag{3.14}$$

$$\xi(s_2, \#...a_{m+1}) = a_{m+1} \tag{3.15}$$

$$\xi(s_3, \#...a_{m+1}) = 0 = a_{m+1}x_{m+1} \tag{3.16}$$

$$\xi(s_4, \#...a_{m+1}) = \sum_{i=0}^{m} a_i x_i \tag{3.17}$$

And for $|\operatorname{bin}(x_{m+1})\sigma| > 0$ (where $\sigma \in \{0, 1\}$)

$$\xi(s_1, \#...\operatorname{bin}(x_{m+1})\sigma) = 0 \tag{3.18}$$

$$\xi(s_2, \#...\operatorname{bin}(x_{m+1})\sigma) = \sigma \cdot \xi(s_1, \#...\operatorname{bin}(x_{m+1})) + \xi(s_2, \#...\operatorname{bin}(x_{m+1})) \tag{3.19}$$

$$\stackrel{\text{ih}}{=} a_{m+1} \tag{3.20}$$

$$\xi(s_3, \#...\operatorname{bin}(x_{m+1})\sigma) = \xi(s_3, \#...\operatorname{bin}(2 \cdot x_{m+1} + \sigma)) \tag{3.21}$$

$$= 2 \cdot \xi(s_3, \#...\operatorname{bin}(x_{m+1})) + \sigma \cdot \xi(s_2, \#...\operatorname{bin}(x_{m+1})) \tag{3.22}$$

$$\stackrel{\text{ih}}{=} 2 \cdot a_{m+1}x_{m+1} + \sigma \cdot a_{m+1} = a_{m+1}(2 \cdot x_{m+1} + \sigma) \tag{3.23}$$

$$\xi(s_4, \#...\operatorname{bin}(x_{m+1})\sigma) = \xi(s_4, \#...\operatorname{bin}(x_{m+1})) = \sum_{i=0}^{m} a_i x_i \tag{3.24}$$

Based on the result of the induction we can now conclude that

$$\#M(\#...\#a_m\operatorname{bin}(x_m)) = \xi(s_3, \#...\#a_m\operatorname{bin}(x_m)) + \xi(s_4, \#...\#a_m\operatorname{bin}(x_m)) \tag{3.25}$$

$$= a_m x_m + \sum_{i=0}^{m-1} a_i x_i = \sum_{i=0}^{m} a_i x_i = \sum_{i=1}^{m} a_i x_i \tag{3.26}$$

□

Lemma 4. *The functions* $\mathrm{el}_c : \Sigma^* \to \mathbb{N}$ *with* $\mathrm{el}_c(x) = c^{|x|}$ *for an arbitrary alphabet* Σ *and a constant* $c \in \mathbb{N}$ *are in* $\#\,\mathrm{NFA}$.

Proof. The counting function of the automaton $M = (S, \Sigma, \delta, I, F)$ where $S = I = F = \{s_0\}$ and $\forall(\sigma \in \Sigma) : \delta(s_0, \sigma, s_0) = c$ evidently equals el_c. □

Lemma 5. *For all regular sets* A *the characteristic function* χ_A *is in* $\#\,\mathrm{NFA}$.

Proof. Using the usual techniques for decision DFAs and NFAs (cf. [HMU07] 3.2.3 and 2.3.5) we can easily obtain a DFA that decides the set A. If we regard this deterministic automaton as a LA we get the automaton M which computes

$$\#M(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} = \chi_A(x) \tag{3.27}$$

□

3.2 Closure Properties

We will also need the closure of # NFA under addition and multiplication. The lemmas are stated in the paper from Rich and Slutzki.

Lemma 6 ([RS88])**.** # NFA *is closed under addition, that is, for two automata* M, M' *with the same alphabet* Σ *there is an automaton* M^+ *satisfying* $\#M^+ = \#M + \#M'$.

Lemma 7 ([RS88])**.** # NFA *is closed under multiplication, that is, for two automata* M, M' *with the same alphabet* Σ *there is an automaton* $M^\star$ *satisfying* $\#M \cdot \#M' = \#M^\star$.

From Lemma 2, Lemma 6, and Lemma 7 we conclude

Corollary 8. *If a set* A *is in* range(# NFA)*, then all* $A_m^k = \{k \cdot x + m \mid x \in A\}$ *for* $k, m \in \mathbb{N}$ *are in* range(# NFA).

4 A NFA with an NP-Complete Range

In this section we prove that there is a NFA with an NP-complete range, starting with a short summary of related work concerning these ranges.

4.1 Related Work

In [RS88], counting functions for NFAs (cf. Section 2.1) and DFAs (every state-symbol-tuple is assigned an output word that are concatenated when reading the input) are introduced. Based on previous work ([GG67]), which shows that all ranges of DFAs are regular, the authors prove that range(# DFA) $\subsetneq$ range(# NFA) by constructing an equivalent NFA for every DFA, and giving a NFA whose range is not context-free. They also give an algorithm that decides the range of a NFA in NSPACE(n), and conjecture that there might be NFAs whose ranges are complete for NSPACE(n) or NP.

4.2 The Proof

The idea is to reduce the subset sum (SS) problem, a classic NP-complete decision problem, to a set in range(# NFA). Definitions for SS can be found in [CLRS01] 35.5, a proof of its NP-completeness in [CLRS01] 34.5.5.

Definition (Subset sum problem)**.** *The subset sum problem is given by*

$$\mathrm{SS} = \left\{ (x_1, ..., x_m, S) \mid S \in \mathbb{N} \wedge \forall i : (x_i \in \mathbb{N}) \wedge \exists (a_1, ..., a_m \in \{0,1\}) : \left(\sum_{i=1}^{m} a_i x_i = S \right) \right\} \tag{4.1}$$

The range $\mathrm{SS_b}$ we want to construct is such that $\mathrm{bin}(\mathrm{SS_b})$ equals

$$\left\{ \mathrm{bin}\left(\sum_{i=1}^{m} a_i x_i \right) 00 \, \mathrm{code}(\#\#\,\mathrm{bin}(x_1) ... \#\#\,\mathrm{bin}(x_m)) \mid \forall i : (a_i \in \{0,1\} \wedge x_i \in \mathbb{N}) \right\} \cup \{\varepsilon\} \tag{4.2}$$

where code is a 2-bit block code defined by code(0) = 01, code(1) = 10 and code(#) = 11.

Lemma 9. SS_b *is NP-hard.*

Proof. We show that SS can be reduced to $\mathrm{SS_b}$ in logarithmic space. Consider the function f defined by

$$\mathrm{bin}(f(x)) = \begin{cases} \mathrm{bin}(S)00\,\mathrm{code}(\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_m)) & \text{if } x = (x_1, ..., x_m, S) \text{ for } x_i, S \in \mathbb{N} \\ 1 & \text{else} \end{cases} \tag{4.3}$$

f reduces SS to $\mathrm{SS_b}$ since

$$x \in \mathrm{SS} \Leftrightarrow x = (x_1, ..., x_m, S) \text{ for some } x_i, S \in \mathbb{N} \tag{4.4}$$

$$\text{and } \sum_{i=1}^{m} a_i x_i = S \text{ for some } a_1, ..., a_m \in \{0, 1\} \tag{4.5}$$

$$\Leftrightarrow {}^{3}\,\mathrm{bin}(f(x)) = \mathrm{bin}(S)00\,\mathrm{code}(\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_m)) \tag{4.6}$$

$$\text{where } \sum_{i=1}^{m} a_i x_i = S \text{ for some } a_1, ..., a_m \in \{0, 1\} \tag{4.7}$$

$$\Leftrightarrow {}^{4} f(x) \in \mathrm{SS_b} \tag{4.8}$$

The reduction f is easily computable in logarithmic space, as the set of all tuples $(x_1, ..., x_m, S)$ for some $x_i, S \in \mathbb{N}$ (in a usual coding such as $\mathrm{bin}(x_1)\#...\#\,\mathrm{bin}(x_m)\#S$) is regular and therefore decidable without any space, and the rest is a simple rearrangement. □

Lemma 10. SS_b *is in NP.*

Proof. A decision algorithm for NP accepts if the input equals the empty word. It can simulate a deterministic finite automaton to check whether the input equals $\mathrm{bin}(S)00\,\mathrm{code}(\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_m))$ for some $S, x_i \in \mathbb{N}$ and reject if not. Then it guesses nondeterministically $a_1, ..., a_m \in \{0, 1\}$ and accepts if $(\sum_{i=1}^{m} a_i x_i) = S$. □

For Lemma 12 we will need an auxiliary lemma.

Lemma 11. *For* $a_1, ..., a_m \in \{0, 1\}$ *and* $x_1, ..., x_m \in \mathbb{N}$ *there is a NFA* M *for which* $\mathrm{bin}(\#M(\#a_1\,\mathrm{bin}(x_1)...\#a_m\,\mathrm{bin}(x_m))) = \mathrm{code}(\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_m))$.

Proof. Consider the LA given in Figure 3. Obviously, we have $\xi(s_0, \#...\,\mathrm{bin}(x_{m-1})\#) = 0$, $\xi(s_0, \#...\,\mathrm{bin}(x_{m-1})\#a_m) = \xi(s_0, \#...\#a_m\,\mathrm{bin}(x_m)) = 1$, $\xi(s_1, \#...\,\mathrm{bin}(x_{m-1})\#) = 1$ and $\xi(s_1, \#...\,\mathrm{bin}(x_{m-1})\#a_m) = \xi(s_1, \#...\#a_m\,\mathrm{bin}(x_m)) = 0$ for all m and lengths of

[4] Note that $\forall x_1, ..., x_m, S \in \mathbb{N} : \mathrm{bin}(S)00\,\mathrm{code}(\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_m)) \neq 1$.

[5] Note that $\forall x : (f(x) \neq 0)$.

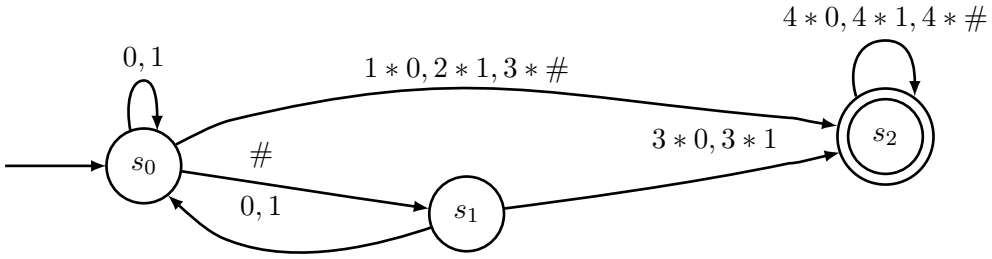

Figure 3: A LA for Lemma 11.

$\mathrm{bin}(x_m)$. We can prove by induction on m that $\xi(s_2, \#a_1\,\mathrm{bin}(x_1)...\#a_m\,\mathrm{bin}(x_m)) = \mathrm{bin}^{-1}(\mathrm{code}(\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_m)))$. For $m = 0$ we have

$$\xi(s_2, \varepsilon) = 0 = \mathrm{bin}^{-1}(\mathrm{code}(\varepsilon)) \tag{4.9}$$

For $(m+1) > 0$ we get for the first two symbols

$$\begin{aligned}
\xi(s_2, \#...\#a_{m+1}) &= 4 \cdot \xi(s_2, \#...\,\mathrm{bin}(x_m)\#) + (a_{m+1}+1) \cdot \xi(s_0, \#...\,\mathrm{bin}(x_m)\#) && (4.10)\\
&\quad + 3 \cdot \xi(s_1, \#...\,\mathrm{bin}(x_m)\#) && (4.11)\\
&= 4 \cdot (4 \cdot \xi(s_2, \#...\,\mathrm{bin}(x_m)) + 3 \cdot \xi(s_0, \#...\,\mathrm{bin}(x_m)) && (4.12)\\
&\quad + 3 \cdot \xi(s_1, \#...\,\mathrm{bin}(x_m))) + 3 && (4.13)\\
&\overset{\text{ih}}{=} 4 \cdot (4 \cdot \mathrm{bin}^{-1}(\mathrm{code}(\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_m))) + 3) + 3 && (4.14)\\
&= \mathrm{bin}^{-1}(\mathrm{code}(\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_m)\#\#)) && (4.15)
\end{aligned}$$

By a subinduction on $|\,\mathrm{bin}(x_{m+1})|$ we show that $\xi(s_2, \#...\#a_{m+1}\,\mathrm{bin}(x_{m+1})) = \mathrm{bin}^{-1}(\mathrm{code}(\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_{m+1})))$.

For $|\,\mathrm{bin}(x_{m+1})| = 0$ we have

$$\xi(s_2, \#...\#a_{m+1}\varepsilon) = \mathrm{bin}^{-1}(\mathrm{code}(\#\#...\,\mathrm{bin}(x_m)\#\#\varepsilon)) \tag{4.16}$$

And for $|\,\mathrm{bin}(2 \cdot x_{m+1} + \sigma)| > 0$ where $\sigma \in \{0, 1\}$ we get

$$\begin{aligned}
\xi(s_2, \#...\,\mathrm{bin}(2 \cdot x_{m+1} + \sigma) &= 4 \cdot \xi(s_2, \#...\,\mathrm{bin}(x_{m+1})) && (4.17)\\
&\quad + (\sigma+1) \cdot \xi(s_0, \#...\,\mathrm{bin}(x_{m+1})) && (4.18)\\
&\quad + 3 \cdot \xi(s_1, \#...\,\mathrm{bin}(x_{m+1})) && (4.19)\\
&\overset{\text{ih}}{=} 4 \cdot \mathrm{bin}^{-1}(\mathrm{code}(\#\#...\,\mathrm{bin}(x_{m+1}))) + (\sigma+1) && (4.20)\\
&= \mathrm{bin}^{-1}(\mathrm{code}(\#\#...\#\#\,\mathrm{bin}(2 \cdot x_{m+1} + \sigma))) && (4.21)
\end{aligned}$$

□

Lemma 12. SS_b *is in* range(# NFA).

Proof. Consider the function $g : \{0,1,\#\}^* \to \mathbb{N}$ given by

$$\operatorname{bin}(g(\#a_1...\#a_m \operatorname{bin}(x_m))) = \operatorname{bin}\left(\sum_{i=1}^{m} a_i x_i\right) 00 \operatorname{code}(\#\# \operatorname{bin}(x_1)...\#\# \operatorname{bin}(x_m)) \quad (4.22)$$

for inputs of the form $\#a_1 \operatorname{bin}(x_1)...\#a_m \operatorname{bin}(x_m)$ for some $a_i \in \{0,1\}, x_i \in \mathbb{N}$, and for all other inputs x by $g(x) = 0$. Note that $\operatorname{range}(g) = \mathrm{SS_b}$. So it suffices to prove that g is the counting funtion of a NFA.

Let $A = \{\#a_1 \operatorname{bin}(x_1)...\#a_m \operatorname{bin}(x_m) \mid \forall(0 \le i \le m) : (a_i \in \{0,1\} \wedge x_i \in \mathbb{N})\}$. Evidently, A is the set of words over $\{0,1,\#\}$ which match the regular expression $R = (\#(0+1)(\varepsilon + 1(0+1)^*))^*$ and therefore regular. Additionally, we define the functions $h_1 : \{0,1,\#\}^* \to \{0,1\}^*$ and $h_2 : \{0,1,\#\}^* \to \{0,1\}^*$ for words from A by[5]

$$\operatorname{bin}(h_1(\#a_1...\#a_m \operatorname{bin}(x_m))) = \operatorname{bin}\left(\sum_{i=1}^{m} a_i x_i\right) \quad (4.23)$$

$$\operatorname{bin}(h_2(\#a_1...\#a_m \operatorname{bin}(x_m))) = \operatorname{code}(\#\# \operatorname{bin}(x_1)...\#\# \operatorname{bin}(x_m)) \quad (4.24)$$

Note that $|h_2(x)| = 2 \cdot |x|$ for all $x \in A$. We can now express $g(x)$ using these functions.

$$g(x) = \chi_A(x) \cdot \Big(h_1(x) \cdot 4 \cdot 4^{|x|} + h_2(x)\Big) \quad (4.25)$$

And as $\chi_A \in \#\,\mathrm{NFA}$ (Lemma 5), $h_1 \in \#\,\mathrm{NFA}$ (Lemma 3), $4^{|x|} \in \#\,\mathrm{NFA}$ (Lemma 4), $h_2 \in \#\,\mathrm{NFA}$ (Lemma 11), the constant function $x \mapsto 4$ is in $\#\,\mathrm{NFA}$ (Lemma 2) and $\#\,\mathrm{NFA}$ is closed under addition and multiplication (Lemmas 6, 7), g is in $\#\,\mathrm{NFA}$. □

Lemmas 9, 10 and 12 immediately prove the conjecture from [RS88].

Theorem 13. $\operatorname{range}(\#\,\mathrm{NFA})$ *contains NP-complete sets.*

5 Transfering the Result to Probabilistic Automata

We will now prove a similar corollary for *probabilistic finite automata* (*PFAs*). A (rational) PFA is a 5-tuple $P = (S, \Sigma, \delta, \pi, F)$, where $S = \{s_1, s_2, ..., s_{|S|}\}$ is the finite set of *states*, Σ is the finite *input alphabet*, $\delta : S \times \Sigma \times S \to \mathbb{Q}$[6] where $\forall(s \in S)\forall(\sigma \in \Sigma)\forall(s' \in S) : (\delta(s,\sigma,s') \ge 0)$ and $\forall(s \in S)\forall(\sigma \in \Sigma) : \left(\sum_{s' \in S} \delta(s,\sigma,s') = 1\right)$ is the *transition function*, $\pi = (\pi_1, \pi_2, ..., \pi_{|S|}) \in \mathbb{Q}^{|S|}$ where $\sum_{1 \le i \le |S|} \pi_i = 1$ is the *initial state vector* and $F \subseteq S$ is the set of *final states*. Informally, $\delta(q,\sigma,p)$ is the probability that the state p will be assumed if the current state is q and input symbol is σ.

The probability that P is in state s_i having read the input x, is defined by

$$\rho(s_i, \varepsilon) = \pi_i \quad (5.1)$$

$$\rho(s_i, x\sigma) = \sum_{s' \in S} \big(\rho(s', x) \cdot \delta(s', \sigma, s_i)\big) \quad (5.2)$$

[5] The results for all other inputs are arbitrary.

[6] As usual, $\mathbb{Q}$ is the set of rational numbers.

The *acceptance probability* of P on the input x is given by

$$P(x) = \sum_{s \in F} \rho(s, x) \tag{5.3}$$

The equivalence of this definition to the usual matrix notation (e.g. [Mac94]) can again be proved analogously to [RS88] Lemma 4.1. Additionally, we define the range of a PFA P by $\text{range}(P) = \{P(x) \mid x \in \Sigma^*\}$.

We can now link LAs to PFAs.

Lemma 14. *For every LA M there is a PFA P and a constant k with*

$$\forall x : P(x) = \frac{\#M(x)}{k^{|x|+1}} \tag{5.4}$$

Proof. The idea is to choose a denominator for all transitions in the PFA high enough such that we can use the multiplicity from the transitions of the LA as the numerator. This necessitates a new non-accepting state $s_{|S|+1}$ for the excess probability which we don't need for the simulation.

For a given LA $M = (S = \{s_1, s_2, ..., s_{|S|}\}, \Sigma, \delta, I, F)$ let

$$k = \max\left(\max_{\sigma \in \Sigma}\left(\sum_{s,s' \in S} \delta(s, \sigma, s')\right) + 1, |I| + 1\right) \tag{5.5}$$

Consider the PFA $P = (S_P, \Sigma_P, \delta_P, \pi_P, F_P)$ with

$$S_P = S \cup \{s_{|S|+1}\} \tag{5.6}$$

$$\Sigma_P = \Sigma \tag{5.7}$$

$$\delta_P(s, \sigma, s') = \begin{cases} \delta(s, \sigma, s')/k & \text{if } s, s' \in S \\ 1 - \sum_{s' \in S} \left(\delta(s, \sigma, s')/k\right) & \text{if } (s \in S) \wedge (s' = s_{|S|+1}) \\ 1 & \text{if } (s = s_{|S|+1}) \wedge (s' = s) \\ 0 & \text{if } (s = s_{|S|+1}) \wedge (s' \neq s) \end{cases} \tag{5.8}$$

$$\pi_{Pi} = \begin{cases} 1/k & \text{if } s_i \in I \\ 1 - \sum_{1 \leq i \leq |S|} \pi_{Pi} & \text{if } s_i = s_{|S|+1} \\ 0 & \text{else} \end{cases} \tag{5.9}$$

$$F_P = F \tag{5.10}$$

Note that that $\left(1 - \sum_{s' \in S} \left(\delta(s, \sigma, s')/k\right)\right) \geq 0$ and $\left(1 - \sum_{1 \leq i \leq |S|} \pi_{Pi}\right) \geq 0$ due to the definition of k. By an induction on the length of the input x one can easily prove that

$$\rho_P(s_i, x) = \frac{\xi(s_i, x)}{k^{|x|+1}} \tag{5.11}$$

The conditions for δ_P and π_P are satisfied by their definitions. □

Corollary 15. *There exist PFAs whose range is NP-complete.*

Proof. Consider the PFA $P = (S_P, \Sigma_P, \delta_P, \pi_P, F_P)$ we get from Lemma 14 for the set $\mathrm{SS}_{\mathrm{b}1}^k$ from Corollary 8, where SS_{b} is from Section 4.2 and k from Lemma 14. We need Corollary 8 since two inputs x, x' where $\#M(x') = \#M(x) \cdot k^n$ for some $n \neq 0$ lead to the same $P(x) = P(x')$ even though $\#M(x') \neq \#M(x)$ if $|x'| = |x| + n$.

We can now reduce the subset sum problem to range(P) by creating a new reduction f' based on the function f from Lemma 9. If the input to f' is not of the form $(x_1, ..., x_m, S)$, output 1. $1 \notin \mathrm{range}(P)$ since $(P(x) = 1) \Rightarrow (\rho(s_{|S|+1}, x) = 0) \Rightarrow (\pi_{P|S|+1} = 0)$ and thats not possible as

$$\sum_{1 \leq i \leq |S|} \pi_{Pi} = \sum_{s_i \in I} \frac{1}{k} = \frac{|I|}{k} \leq \frac{k-1}{k} < 1 \tag{5.12}$$

and therefore $\pi_{P|S|+1} = 1 - \left(\sum_{1 \leq i \leq |S|} \pi_{Pi}\right) > 0$. This format check can be done in linear time, for example by simulating a finite automaton.

If the input x equals the tuple $(x_1, ..., x_m, S)$ for some $x_i, S \in \mathbb{N}$, then calculate $l = |\#\#\,\mathrm{bin}(x_1)...\#\#\,\mathrm{bin}(x_m)|$, which is the length of the input to M that leads to the acceptance of $f(x)$ if $f(x)$ accepted by M. We need linear time for this. Now the reduction outputs

$$\frac{f(x) \cdot k + 1}{k^{l+1}} \tag{5.13}$$

(k is from Lemma 14), in polynomial time since f is computable in logarithmic space. Notice that the numerator is not divisible by k and it therefore holds that

$$\left(\frac{f(x) \cdot k + 1}{k^{l+1}} = \frac{f(x') \cdot k + 1}{k^{l+1}}\right) \Leftrightarrow ((f(x) = f(x')) \wedge (l = l')) \tag{5.14}$$

Hence we get

$$\begin{aligned}
(x_1, ..., x_m, S) \in \mathrm{SS} &\Leftrightarrow f(x_1, ..., x_m, S) \in \mathrm{SS}_{\mathrm{b}} \qquad \text{(Lemma 9)} && (5.15)\\
&\Leftrightarrow f(x_1, ..., x_m, S) \cdot k + 1 \in \mathrm{SS}_{\mathrm{b}1}^k && (5.16)\\
&\Leftrightarrow f'(x_1, ..., x_m, S) \in \mathrm{range}(P) && (5.17)
\end{aligned}$$

This proves the NP-hardness.

For an $\frac{x}{y} \in \mathbb{Q}$ a decision algorithm for range(P) rejects if x is not of the form $k \cdot x' + 1$ for some $x' \in \mathbb{N}$. Otherwise, it inputs $\frac{x-1}{k}$ in the algorithm from Lemma 10. Thus we can nondeterministically decide range(P) in polynomial time. □

6 Summary and Open Questions

In Section 4 we proved that there is a NP-complete set in range(# NFA). As the current nondeterministic upper space bound for range(# NFA) is NSPACE(n) (cf. 4.1) that leaves room for a more difficult complete set in range(# NFA) as well as for a lower space bound.

A proof of range(# NFA) $\subseteq$ NP might also be possible, but nontrivial as there are automata M for which $\#M(x) \in \Theta(\log(x))$ and therefore we cannot guess nondeterministically all inputs that could yield a given output y in a time polynomial in $|y|$.

In Section 5 we showed that there is a PFA P for which range(P) is NP complete. Again, it remains to show whether or not range(P) $\subseteq$ NP or if range(P) contains sets more difficult then NP.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 2009.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2nd revised edition, 2001.

[GG67] Seymour Ginsburg and Sheila A. Greibach. Abstract families of languages. In *Conference Record of 1967 Eighth Annual Symposium on Switching and Automata Theory, 18-20 October 1967, Austin, Texas, USA*, 1967.

[HMU07] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Pearson/Addison Wesley, 2007.

[Mac94] Ioan I. Macarie. Space-efficient deterministic simulation of probabilistic automata (extended abstract). In Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner, editors, *STACS*, volume 775 of *Lecture Notes in Computer Science*, pages 109–122. Springer, 1994.

[RS88] Craig A. Rich and Giora Slutzki. The complexity of a counting finite-state automaton. In *Proceedings of the Eighth Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 225–239, London, UK, 1988. Springer.

[Tod91] Seinosuke Toda. Pp is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991.