Comparing Counting Classes for Logspace, One-way Logspace, Logtime, and First-Order

Hans-Jörg Burtschick
TU - Berlin, Fachberich Informatik
Franklinstr. 28/29
D-10587 Berlin

(revised version)

Abstract

We generalize the definition of first-order counting classes [SST92] to use $<$, $SUCC$, and $+$ as linear orderings. It turns out that $\#\Pi_2[<] = \#\Pi_1[\mathrm{SUCC}] = \#\Pi_1[+]$.

We introduce certain classes of logtime counting functions and show that the classes of first-order definable counting functions are subclasses of the corresponding logtime counting classes.

These logtime counting classes are itself subclasses of the corresponding one-way logspace counting classes. These logspace counting classes form a strict hierachy within #P:

$$\mathrm{F1L} \subsetneq \#\mathrm{1L} \subsetneq \text{span-1L} \subsetneq \#\mathrm{P}.$$

Using the logical characterization of #P we obtain a characterization of #P via universally branching logtime Turing machines.

1 Introduction

An important open question in complexity theory is whether the two classes NL and NP are equal. Although in the case of computing partial multivalued functions nondeterministically the corresponding classes can be separated [Bur89], a solution for the class of decision problems is not in sight.

Counting classes, such as #P [Val79] or the corresponding logspace counting class #L, contain the functions that count the accepting paths of polynomial time bounded (or logarithmic space bounded resp.) nondeterministic Turing machines. Thus they may be seen as intermediate between NP and NPMV. That is, the complexity of 'counting the number of solutions' may be seen as between that of 'computing the solution' and that of 'checking the existence of a solution'. But, as for NL and NP, the question whether #L equals #P is open.

Considering this, the existence of strict subclasses of #P, proven by Saluja, Subrahmanyam, and Thakur [SST92], is surprising. Their results use notions from finite model theory to obtain logical characterizations of complexity classes. This line of research was initiated by [Fag74]. In [SST92] the descriptional complexity of counting functions is considered. The number of alternations of first-order quantifiers in formulas is used as a measure of the complexity. For example, formulas of the form $\forall\vec{x}\exists\vec{y}\,\varphi(\vec{x},\vec{y})$ are called Π_2-formulas. The class $\#\Pi_2$ contains the counting functions that are definable by Π_2-formulas (see Section 2 for precise definitions). One of their main results is: $\#\Sigma_0 \subsetneq \#\Sigma_1 \subsetneq \#\Pi_1 \subsetneq \#\Sigma_2 \subsetneq \#\Pi_2\ (= \#\mathrm{P})$.

There are functions in $\#\Sigma_2 \setminus \#\Pi_1$ that are computable by deterministic logspace bounded Turing machines. On the other hand, $\#\Pi_1$ contains a function that is complete for #P via logspace reductions that preserve the number of solutions. Thus at first sight, these results seem not to fit into the structure of more traditional Turing machine based counting classes.

The aim of this paper is to compare logically characterized counting classes to counting classes based on Turing machines.

In section 3 we consider one-way logspace counting classes. A one-way Turing machine scans its input only once (from left to right, say). Studying this type of Turing machines has a long tradition in theoretical computer science [HU69] [HIM78]. It is obvious that one can define the counting classes #1L and span-1L via one-way logspace Turing machines, similar to the well-known classes #L and span-L. In [AJ93] it is shown that 1UL = 1NL $\Leftrightarrow$ #1L = span-1L $\Leftrightarrow$ #1L $\subseteq$ opt-1L. From a result in communication complexity [MS82] it follows that 1UL$\subsetneq$ 1NL. Thus we can conclude that #1L$\subsetneq$ span-1L and F1L$\subsetneq$ #L. Furthermore, all the one-way logspace counting classes are strictly included in the corresponding (two-way) logspace counting classes.

We consider in Section 4 logically characterized counting classes. First, we generalize the notion of first-order counting function as defined in [SST92] with respect to the ordering. First-order counting functions map ordered finite structures to nonnegative integers. It is crucial here that the structure is ordered. If isomorphic copies of the same solution are indistinguishable, the counting function will not only count the distinct solutions, but also all isomorphic copies of the each solution! There are interesting results in the case of counting functions on unordered structures (see [CG94]). But for the comparison of first-order and Turing machine based counting classes it is useful to consider ordered structures. In [SST92] a linear ordering is used. If we look at Turing machines (and thus at strings) it seems artificial, not to consider the succ relation or the plus relation as orderings as well. For $k \in \mathrm{N}$ and $op \in \{<, SUCC, +\}$ we therefore define the classes $\#\Sigma_k[op]$ and $\#\Pi_k[op]$ of counting functions definable by first-order Π_k and Σ_k formulas using op as linear ordering. The classes $\#\Sigma_k[<]$ and $\#\Pi_k[<]$ coincide with $\#\Sigma_k$ and $\#\Pi_k$.

In [SST92] it is shown that $\#\Pi_1[<] \subsetneq \#\Sigma_2[<] \subsetneq \#\Pi_2[<] = \#\mathrm{P}$ which is surprising, since $\#3SAT$ (a #P complete function) is contained in $\#\Pi_1[<]$. But these results depend heavily on the use of the linear order instead of the $SUCC$ relation or the + relation. It turns out that $\#\Pi_2[<] = \#\Pi_1[\mathrm{SUCC}] = \#\Pi_1[+]$. (It follows, that $\#\Pi_1[+] = \#\Pi_2[+]$. This may also be seen as an answer to the question posed in [Lyn90], whether $\Sigma_1^1\Pi_1^0[+]$ is strictly included in $\Sigma_1^1\Pi_2^0[+]$.)

In order to compare first-order and Turing machine based counting functions we have to take into account that they differ in their domain. In [SST92] the authors redefine #P such that the polynomial time bounded Turing machines have ordered finite structures as input. We want to keep the definition of #P to be based on strings. Thus we have to encode finite ordered structures in strings and vice versa. We define a natural encoding of finite ordered structures into strings. Under that encoding, the class $\#\Sigma_0[op]$ is a subclass of #1L and $\#\Sigma_1[op]$ is a subclass of span-1L. Since this encoding is one-one but not onto, the inclusions are strict by definition. But from results in [SST92] it follows that #1L and $\#\Sigma_0[<]$, and also span-1L and $\#\Sigma_1[<]$, are separated even in the range of the encoding.

Based on Fagins Theorem [Fag74] one can show that (under a natural encoding of strings into finite ordered structures) #P is a subclass of $\#\Pi_1[\mathrm{SUCC}]$. Therefore, the classes $\#\Pi_1[\mathrm{SUCC}]$ and $\#\Pi_1[+]$ may be seen as logical characterizations of #P.

In [BIS90] it is shown, that the class of first-order definable sets equals the logarithmic time hierarchy LH. So this leads obviously to the question whether it is possible to define certain classes of logtime counting functions and to show that these classes are intermediate between the first-order counting classes and the one-way logspace counting classes. We define in Section 5 logtime counting classes. But due to the restricted computational power, we have to extend the usual model to access relational inputs. It turns out that in fact the first-order counting classes are subclasses of the corresponding logtime counting classes. From this, we obtain a characterization of #P via universally branching logtime Turing machines.

In Section 6 we relate the logtime counting classes to the one-way logspace counting classes. We show that the logtime counting classes are subclasses of the corresponding one-way logspace counting classes.

2 Preliminaries

We will use standard notation as e.g. in [BDG89]. We write $\mathbf{N}$ for the set of nonnegative integers. If A is a finite set then $||A||$ denotes its cardinality. For any set A let χ_A be the characteristic function of A, that is, $\chi_A(x) = 1$ if $x \in A$ and $\chi_A(x) = 0$ otherwise. For a string $w \in \Sigma^*$ we write $|w|$ for the length of w. If $i \leq |w|$ we denote the ith bit of w by $w(i)$. Let Σ^n denote the set of strings of length n. We write $bin(n)$ to denote the binary representation of $n \in \mathbf{N}$ and let $bin(k,n)$ denote the binary representation of n with leading zeroes such that $|bin(k,n)| = k$, if $k \geq |bin(n)|$.

Notions from Complexity Theory

We use the standard definitions for the complexity classes L, FL, UL, NL, P, and NP. Let #P (#L) be the class of functions $f : \Sigma^* \to \mathbf{N}$ such that there exists a polynomial time bounded (logarithmic space bounded) NTM M which for all $x \in \Sigma^*$ on input x has exactly $f(x)$ accepting paths. A NTM computes a binary relation R, if $(x,y) \in R$ iff on input x there exists a computation path such that M outputs y and accepts. A relation R is logspace computable if there exists a logspace NTM that computes R. NLMV is the class of logspace computable relations. Let span-L be the class of functions f, such that there exists a logspace computable relation R and for all x it holds that $f(x) = ||\{y : (x,y) \in R\}||$.

A one-way logspace TM (NTM, ATM) is a Turing machine with a logarithmic space bound, a one-way read-only input tape, and a one-way write-only output tape. We follow the definition (as, e.g., in [HIM78]) that $c \cdot \lceil \log(n) \rceil$ tape cells are marked on the work tapes before the beginning of a computation on an input of size n (c is a constant). We define F1L, 1NL, 1UL, #1L, 1NLMV, and span-1L similar to the classes FL, NL, UL, #L, NLMV, and span-L except that we use one-way logspace Turing machines instead of ordinary (two-way) Turing machines.

Notions from Logic

Turing machine based counting functions map strings to natural numbers. The strings can be seen as sequences of symbols. The input objects of logically defined counting functions may have a more complicated internal structure. Depending on the problem, they may consist of a constant number of finite relations of arity greater than one. This internal structure is described in a signature. A signature $\sigma = (R_1, \ldots, R_k, c_1, \ldots, c_l)$ consists of relation symbols $R_1, \ldots, R_k$ with arities $a_1, \ldots, a_k$ and of constant symbols $c_1, \ldots, c_l$. A finite σ-structure $\mathcal{A} = (U^{\mathcal{A}}, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_l^{\mathcal{A}})$ consists of a finite set $U^{\mathcal{A}}$ called the universe of $\mathcal{A}$, relations over $U^{\mathcal{A}}$ corresponding to the relation symbols, and constants from the universe corresponding to the constant symbols in σ. We omit the superscript $\mathcal{A}$ whenever it is clear from the context that we do not mean the relation symbol or the constant symbol. We denote the set of σ-structures by $STRUC[\sigma]$.

Let $\tau = (T_1, \ldots, T_{k'}, d_1, \ldots, d_{l'})$ be a signature such that $\tau \cap \sigma = \emptyset$. A τ-expansion of $\mathcal{A}$ is $\mathcal{T} = (T_1^{\mathcal{A}}, \ldots, T_{k'}^{\mathcal{A}}, d_1^{\mathcal{A}}, \ldots, d_{l'}^{\mathcal{A}})$ and it consists of relations and constants over the universe of $\mathcal{A}$ corresponding to the relation symbols and constant symbols in τ. A τ-expanded σ-structure $(\mathcal{A}, \mathcal{T}) = (U^{\mathcal{A}}, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_l^{\mathcal{A}}, T_1^{\mathcal{A}}, \ldots, T_k'^{\mathcal{A}}, d_1^{\mathcal{A}}, \ldots, d_{l'}^{\mathcal{A}})$ consists of the σ-structure $\mathcal{A}$ and the τ-expansion of $\mathcal{A}$.

Let $\prec^{\mathcal{A}}$ be a linear ordering on $U^{\mathcal{A}}$. We call $(\mathcal{A}, \prec^{\mathcal{A}})$ an ordered σ-structure.

A formula of the form $\exists \vec{x_1} \forall \vec{x_2} \ldots Q\vec{x_k}\ \varphi(\vec{x_1}, \vec{x_2}, \ldots, \vec{x_k})$ $(\forall \vec{x_1} \exists \vec{x_2} \ldots Q\vec{x_k}\ \varphi(\vec{x_1}, \vec{x_2}, \ldots, \vec{x_k}))$ such that $\varphi(\vec{x_1}, \vec{x_2}, \ldots, \vec{x_k})$ is quantifier-free is called a Σ_k-formula (Π_k-formula), if $k \geq 1$. A formula without quantifiers and variables is called a Π_0-formula or Σ_0-formula.

In [SST92] for $k \in \mathrm{N}$ the class $\#\Sigma_k$ $(\#\Pi_k)$ is defined to be the class of functions f such that there exists a Σ_k-formula (Π_k-formula) ϕ over $(\sigma, \tau, <)$ such that $f(\mathcal{A}, \prec^{\mathcal{A}}) = ||\{\mathcal{T} : (\mathcal{A}, \mathcal{T}, \prec^{\mathcal{A}}) \models \phi\}||$ for all ordered σ-structures $(\mathcal{A}, \prec^{\mathcal{A}})$. It is important, that the function depends also on the ordering, since there are $n!$ many linear orderings on a universe of cardinality n. For results on unordered structures see [CG94].

Let $(\mathcal{A}, \prec^{\mathcal{A}})$ and $(\mathcal{A}', \prec^{\mathcal{A}'})$ be ordered σ-structures and $f \in \#\Sigma_k$ (or $f \in \#\Pi_k$). It is easy to see that $f(\mathcal{A}, \prec^{\mathcal{A}}) = f((\mathcal{A}', \prec^{\mathcal{A}'})$ if $(\mathcal{A}, \prec^{\mathcal{A}})$ and $(\mathcal{A}', \prec^{\mathcal{A}'})$ are isomorphic. Therefore we sometimes assume the universe of finite structures of cardinality $n \in \mathrm{N}$ to be $\{0, \ldots, n-1\}$ and the linear ordering to be the $<$ relation on N, restricted to that universe.

In [SST92] it is proven that $\#DEG\_1\_NGB \in \#\Sigma_2 \setminus \#\Pi_1$ and $DIST2 \in \#\Sigma_1 \setminus \#\Sigma_0$, where $\#DEG\_1\_NGB$ counts the nodes of a graph that have a neighbour of degree 1, and $\#DIST2$ counts the pairs of nodes for which the shortest path in the graph is of length two.

Counting operator

A binary relation $R \subseteq \Sigma^* \times \Sigma^*$ is called polynomially length-bounded, if there exists a ploynomial $p : \mathrm{N} \to \mathrm{N}$ such that $|y| \leq p(|x|)$ for all $(x, y) \in R$. Throughout this paper we consider polynomially lentgh-bounded relations, only.

In order to compare Turing machine based and logically defined counting classes, we need an unifying notion of a counting function. Therefore we use counting operators as studied for example in [Vol94a], [Vol94b], or [Bur92]. A survey on this subject can be found in [HV95].

Definition 2.1 *Let $\mathcal{C}$ be a class of polynomially length-bounded binary relations. A function $f : \Sigma^* \to \mathrm{N}$ is contained in $\# \cdot \mathcal{C}$, iff there is a relation $R \subseteq \Sigma^* \times \Sigma^*$ contained in $\mathcal{C}$ such that*

$$f(x) = ||\{y \,:\, (x, y) \in R\}||$$

for all $x \in \Sigma^*$.

It is possible to characterize the known counting classes using this counting operator: $\# \cdot \mathrm{P} = \#\mathrm{P}$ and span-P $= \# \cdot \mathrm{NPMV}$. See for example [Vol94b].

3 Logspace and one-way logspace counting classes

In order to define a counting operator for logspace counting classes, we have to use a computational model that reflects the following fact: A logspace bounded nondeterministic Turing machine cannot access the nondeterministically guessed bits (up to logarithmically many bits stored on the work tapes). We therefore use *one-way protocol machines* defined by K.-J. Lange [Lan86]. We call these Turing machines 2-1-, and 1-1-Turing machines to avoid confusion in the definition of one-way logspace counting classes.

Definition 3.1 *A 2-1-TM (2-1-ATM) is a Turing machine (alternating Turing machine) with two input tapes. The first is a two-way and the second is a one-way input tape. A 2-1-Turing machine M accepts a binary Relation R, iff M accepts all $(x, y) \in R$, such that x is on the first input tape and y is on the second input tape. We call x the first input and y the second input.*

Analogously, a logspace 1-1-TM (1-1-ATM) is a one-way logspace Turing machine with two one-way input tapes.

For $k > 0$ we call a 2-1-ATM (1-1-ATM) that makes at most $k - 1$ alternations a 2-1-Σ_k-TM (1-1-Σ_k-TM) if it starts in an existential configuration and a 2-1-Π_k-TM (1-1-Π_k-TM) if it starts in an universal configuration.

A deterministic 2-1-TM (1-1-TM) is called a 2-1-Σ_0-TM or 2-1-Π_0-TM (1-1-Σ_0-TM or 1-1-Π_0-TM resp.)

Definition 3.2 *For each $k \in \mathrm{N}$ let 2-1-$\Sigma_k L$ (2-1-$\Pi_k L$, 1-1-$\Sigma_k L$, 1-1-$\Pi_k L$) be the class of polynomially length-bounded binary relations that are accepted by logspace 2-1-Σ_k-Turing machines (2-1-Π_k-Turing machines, 1-1-Σ_k-Turing machines, 1-1-Π_k-Turing machines resp.).*

First, we show that every binary relation R is logspace computable, iff it is acceptable by a logspace 2-1-Σ_1-TM.

Lemma 3.3 *1.* NLMV = *2-1-$\Sigma_1 L$* *2.* 1NLMV = *1-1-$\Sigma_1 L$*

Proof.(sketch) (1.) (NLMV $\subseteq$ 2-1-Σ_1L): Let $R \in$ NLMV and M_R be the logspace NTM that computes R. A logspace 2-1-Σ_1-TM M simulates M_R and checks bit by bit whether the output of M_R and the second input of M is equal.

(2-1-Σ_1L $\subseteq$ NLMV): Let $R \in$ 2-1-Σ_1L and M_R be the 2-1-Σ_1-TM that accepts R. A logspace NTM simulates M_R and guesses each bit of the second input of M_R and writes it on the output tape. M accepts iff M_R accepts.
(2.) The proof is similar to (1.) since the simulations do not depend on the head-moves. $\square$

Remark. Let $F = (a_1 \vee a_{n+1}) \wedge (a_2 \vee a_{n+2}) \wedge \ldots \wedge (a_n \vee a_{2n})$ be a propositional formula and $b \in \{0,1\}^{2n}$ a truth assignment such that $a_i = b(i)$, for $1 \leq i \leq 2n$. Given F as first input and b as second input a logspace 2-1-NTM cannot check whether b satisfies F [Bur89]. Thus NLMV$\subsetneq$ NPMV.

In this paper we look at the logspace counting classes $\# \cdot$ 2-1-Σ_kL, $\# \cdot$ 2-1-Π_kL, $\# \cdot$ 1-1-Σ_kL, and $\# \cdot$ 1-1-Π_kL, for $k \in \mathbb{N}$. We show that these counting classes characterize the known logspace counting classes for $k = 0$ and $k = 1$.

Theorem 3.4 1. $\# \cdot$ *1-1-*Σ_0*L* $=$ *#1L* 3. $\# \cdot$ *2-1-*Σ_0*L* $=$ *#L*
2. $\# \cdot$ *1-1-*Σ_1*L* $=$ *span-1L* 4. $\# \cdot$ *2-1-*Σ_1*L* $=$ *span-L*

Proof. (sketch)
(1.) ($\# \cdot$ 1-1-Σ_0L $\subseteq$ #1L): Let $R \in$ 1-1-Σ_0L and let M_R be a logspace bounded 1-1-DTM that accepts R. A logspace bounded NTM M simulates M_R and guesses each bit that M_R reads from the second input tape. M does not need to store more than one guessed bit at any step of the computation because of the one-way restriction on the second input tape of M_R. Since M_R is deterministic, every $(x, y) \in R$ yields at most one accepting path of M.
(#1L $\subseteq \# \cdot$ 1-1-Σ_0L): Let M be a logspace NTM. The relation R is in 1-1-Σ_0L: $R = \{(x, y) : y$ denotes an accepting computation of M on input $x\}$.
(3.) The proof is similar to (1.) since the simulations above do not depend on the head-moves on the first tape.
(2. & 4.) $\# \cdot$ 1-1-Σ_1L $= \# \cdot$ 1NLMV $=$ span-1L and $\# \cdot$ 2-1-Σ_1L $= \# \cdot$ NLMV $=$ span-L follow directly from Lemma 3.3. $\square$

In the following we observe that the one-way logspace counting classes form a strict hierarchy within #P. That is, we get F1L$\subsetneq$ #1L$\subsetneq$ span-1L$\subsetneq$ $\# \cdot$ 1-1-Π_1L $=$ #P. In [AJ93] it is shown that 1UL $=$ 1NL $\Leftrightarrow$ #1L $=$ span-1L $\Leftrightarrow$ opt-1L $\subseteq$ #1L. For proving the following separations we use the sets $L_= = \{w@w : w \in \{a,b\}^*\}$ and $L_{\neq} = \{w@w' : w, w' \in \{a,b\}^* \wedge |w| = |w'| \wedge \exists i : w(i) \neq w'(i)\}$.

Theorem 3.5 follows from a result of Mehlhorn and Schmidt [MS82] in the field of communication complexity. In their proof they use notions and methods from linear algebra. From that it follows that #1L $\neq$ span-1L. From 1UL $\neq$ 1NL we also conclude F1L $\neq$ #1L.

Theorem 3.5 *[MS82]* $L_{\neq} \in$ *1NL* $\setminus$ *1UL.*

Remark. In order to show that $L_{\neq} \in$ 1NL we can use an algorithm which works without having the workspace marked in advance. Furthermore this proof also works if we do not use a special symbol to separate the strings w and w'.

From the following Theorem we can conclude that the one-way logspace classes form a strict hierarchy within #P.

Theorem 3.6 1. *F1L*$\subsetneq \# \cdot$ *1-1-*$\Sigma_0 L$ 2. $\# \cdot$ *1-1-*$\Sigma_0 L \subsetneq \# \cdot$ *1-1-*$\Sigma_1 L$

Proof. (sketch)
(1.) Using standard techniques it is easy to see that F1L $\subseteq \# \cdot$ 1-1-Σ_0L.
Consider the one-way logspace NTM $M_{\neq}$ that is described in the proof of Theorem 3.5 (and which accepts $L_{\neq}$). Let $f_{\neq}$ be the counting function induced by $M_{\neq}$ that is, $f_{\neq} \in \#$1L. Assume $f_{\neq} \in$ F1L, it follows that $L_{\neq} \in$ 1L and thus $L_{\neq} \in$ 1UL. This contradicts Theorem 3.5.

(2.) It is easy to see that for all sets A:

$$\chi_A \in \#1\mathrm{L} \Rightarrow A \in 1\mathrm{UL} \quad \text{and} \quad A \in 1\mathrm{NL} \Rightarrow \chi_A \in \text{span-1L}.$$

Thus, if there exists a set A such that $A \in$ 1NL $\setminus$ 1UL we get that $\chi_A \in$ span-1L $\setminus \#$1L. (In [AJ93] the opposite direction is also proven. That is, they proved $\#$1L = span-1L $\Leftrightarrow$ 1UL = 1NL.)

From [MS82] we can conclude that $\chi_{L_{\neq}} \in$ span-1L $\setminus \#$1L. □

It is shown in [HU69], that $L_= \notin$ 1NL and it is easy to see, that $L_= \in L$ thus $\chi_{L_=} \in$ FL $\setminus \# \cdot$ 1-1-Σ_1L. It follows that the one-way logspace counting classes are strict subclasses of the corresponding two-way logspace counting classes.

Observation 3.7

1. *F1L* $\subsetneq$ *FL* 2. $\# \cdot$ *1-1-*$\Sigma_0 L \subsetneq \# \cdot$ *2-1-*$\Sigma_0 L$ 3. $\# \cdot$ *1-1-*$\Sigma_1 L \subsetneq \# \cdot$ *2-1-*$\Sigma_1 L$

It is easy to see that for all $k \in$ N the classes $\# \cdot$ 2-1-Σ_kL and $\# \cdot$ 2-1-Π_kL are contained in $\#$P: A polynomially time-bounded NTM may guess the second input and simulate the 2-1-ATM deterministically. Therefore all the logspace counting classes are subclasses of $\#$P.

Observation 3.8 $\forall k \in$ N :

$$\# \cdot \textit{2-1-}\Sigma_k L \subseteq \#P \qquad \textit{and} \qquad \# \cdot \textit{2-1-}\Pi_k L \subseteq \#P$$

Theorem 3.9

$$\# \cdot \textit{1-1-}\Pi_1 L \quad = \quad \# \cdot \textit{2-1-}\Pi_1 L \quad = \quad \#P$$

We only sketch the proof here since this Theorem will follow from Theorem 5.6 as a corollary. We also refer the reader to the discussion of the results in Section 7.

Proof.(sketch)

Let $3CNF \in$ 1L be the set containing (encodings of) propositional formulas in conjunctive normal form with at most 3 literals per clause. Let $\#3SAT$ be the function, that assigns to every $F \in 3CNF$ the number of satisfying assignments. A close inspection of the reductions defined in [HIM78] and [Val79] shows, that $\#3SAT$ is complete for $\#$P via parsimonious 1L-reductions. Let $R_{SAT} = \{(F, b) : F \in CNF \wedge\ b$ is a satisfying assignment for $F\}$ and $R_{3SAT} = \{(F, b) : F \in 3CNF \wedge (F, b) \in R_{SAT}\}$. In [Bur92] it is shown that $R_{SAT} \in$ 2-1-Π_1L and using essentially the same proof it is shown in [AJ93] that $\{F@b : (F, b) \in R_{3SAT}\} \in$ co-1NL: Universally choose a clause and check whether it is satisfiable. □

Corollary 3.10 $\forall k > 1$:

$$\# \cdot \textit{1-1-}\Sigma_k L \quad = \quad \# \cdot \textit{1-1-}\Pi_k L \quad = \quad \# \cdot \textit{2-1-}\Sigma_k L \quad = \quad \# \cdot \textit{2-1-}\Pi_k L \quad = \quad \#P$$

4 Logically characterized counting classes

As described in the introduction we consider ordered structures as domain of first-order counting functions. But in order to define these functions, we use not only a linear ordering but also the successor relation or the plus relation. In this case the ordering can not only be used to avoid the counting isomorphic copies of the same solution, but also to decrease the descriptional complexity.

Definition 4.1 *Let op be one of the following predicates $<$, $SUCC$, $+$. Let σ and τ be signatures that do not contain op and $\prec$. Let $(\mathcal{A}, \prec^{\mathcal{A}})$ be an ordered σ-structure and let $op^{\prec^{\mathcal{A}}}$ be $\prec^{\mathcal{A}}$ or the successor relation or the plus relation that induces $\prec^{\mathcal{A}}$.*

For $k \in \mathbb{N}$ we define the class $\#\Sigma_k[op]$ ($\#\Pi_k[op]$) to be the class of functions f such that there exists a Σ_k-formula (Π_k-formula) ϕ over (σ, τ, op) and $f(\mathcal{A}, \prec^{\mathcal{A}}) = ||\{\mathcal{T} \; : \; (\mathcal{A}, \mathcal{T}, op^{\prec^{\mathcal{A}}}) \models \phi\}||$ for all ordered σ-structures $(\mathcal{A}, \prec^{\mathcal{A}})$.

Note, that given a linear ordering $\prec^{\mathcal{A}}$ the relations $SUCC^{\mathcal{A}}$ and $+^{\mathcal{A}}$ compatible to $\prec^{\mathcal{A}}$, are uniquely determined.

Since the counting functions do not distinguish between isomorphic ordered structures, we assume the universe to be of the form $\{0, \ldots, n-1\}$, the successor to be the $SUCC$ relation on $\mathbb{N}$ restricted to the universe, and the plus relation to be the $+$ on $\mathbb{N}$ restricted to the universe.

Our definition of first-order counting classes is a generalization of the definition given in [SST92]: $\#\Sigma_k = \#\Sigma_k[<]$ and $\#\Pi_k = \#\Pi_k[<]$ for $k \in \mathbb{N}$.

Note, that we use $<$, $SUCC$, $+$ as numerical predicates and $<^{\mathcal{A}}$, $SUCC^{\mathcal{A}}$, and $+^{\mathcal{A}}$ as numerical relations, like, for example, in [Str94]. They are *not* built-in relations, as defined for example in [Ajt83], since there may exist a function f and two isomorphic σ-structures $\mathcal{A}$ and $\mathcal{A}'$, such that $f(\mathcal{A}) \neq f(\mathcal{A}')$.

In [Lyn90] it is asked, whether $\Sigma^1_1\Pi^0_1[+]$ is strictly included in $\Sigma^1_1\Pi^0_2[+]$. It follows from Theorem 4.2 that this is not the case. To prove that $\#\Pi_1[\mathrm{SUCC}] = \#\Pi_1[+]$ we use a trick from complexity theory that is used, for example, to prove $\# \cdot \mathrm{UP} \subseteq \#\mathrm{P}$:

An UP-Turing machine accepts $(x, y) \in R$ on at most one computation path, say $p_{x,y}$. It is possible to define a new relation R' by encoding the accepting path $p_{x,y}$ and y in a single string, for all $(x, y) \in R$. The tuple $(x, \langle p_{x,y}, y\rangle)$ now, can be accepted by a deterministic Turing machine. Since $p_{x,y}$ is uniquely defined for each (x, y), the resulting counting function is the same.

In our case, we simply define for each successor relation a plus relation inducing the same linear order. Now, we can take a formula that uses a plus relation to define a counting function. Since the plus relation is uniquely defined, we get the same counting function using a successor relation instead of a plus relation. Since this is also possible in the other direction (that is defining a successor relation using a plus relation) we get $\#\Pi_1[\mathrm{SUCC}] = \#\Pi_1[+]$.

Theorem 4.2 $\#\Pi_1[SUCC] = \#\Pi_1[+]$

Proof.

($\#\Pi_1$[SUCC] $\subseteq \#\Pi_1$[+]): Let f be a counting function that is defined by a Π_1-formula ϕ using a successor relation. That is, there exists signatures σ,τ such that $f(\mathcal{A}, \prec^{\mathcal{A}}) = ||\{\mathcal{T} : (\mathcal{A}, \mathcal{T}, SUCC^{\mathcal{A}}) \models \phi\}||$ for all ordered σ-structures $(\mathcal{A}, \prec^{\mathcal{A}})$.

Due to Lemma 4.4 there exists a Π_1-formula ϕ_+ such that

$$(\mathcal{A}, 0, n-1, <^{\mathcal{A}}, +^{\mathcal{A}}, \mathcal{T}, SUCC^{\mathcal{A}}) \models (\phi \wedge \phi_+), \text{ iff } (\mathcal{A}, \mathcal{T}, SUCC^{\mathcal{A}}) \models \phi.$$

Let $\tau' = (c_0, c_{max}, <, SUCC, \tau)$ then there exist τ'-expansions $\mathcal{T}'$ for all ordered σ-structures $\mathcal{A}$ such that $f(\mathcal{A}, \prec^{\mathcal{A}}) = ||\{\mathcal{T}' : (\mathcal{A}, \mathcal{T}', +^{\mathcal{A}}) \models (\phi \wedge \phi_+)\}||$. Thus, $f \in \#\Pi_1[+]$.

($\#\Pi_1[+] \subseteq \#\Pi_1$[SUCC]): Similar as above, we use the formula ϕ_{SUCC} from Lemma 4.5 to define a successor relation using the plus relation. Therefore for all Π_1-formulas ϕ that use a plus relation to define a counting function f, there exists a Π_1-formula $(\phi \wedge \phi_{SUCC})$, that defines the same function but using a successor relation instead. □

Lemma 4.3

Let σ be a signature and $\phi_<$ be the following Π_1-formula over $(\sigma, c_0, c_{max}, <, SUCC)$:

(i) $[\forall x : \neg SUCC(x, c_0) \wedge \neg SUCC(c_{max}, x)] \wedge$

(ii) $[\forall x, y : SUCC(x, y) \rightarrow x < y] \wedge$

(iii) $[\forall x, y, z : (x < y \wedge SUCC(y, z)) \rightarrow x < z] \wedge$

(iv) $[\forall x, y : x < y \rightarrow \neg(y < x)]$

Then $(\mathcal{A}, \mathcal{T}, SUCC^{\mathcal{A}}) \models \phi_<$ iff $\mathcal{T} = (0, n-1, <^{\mathcal{A}})$, for all σ-structures $\mathcal{A}$ with $||U_{\mathcal{A}}|| = n$.

Proof.
($\Leftarrow$): It is easy to see that $(\mathcal{A}, 0, n-1, <^{\mathcal{A}}, SUCC^{\mathcal{A}}) \models \phi_<$, for all $\mathcal{A} \in STRUC[\sigma]$.

($\Rightarrow$): Let $R^{\mathcal{A}}$ be a relation, such that $(\mathcal{A}, R^{\mathcal{A}}, SUCC^{\mathcal{A}}) \models \phi_<$.
(i): The subformula (i) defines that in $R^{\mathcal{A}}$ c_0 has to be assigned with 0, and c_{max} has to be assigned with $n-1$, for $||U_{\mathcal{A}}|| = n$
(ii): This subformula defines that the successor relation is a subset of $R^{\mathcal{A}}$, in every model of $\phi_<$.
(iii): This subformula inductively defines the ordering using the successor relation.
(iv): The formulas (i) - (iii) defines $R^{\mathcal{A}}$ to be a superset of the ordering. The subformula (iv) states that $R^{\mathcal{A}}$ is not symmetric and therefore $R^{\mathcal{A}} =<^{\mathcal{A}}$. □

Lemma 4.4

Let σ be a signature and ϕ_+ be the following Π_1-formula over $(\sigma, c_0, c_{max}, <, +, SUCC)$:

(i) $\phi_< \wedge [\forall x, y : (x + c_0 = y) \leftrightarrow (x = y)] \wedge$

(ii) $[\forall x, y, z : (x + y = z) \leftrightarrow (y + x = z)] \wedge$

(iii) $[\forall x, y, z : (x + y = z \wedge c_0 < y) \rightarrow (x < z)] \wedge$

(iv) $[\forall x, y, y', z, z' : (x + y = z \wedge SUCC(y, y') \wedge SUCC(z, z')) \rightarrow (x + y' = z')]$

Then $(\mathcal{A}, \mathcal{T}, SUCC^{\mathcal{A}}) \models \phi_+$ iff $\mathcal{T} = (0, n-1, <^{\mathcal{A}}, +^{\mathcal{A}})$, for all σ-structures $\mathcal{A}$ with $||U_{\mathcal{A}}|| = n$.

Proof.(sketch)
($\Rightarrow$): Let $R^{\mathcal{A}}$ be a relation, such that $(\mathcal{A}, R^{\mathcal{A}}, SUCC^{\mathcal{A}}) \models \phi_{+}$. The formula (i) defines that $(x, 0, x) \in R^{\mathcal{A}}$, for all $x \in U^{\mathcal{A}}$ and the formula (iv) defines inductively the plus relation. Thus, the plus relation is a subset of $R^{\mathcal{A}}$. The formulas (ii) and (iii) restrict $R^{\mathcal{A}}$ to be the plus relation on $U_{\mathcal{A}}$. $\square$

Lemma 4.5

Let σ be a signature and ϕ_{SUCC} be the following Π_1-formula over $(\sigma, c_0, c_1, c_{max}, SUCC, +)$:

(i) $[\forall x : x + c_0 = x] \wedge [\forall x, y : (x + c_{max} = y) \rightarrow (x + c_0 = c_0)] \wedge$

(ii) $[c_1 \neq c_0] \wedge [\forall x, y : SUCC(x, y) \leftrightarrow x + c_1 = y] \wedge$

(iii) $\forall x, y, y', z, z'$:

$$[SUCC(x, c_{max}) \wedge (x + y = z) \wedge (x + y' = z') \wedge (x \neq z) \wedge (x \neq z')] \rightarrow [(y = y') \wedge (y = c_1)]$$

Then $(\mathcal{A}, \mathcal{T}, +^{\mathcal{A}}) \models \phi_{SUCC}$ *iff* $\mathcal{T} = (0, 1, n-1, SUCC^{\mathcal{A}})$, *for all σ-structures $\mathcal{A}$ with $||U_{\mathcal{A}}|| = n$.*

Proof.

($\Leftarrow$): It is easy to see that $(\mathcal{A}, 0, 1, n-1, SUCC^{\mathcal{A}}, +^{\mathcal{A}}) \models \phi_{<}$, for all $\mathcal{A} \in STRUC[\sigma]$.
($\Rightarrow$): The formula (i) defines c_0 to be assigned with 0 and c_{max} to be assigned with $n-1$. (To see the latter, assume that c_{max} is assigned with $n-2$, for example. Since the plus relation is given and $1 + (n-2) = n-1$ but $1 + 0 \neq 0$, the formula (i) is false for this assignment.)
(ii): This formula defines for each $1 \leq k \leq n-1$ the relation $\{(x, x+k) : x + k \leq n-1\}$. That is, it defines a successor relation, but also a +2-, +3-, $\ldots$ +$(n-1)$-relation.
(iii): The formula (iii) defines that c_1 has to be 1. It follows, that the +k-relation is not contained in any model, for $1 < k \leq n-1$. Thus $R^{\mathcal{A}}$ is the successor relation. $\square$

Theorem 4.6 $\#\Pi_2$*[<]* $= \#\Pi_1$*[SUCC]*

Before presenting the proof we informally describe the underlying ideas.

At first, it is easy to see that $\#\Pi_1$[SUCC] $\subseteq \#\Pi_2$[<]. Similar to the proof of Theorem 4.2, we use the formula ϕ_{SUCC} from Lemma 4.7 to define a successor relation using a given linear order. Thus, all counting functions that are definable via Π_1-formulas that use a successor relation are definable also by Π_2-formulas using a linear ordering.

The converse direction, that $\#\Pi_2$[<] $\subseteq \#\Pi_1$[SUCC] is much more difficult to prove. For each counting function that is definable by a Π_2-formula using a linear ordering we have to construct a Π_1-formula that defines the same counting function using a successor relation. More formally: For each Π_2-formula ϕ over $(\sigma, \tau, <)$ we construct a Π_1-formula $\widehat{\phi}$ over $(\sigma, \widehat{\tau}, SUCC)$, such that for all ordered σ-structures $(\mathcal{A}, \prec^{\mathcal{A}})$ it holds that

$$||\{\mathcal{T}^{\mathcal{A}} : (\mathcal{A}, \mathcal{T}^{\mathcal{A}}, <^{\mathcal{A}}) \models \phi\}|| = ||\{\widehat{\mathcal{T}}^{\mathcal{A}} : (\mathcal{A}, \widehat{\mathcal{T}}^{\mathcal{A}}, SUCC^{\mathcal{A}}) \models \widehat{\phi}\}||$$

Let ϕ be of the form $\forall x_1, \ldots, x_k \exists y_1, \ldots, y_l \varphi(x_1, \ldots, x_k, y_1, \ldots, y_l)$, such that φ has no quantifiers. Let $(\mathcal{A}, \mathcal{T}^{\mathcal{A}}, <^{\mathcal{A}})$ be an appropriate structure for ϕ.

The basic idea is, that iff $(\mathcal{A}, \mathcal{T}^{\mathcal{A}}, <^{\mathcal{A}})$ is a model of ϕ, then $\widehat{\phi}$ defines uniquely a relation that witnesses the fact that for all k-tuples $\vec{x}^{\mathcal{A}}$ there exists a l-tuple $\vec{y}^{\mathcal{A}}$ such that φ evaluates to true for $\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}$. This is done by encoding for all $\vec{x}^{\mathcal{A}}$ the lexicographic smallest l-tuple $\vec{y}^{\mathcal{A}}$ such that φ evaluates to true for $\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}$. We choose an encoding, that is definable by a Π_1-formula: The relation contains for all $\vec{x}^{\mathcal{A}}$ all l-tuples $\vec{z}^{\mathcal{A}}$ that are less or equal than $\vec{y}^{\mathcal{A}}$ in simultaneously all their components.

The signature $\widehat{\tau}$ consists of τ, the constant symbols c_0, c_{max}, the binary relation symbol $<$, and a $k+l$-ary relation symbol R.

Let $\mathcal{A}$ be a σ-structure and $(\mathcal{A}, \mathcal{T}^{\mathcal{A}}, c_0^{\mathcal{A}}, c_{max}^{\mathcal{A}}, R^{\mathcal{A}}, SUCC^{\mathcal{A}})$ be a model of $\widehat{\phi}$. We first present and informally describe the formula $\widehat{\phi}$.

(i) Since we want to use φ as a subformula in $\widehat{\phi}$, we have to define implicitly the $<$-relation. In order to do this the formula $\phi_<$ presented in Lemma 4.3 is a subformula of $\widehat{\phi}$. By Lemma 4.3 each model of $\widehat{\phi}$ contains the uniquely determined $<$-relation. It follows that the number of $\widehat{\tau}$-expansions is not affected by the formula $\widehat{\phi}$.

(ii) We have to ensure, that $R^{\mathcal{A}}$ is not the empty set. Therefore the following formula is a subformula of $\widehat{\phi}$.

$$\forall \vec{x} \ : \ R(\vec{x}, \vec{c_0})$$

(iii) We like to model the following property:

For all $\vec{x}^{\mathcal{A}}$ there exist a $\vec{y}^{\mathcal{A}}$ such that $(\vec{x}^{\mathcal{A}}, \vec{z}^{\mathcal{A}}) \in R^{\mathcal{A}}$ for all $\vec{z}^{\mathcal{A}}$ that are smaller than $\vec{y}^{\mathcal{A}}$ in all components (that is, $\forall 1 \leq i \leq l \ : \ z_i \leq y_i$), if $(\mathcal{A}, \mathcal{T}^{\mathcal{A}}, c_0^{\mathcal{A}}, c_{max}^{\mathcal{A}}, R^{\mathcal{A}}, SUCC^{\mathcal{A}})$ is a model of $\widehat{\phi}$.

This property of $R^{\mathcal{A}}$ is defined by the following subformula of $\widehat{\phi}$.

$$\left(\bigwedge_{1 \leq i \leq l} \forall \vec{x}, \vec{y}, y_i' \ : \ SUCC(y_i, y_i') \rightarrow [R(\vec{x}, y_1, \ldots y_{i-1}, y_i', y_{i+1}, \ldots y_l) \rightarrow R(\vec{x}, \vec{y})] \right) \ \wedge$$

$$\Bigg(\forall \vec{x}, \vec{y}, \vec{y'} \ : \ [\neg R(\vec{x}, \vec{y'}) \bigwedge_{1 \leq i \leq l} SUCC(y_i, y_i')] \rightarrow$$

$$\neg \ [\bigvee_{1 \leq i < l} \bigvee_{i < j \leq l} (R(\vec{x}, y_1', \ldots, y_{i-1}', y_i, y_{i+1}', \ldots, y_l') \wedge R(\vec{x}, y_1', \ldots, y_{j-1}', y_j, y_{j+1}', \ldots, y_l'))] \Bigg)$$

(iv) We are now ready to model the following property of $R^{\mathcal{A}}$:

Let $(\mathcal{A}, \mathcal{T}^{\mathcal{A}}, c_0^{\mathcal{A}}, c_{max}^{\mathcal{A}}, R^{\mathcal{A}}, SUCC^{\mathcal{A}})$ be a model of $\widehat{\phi}$. If, for all $\vec{x}^{\mathcal{A}}$, the l-tuple $\vec{y}^{\mathcal{A}}$ is the maximal l-tuple such that $(\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}) \in R^{\mathcal{A}}$, then $\varphi(\vec{x}, \vec{y})$ evaluates to true for $\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}$.

Unfortunately it is a bit tedious to express the maximality of $\vec{y}^{\mathcal{A}}$ with a Π_1-formula. We introduce some abbreviations to present this formula as succinct as possible. For $I \subseteq \{1, 2, \ldots, l\}$ we define $\langle \vec{y} \rangle_I$ to be the subsequence of $\vec{y}$ determined by I (more formally: $\langle \vec{y} \rangle_I = y_{i_1}, y_{i_2}, \ldots, y_{i_{|I|}}$, iff $I = \{i_1, i_2, \ldots, i_{|I|}\}$), and $\langle \vec{y}, c_{max} \rangle_I$ to be the sequence $u_1, u_2, \ldots, u_l$ such that for all $1 \leq i \leq l$ it holds that $u_i = y_i$ if $i \in I$ and $u_i = c_{max}$ otherwise. For $J \subseteq I \subseteq \{1, 2, \ldots, l\}$ we define $\langle \vec{y}, \vec{z}, c_{max} \rangle_{I,J}$ to be the sequence $u_1, u_2, \ldots, u_l$ such that $u_i = z_i$ if $i \in J$, $u_i = y_i$ if $i \in I \setminus J$, and $u_i = c_{max}$ otherwise (if $i \in \{1, 2, \ldots, l\} \setminus I$).

The formula that describes the above mentioned property of $R^{\mathcal{A}}$ consists of a conjunction of the following formula for all $I \subseteq \{1, 2, \ldots, l\}$:

$\forall \vec{x}, \langle\vec{y}\rangle_I, \langle\vec{z}\rangle_I$:

$$\left([\bigwedge_{i\in I} SUCC(y_i, z_i)] \wedge R(\vec{x}, \langle\vec{y}, c_{max}\rangle_I) \wedge [\bigwedge_{(J\subset I)\wedge(J\neq\emptyset)} \neg R(\vec{x}, \langle\vec{y}, \vec{z}, c_{max}\rangle_{I,J})] \right) \rightarrow \varphi(\vec{x}, \langle\vec{y}, c_{max}\rangle_I)$$

(v) The formula above describes a property of $R^{\mathcal{A}}$:

If $\vec{y}^{\mathcal{A}}$ is the maximal l-tuple such that $(\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}) \in R^{\mathcal{A}}$ for all $\vec{x}^{\mathcal{A}}$, then $\varphi(\vec{x}, \vec{y})$ evaluates to true for $\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}$.

In order to determine $R^{\mathcal{A}}$ uniquely, we formalize now that for all $\vec{z}^{\mathcal{A}}$ such that $\vec{z}^{\mathcal{A}}$ is 'smaller' than $\vec{y}^{\mathcal{A}}$ and $(\vec{x}^{\mathcal{A}}, \vec{z}^{\mathcal{A}}) \in R^{\mathcal{A}}$, $\varphi(\vec{x}, \vec{z})$ evaluates to false for $(\vec{x}^{\mathcal{A}}, \vec{z}^{\mathcal{A}})$.

That is, for all $\vec{x}^{\mathcal{A}}$ there exists exactly one $\vec{y}^{\mathcal{A}}$, such that $(\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}) \in R^{\mathcal{A}}$ and $\varphi(\vec{x}, \vec{y})$ evaluates to true for $\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}$.

$$\forall \vec{x}, \vec{y}, \vec{z} \, : \, [R(\vec{x}, \vec{y}) \wedge \varphi(\vec{x}, \vec{z})] \rightarrow$$

$$(\, [(y_1 < z_1)] \;\vee\; [(y_1 = z_1) \wedge (y_2 < z_2)] \;\vee\; [(y_1 = z_1) \wedge (y_2 = z_2) \wedge (y_3 < z_3)] \;\vee \ldots \vee$$
$$[(y_1 = z_1) \wedge \ldots \wedge (y_{l-1} = z_{l-1}) \wedge (y_l < z_l)] \;\vee\; [(y_1 = z_1) \wedge \ldots \wedge (y_l = z_l)] \,)$$

We combine the formulas presented in (i) - (v) as a conjunction to build the formula $\widehat{\phi}$. Since all subformulas are Π_1-formulas, it is possible to define $\widehat{\phi}$ itself as a Π_1-formula.

Proof of Theorem 4.6.

We have to prove the following:

For every τ-expansion $\mathcal{T}^{\mathcal{A}}$ of an ordered σ-structure $(\mathcal{A}, \prec^{\mathcal{A}})$ (with $n = ||U_{\mathcal{A}}||$) there exists exactly one $\widehat{\tau}$-expansion $\widehat{\mathcal{T}}^{\mathcal{A}}$ such that

$$(\mathcal{A}, \mathcal{T}^{\mathcal{A}}, <^{\mathcal{A}}) \models \phi, \quad \text{iff} \quad (\mathcal{A}, \widehat{\mathcal{T}}^{\mathcal{A}}, SUCC^{\mathcal{A}}) \models \widehat{\phi}.$$

Since $0, n-1, SUCC^{\mathcal{A}}$ are uniquely determined, it thus suffices to prove (*):

A model of ϕ contains the relation $R^{\mathcal{A}} = \{(\vec{x}^{\mathcal{A}}, \vec{z}^{\mathcal{A}}) \, : \, \vec{y}^{\mathcal{A}}$ is the lexicographically smallest l-tuple, such that $\varphi(\vec{x}, \vec{y})$ evaluates to true for $\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}$ and $z_i^{\mathcal{A}} \leq y_i^{\mathcal{A}}$, for all $1 \leq i \leq l \,\}$, iff $(\mathcal{A}, \mathcal{T}^{\mathcal{A}}, <^{\mathcal{A}}) \models \phi$. (It follows from subformulas (iii) and (v) that $R^{\mathcal{A}}$ is uniquely determined, if it exists.)

Proof of (*): $(\Rightarrow)$ If $(\mathcal{A}, \mathcal{T}^{\mathcal{A}}, <^{\mathcal{A}}) \not\models \phi$, then for all $\vec{y}^{\mathcal{A}}$ there exists a $\vec{x}^{\mathcal{A}}$ such that $\varphi(\vec{x}, \vec{y})$ evaluates to false on $\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}$. From subformula (iv) it would follow that $R^{\mathcal{A}}$ has to be the empty set, which would contradict subformula (ii). Thus there is no relation $R^{\mathcal{A}}$ in this case.

$(\Leftarrow)$ We have to show that the subformulas (ii) - (v) evaluate to true on the relation $R^{\mathcal{A}}$ as described in (*). This is easy to see for (ii), (iv), and (v).
(iii) This formula is a conjunction of two subformulas. It is easy to see that the first evaluates to true in this case.

Let z_i and z_j be two components such that $(\vec{x}^{\mathcal{A}}, \ldots, z_i - 1, \ldots, z_j, \ldots) \in R^{\mathcal{A}}$, and $(\vec{x}^{\mathcal{A}}, \ldots, z_i, \ldots, z_j - 1, \ldots) \in R^{\mathcal{A}}$. The conclusion of the second part of (iii) evaluates to false. Let $\vec{y}^{\mathcal{A}}$ be the maximal l-tuple, such that $(\vec{x}^{\mathcal{A}}, \vec{y}^{\mathcal{A}}) \in R^{\mathcal{A}}$. It follows that $z_i \leq y_i^{\mathcal{A}}$ and $z_j \leq y_j^{\mathcal{A}}$, and therefore $(\vec{x}^{\mathcal{A}}, \ldots, z_i, \ldots, z_j, \ldots) \in R^{\mathcal{A}}$. Thus, the formula (iii) evaluates to true. $\square$

Lemma 4.7

Let σ be a signature and ϕ_{SUCC} be the following Π_2-formula over $(\sigma, SUCC, <)$:

(i) $[\forall x, y : SUCC(x,y) \rightarrow x < y] \wedge [\forall x, y, z : SUCC(x,y) \rightarrow (x \geq z \vee z \geq y)] \wedge$

(ii) $[\forall x, y \, \exists z : (\neg SUCC(x,y) \wedge x < y) \rightarrow (x < z \wedge z < y)]$

Then $(\mathcal{A}, \mathcal{T}, <^{\mathcal{A}}) \models \phi_{SUCC}$ *iff* $\mathcal{T} = (SUCC^{\mathcal{A}})$*, for all σ-structures $\mathcal{A}$ with $||U_{\mathcal{A}}|| = n$.*

Proof.
($\Leftarrow$): It is easy to see that for all $\mathcal{A} \in STRUC[\sigma]$ it holds, that $(\mathcal{A}, SUCC^{\mathcal{A}}, <^{\mathcal{A}}) \models \phi_{SUCC}$.

($\Rightarrow$): Let $R^{\mathcal{A}}$ be a relation, such that $(\mathcal{A}, R^{\mathcal{A}}, <^{\mathcal{A}}) \models \phi_{SUCC}$. Because of (i) $R^{\mathcal{A}}$ has to be a subset of $SUCC^{\mathcal{A}}$. Assume to the contrary that there exists a pair $(x^{\mathcal{A}}, y^{\mathcal{A}}) \in SUCC^{\mathcal{A}} \setminus R^{\mathcal{A}}$. But for $(x^{\mathcal{A}}, y^{\mathcal{A}})$ there exists no $z^{\mathcal{A}} \in \mathcal{A}$ such that (ii) is fulfilled. $\square$

At the end of this section, we show that #P is (with respect to some encoding) a subclass of $\#\Pi_1$[SUCC]. In [SST92] it is shown that $\#\Pi_2[<]$ = #P but #P is redefined as the class of polynomial time counting functions having ordered σ-structures as domain. We keep the definition of #P unchanged and use a standard encoding of strings using monadic relations, which we denote by e_2. The elements of the universe are the input positions and it is quite obvious to use the $<$ relation over the positive integers as linear ordering.

Let $\Sigma = \{a_1, \ldots, a_n\}$ be an alphabet and k be the smallest integer, such that $n < 2^k$. We encode each string $x \in \Sigma^*$ by the finite structure $e_2(x) = (\{0, \ldots, |x|-1\}, P_1, \ldots, P_k, \prec)$. Let $\prec$ be the $<$ relation on $\mathbb{N}$ restricted on $\{0, \ldots, |x|-1\}$. The Relations $P_1, \ldots, P_k$ are all monadic. These relations form an "binary encoding" of the symbols. For example let $\Sigma = \{a, b, c\}$ and $x \in \Sigma^*$. For $1 \leq i \leq |x|$ we define $P_1(i) = 0$ and $P_2(i) = 0$ if $x(i) = a$, and $P_1(i) = 0$ and $P_2(i) = 1$ if $x(i) = b$. If $x(i) = c$ we define $P_1(i) = 1$ and $P_2(i) = 0$. (Note that if the cardinality of the input alphabet is not a power of two, then the encoding is not onto.) We define $e_2(f)$ such that for all ordered σ-structures $(\mathcal{A}, \prec^{\mathcal{A}})$, $e_2(f)(\mathcal{A}, \prec) = f(x)$ if $(\mathcal{A}, \prec) = e_2(x)$, and $e_2(f)(\mathcal{A}, \prec) = 0$ otherwise. Let $e_2(\#\mathrm{P})$ be the class of functions $e_2(f)$ for $f \in \#\mathrm{P}$.

There are several ways to prove Theorem 4.8. First, it follows from Theorem 4.6 and the result of [SST92]. But it may also be proven more directly by extending the proofs of [Fag74] and [Pap94] that existential second-order characterizes NP. Although the definitions that are used there differ slightly from ours we refer to the proof presented in [Pap94].

Theorem 4.8 $\quad e_2(\#P) \subseteq \#\Pi_1[SUCC]$

Proof.(sketch) In [Pap94] it is shown that for all polynomially time-bounded Turing machines M there exists a Π_1-formula ϕ over $(\sigma, \tau, SUCC)$ such that M accepts an input x iff there is a assignment to the second-order variables $\mathcal{T}$ such that $(e_2(x), \mathcal{T}, SUCC) \models \phi$. We have to add a subformula ϕ_{e_2} to ϕ that checks whether the finite structure is a correct encoding of a string with respect to the input alphabet of the Turing machine. A close inspection of the formula ϕ shows that for each accepting computation path of M there exists an unique assignment $\mathcal{T}$ that satisfies ϕ. $\square$

5 Comparing first-order and logtime counting classes

In [BIS90] it is shown that there are close connections between first-order definability and unbounded alternating logtime Turing machines. It is obvious to ask, whether there are similar results possible in the case of first-order definable counting functions. In this section we show that this is in fact possible. But we are faced with a technical obstacle. Contrary to the case of first-order definability of sets the hierarchy of first-order counting classes with respect to the quantifier complexity collapsesto $\#\Pi_1[op]$, for $op = SUCC$ and $op = +$, and to $\#\Pi_2[<]$. That is, we have to deal with logtime Turing machines that are deterministic, nondeterministic, and universally branching. The computational power is very low in this case. For this reason, we have to extend the usual model of random access Turing machines as used for example in [Bus87] to be able to access relations of higher arity rather than to read strings.

It is possible to obtain a characterization of #P vial universally branching logtime Turing machines using the logical characterization of NP as it is proven by [Fag74] or [Pap94].

First, we describe an encoding e_1 of ordered σ-structures into strings over the alphabet $\{0, 1\}$. Let $(\mathcal{A}, \prec^{\mathcal{A}})$ be a ordered σ-structure and n be the cardinality of the universe $U^{\mathcal{A}}$. $e_1(\mathcal{A}, \prec^{\mathcal{A}})$ starts with a binary representation of n. It must be possible that a logtime-bounded Turing machine can detect the end of $bin(n)$. Therefore, we encode $bin(n)$ by doubling each bit and adding 01 at the end. We use the enumeration induced by $\prec^{\mathcal{A}}$ to represent the constants by binary strings. As above, we encode these representations by doubling each bit and separate these encodings by 01. The relations are represented in binary strings in the standard way. Let R be, for example, a k-ary Relation, then the bit at position $x_k \cdot n^{k-1} + x_{k-1} \cdot n^{k-2} + \ldots + x_2 \cdot n + x_k$ in the encoding of R is 1, iff $(x_1, x_2, \ldots, x_k) \in R$. We encode a τ-expansion $\mathcal{T}$ of $\mathcal{A}$ using $\prec^{\mathcal{A}}$ as described above, but without the encoding of n.

For all functions $f \in \#\mathcal{FO}[op]$ let $e_1(f) : \{0, 1\}^* \to \mathrm{N}$ be the function such that for all $x \in \{0, 1\}^*$, $e_1(f)(x) = f(\mathcal{A}, \prec^{\mathcal{A}})$ if $x = e_1(\mathcal{A}, \prec^{\mathcal{A}})$, and $f(x) = 0$ otherwise. Since our encoding is one-one, the function $e_1(f)$ is well-defined. For any class of first-order counting functions $\mathcal{C} \subseteq \#\mathcal{FO}[op]$ let $e_1(\mathcal{C})$ be the class of functions $e_1(f) : \{0, 1\}^* \to \mathrm{N}$ for $f \in \mathcal{C}$.

It is easy to extend random-access Turing machines, with two index tapes to accept binary relations, similar as 2-1-TM's or 1-1-TM's. In the following, we would like to prove a result like this: Given two signatures σ and τ, a formula ϕ that defines a function f. Then there is a logtime bounded random-access Turing machine that accepts an input $(e_1(\mathcal{A}, \prec^{\mathcal{A}}), e_1(\mathcal{T}^{\mathcal{A}}))$ if and only if $(\mathcal{A}, \mathcal{T}, \prec^{\mathcal{A}}, op^{\mathcal{A}})$ is a model of ϕ. But in order to evaluate ϕ the logtime Turing machine may has to access a bit in a relation of higher arity. As it is described above, to do this the Turing machine has to perform a multiplication of $log(n)$-binary numbers. It is not clear, whether this can be done (deterministically) in logtime. We therefore extend the usual model to be able to access relations of higher arity.

To have a consistent naming we use the prefix *i-i-* which indicates that the input is accessed via an index tape. To indicate random-access Turing machines that are able to read relations of arity $k \geq 1$, we use i^k-i^k- .

Definition 5.1

For $n > 0$ a i^n-i^n-ATM M is an alternating n-dimensional random-access Turing machine with n first index tapes and n second index tapes and a base index tape. The input that is accessible via the n first input tapes, we call first input and the input accessible via the n second input tapes, we call the second input.

Let b be the content of the base index tape and let $i_{1,0}, i_{1,1}, \ldots, i_{1,n-1}$ be the contents of the n first index tapes, then M accesses the first input at position $i_{1,n-1} \cdot b^{n-1} + i_{1,n-2} \cdot b^{n-2} + \ldots i_{1,1} \cdot b + i_{1,0}$.

Let $i_{2,0}, i_{2,1}, \ldots, i_{2,n-1}$ be the contents of the n second index tapes, then M accesses the second input at position $i_{2,n-1} \cdot b^{n-1} + i_{2,n-2} \cdot b^{n-2} + \ldots i_{2,1} \cdot b + i_{2,0}$.

A i^n-i^n-ATM accepts a binary relation R of arity n, iff it accepts all $(x, y) \in R$ with x as the first input and y as the second input.

For $k > 0$ we call an i^n-i^n-ATM that makes at most $k-1$ alternations i^n-i^n-Σ_k-TM, if it starts in an existential configuration and i^n-i^n-Π_k-TM, if it starts in an universal configuration. A deterministic i^n-i^n-TM is called a i^n-i^n-Σ_0-TM or i^n-i^n-Π_0-TM.

If $n = 1$, we omit the superscript and write i-i-TM.

Definition 5.2

For every $k \in \mathbb{N}$ let i^m-i^m_{pol}-Σ_kLOGTIME (i^m-i^m_{pol}-Π_kLOGTIME) be the class of polynomially length-bounded binary relations, that are accepted by logarithmically time-bounded i^n-i^n-Σ_k-Turing machines (i^n-i^n-Π_k-Turing machines resp.) for $n > 0$.

It is easy to see that it is possible to simulate a logtime-bounded i-i-ATM by an ordinary random-access Turing machine in logtime if the (relational) input is encoded in a single string using logtime-parsable pairing functions (as described for example in [BIS90]).

We separate the above defined classes of logtime acceptable binary relations using the following sets $L_{unary}, L_{\neg\, unary} \subseteq \{0,1\}^*$:

$L_{unary} = \{ (1^n, 1) \ : \ n \in \mathbb{N} \}$

$L_{\neg\, unary} = \{ (w, 1) \ : \ \exists i : w(i) = 0 \}$

Proposition 5.3

$$\begin{array}{rcl} L_{unary} & \in & i^m\text{-}i^m_{pol}\text{-}\Pi_1 LOGTIME \setminus i^m\text{-}i^m_{pol}\text{-}\Sigma_1 LOGTIME \\ L_{\neg\, unary} & \in & i^m\text{-}i^m_{pol}\text{-}\Sigma_1 LOGTIME \setminus i^m\text{-}i^m_{pol}\text{-}\Sigma_0 LOGTIME \end{array}$$

Proof.(sketch of $L_{unary} \notin i^m$-i^m_{pol}-Σ_1LOGTIME) Assume to the contrary that there exists an accepting logtime-bounded i-i-Σ_1-TM (the arity in this case is 1). Fix an accepting path p. The number of input positions that are queried on p, is logarithmically bounded. Thus, there is at least one position that is not queried. Change the bit at this position to 0.

(sketch of $L_{\neg\, unary} \notin i^m$-$i^m_{pol}$-$\Sigma_0$LOGTIME) Fix an input $x = 1^n$. The number of queried positions is logarithmically bounded. Thus there is a position that is not queried. Change the bit at this position to 0. $\square$

Corollary 5.4

$$\# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_0 LOGTIME \subsetneq \# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_1 LOGTIME \subsetneq \# \cdot i^m\text{-}i^m_{pol}\text{-}\Pi_1 LOGTIME$$

The following theorem relates the first-order counting classes and the logtime counting classes.

Theorem 5.5 *Let op be one of the relations <, SUCC, or +. Then*

(1) $e_1(\#\Sigma_0[op]) \subseteq \# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_0 LOGTIME$

(2) $e_1(\#\Sigma_1[op]) \subseteq \# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_1 LOGTIME$

(3) $e_1(\#\Pi_1[op]) \subseteq \# \cdot i^m\text{-}i^m_{pol}\text{-}\Pi_1 LOGTIME$

Proof.

Let σ and τ be signatures and the arity of the relations is at most k. Let $e_1(f)$ be a function in $e_1(\#\Sigma_0[op])$ ($e_1(\#\Sigma_1[op])$, or $e_1(\#\Pi_1[op])$ resp.) and let ϕ be the Σ_0-formula (Σ_1-, or Π_1-formula) over (σ, τ, op) such that $f(\mathcal{A}, \prec^{\mathcal{A}}) = ||\{\mathcal{T} : (\mathcal{A}, \mathcal{T}, op^{\prec^{\mathcal{A}}}) \models \phi\}||$ for all ordered σ-structures $(\mathcal{A}, \prec^{\mathcal{A}})$.

We define the binary relation R_ϕ such that $(e_1(\mathcal{A}, \prec^{\mathcal{A}}), e_1(\mathcal{T}, \prec^{\mathcal{A}})) \in R_\phi$ iff $(\mathcal{A}, \mathcal{T}, op^{\prec^{\mathcal{A}}}) \models \phi$. Since our encoding is one-one, for all $x \in \{0,1\}^*$ we have $e_1(f)(x) = ||\{y : (x,y) \in R_\phi\}||$. Thus, it is sufficient to show that $R_\phi \in i^m\text{-}i^m_{pol}\text{-}\Sigma_0$LOGTIME ($R_\phi \in i^m\text{-}i^m_{pol}\text{-}\Sigma_1$LOGTIME, $R_\phi \in i^m\text{-}i^m_{pol}\text{-}\Sigma_1$LOGTIME, resp.).

(1) We describe a logtime-bounded i^k-i^k-DTM M_ϕ, that accepts R_ϕ. First M_ϕ reads the cardinality of the universe and then copies all the constants to its work tape. Note that ϕ is a Σ_0-formula and since it is closed, ϕ contains no variables. In order to evaluate the formula M_ϕ has to read a constant number of bits. The input positions are computable from the binary representations of the constants in logtime. To evaluate a k-ary relation, M_ϕ uses k index tapes and the base index tape, which contains the cardinality of the unvierse. M_ϕ checks deterministically the correctness of the representation: Using the cardinality, M_ϕ checks the length of the input, the length of each encoded relation, and the correctness of the encoded constants. Note, that the successor relation and the + relation can be computed by Turing machines.

(2) A Σ_1-formula ϕ contains variables that are bounded by existential quantifiers. A i^k-i^k-NTM M_ϕ evaluates such a formula similar as it is described in (1). At the beginning of the computation M_ϕ first guesses the assignments of the first-order variables. M_ϕ first guesses the assignments of the first-order variables.

(3) A universally branching i^k-i^k-ATM first guesses universally the assignments of the first-order variables and works then deterministically as described in (1). □

Using the logical characterization of #P and the above Theorem we obtain a characterization of #P via universally branching logtime Turing machines.

Theorem 5.6 $\quad \#P = \# \cdot i\text{-}i_{pol}\text{-}\Pi_1 LOGTIME$

Proof. (sketch) We show that $\#\mathrm{P} \subseteq \# \cdot i\text{-}i_{pol}\text{-}\Pi_1$LOGTIME:

Let f be a function in #P. Let ϕ be the Π_1-formula over $(\sigma, \tau, SUCC)$ such that $f(x) = ||\{\mathcal{T} : (e_2(x), \mathcal{T}, SUCC) \models \phi\}$ for all $x \in \Sigma^*$. We define a relation R such that $R = \{(x, e_1(\mathcal{T})) : (e_2(x), \mathcal{T}, SUCC) \models \phi\}$, and describe a logtime i-i-Π_1-TMM_R that accepts R. It follows that $R \in$ *i*-i_{pol}-Π_1LOGTIME and thus $f \in \# \cdot$ *i*-i_{pol}-Π_1LOGTIME.

From the proof of Theorem 5.5 it follows that there exists a i-i-Π_1-TM M_ϕ that accepts the input $(e_1(e_2(x)), e_1(\mathcal{T}))$ iff $(e_2(x), \mathcal{T}, SUCC) \models \phi$ (since all the relations in $e_2(x)$ are monadic).

At first M_R determines the length of x and from that, computes the length of $e_2(x)$. Using this length it is possible to compute $|e_1(e_2(x))|$ (the cardinality of the universe of $e_1(e_2(x))$ is $|x|$). After that M_R starts simulating the i-i-Π_1-TM M_ϕ that we have described in the proof of Theorem 5.5. The second input for M_ϕ can be directly obtained from the second input of M_R. The first input for M_ϕ (which is $e_1(e_2(x))$), can be computed by M_R using its first input x: M_R determines from an input position in $e_1(e_2(x))$ the predicate P and the element i that has to be evaluated. M_R evaluates $P(i)$ by querying x at position i. □

We can separate $e_1(\#\Sigma_1[op])$ from $e_1(\#\Pi_1[op])$ by using the idea from the above proof:

Proposition 5.7 *Let $op_1, op_2 \in \{SUCC, +\}$, then*

$$e_1(\#\Sigma_1[op_1]) \subsetneq e_1(\#\Pi_1[op_2])$$

Proof.

Assume to the contrary that $e_1(\#\Sigma_1[op_1]) = e_1(\#\Pi_1[op_2])$, it would follow that $e_1(e_2(\#\mathrm{P})) \subseteq e_1(\#\Sigma_1[op_1])$ and thus $e_1(e_2(\#\mathrm{P})) \subseteq i^m$-$i^m_{pol}$-$\Sigma_1$LOGTIME. But similar as above it would follow that $\#\mathrm{P} \subseteq \# \cdot i^m$-$i^m_{pol}$-$\Sigma_1$LOGTIME. But this yields a contradiction, since $\# \cdot i^m$-i^m_{pol}-Σ_1LOGTIME$\subsetneq \#\mathrm{P}$. □

It is also possible to separate $e_1(\#\Sigma_0[op])$ from $e_1(\#\Sigma_1[op])$:

Proposition 5.8 *Let $op_1, op_2 \in \{<, SUCC, +\}$, then*

$$e_1(\#\Sigma_0[op_1]) \subsetneq e_1(\#\Sigma_1[op_2])$$

Proof.

Consider the Σ_1 formula $\phi = \exists x : P(x) \wedge Q(x)$ over (σ, τ, op_2), for $\sigma = (P), \tau = (Q)$. Let $f \in \#\Sigma_1[op_2]$ be the function that is defined by ϕ. The function $e_1(f)$ is not in $\# \cdot i^m$-i^m_{pol}-Σ_0LOGTIME, since, for $x \in \{0,1\}^n$, $e_1(f)(bin(n)01x) > 0$, iff $x \neq 0^n$. As in the proof of Proposition 5.3 it can be shown that a logtime-bounded Turing machine can not distinguish between more that polynomially many inputs of the form $x \neq 0^n$.

In order to show that $e_1(\#\Sigma_0[op_1]) \subsetneq e_1(\#\Sigma_1[op_2])$ assume to the contrary that $e_1(\#\Sigma_0[op_1]) = e_1(\#\Sigma_1[op_2])$. The function $e_1(f)$ that is described above is in $e_1(\#\Sigma_1[op_2])$ and therefore it would follow that $e_1(f) \in e_1(\#\Sigma_0[op_1])$. But $e_1(\#\Sigma_0[op_1]) \subsetneq \# \cdot i^m$-$i^m_{pol}$-$\Sigma_0$LOGTIME and thus it would follow that $e_1(f) \in \# \cdot i^m$-i^m_{pol}-Σ_0LOGTIME. But this can not be the case. □

6 Comparing logtime and one-way logspace counting classes

It is obvious to ask whether the logtime counting-classes are included in the corresponding one-way logspace counting classes. These inclusions do not follow from the definitions, since random access Turing machines read their inputs not from left to right, in general. In order to overcome this technical obstacle we use in the following proofs the notion of a potential input.

Let M be a random access TM (with one index tape). A computation of M on input x induces a sequence of input positions $p_1, p_2, \ldots, p_n$ such that p_i is the content of M's index tape in step i. The string $x(p_1)x(p_2)\ldots x(p_n)$ is logarithmically length-bounded, if M is logarithmically time-bounded. Conversely we can associate to a string $\overline{x}$ of length n, a sequence of input positions $q_1, q_2, \ldots, q_n$ as follows: let $q_1 = 1$ and let q_i be the content of M's index tape after reading $\overline{x}(1)\overline{x}(2)\ldots\overline{x}(i-1)$, for all $1 < i \leq n$. We will call such a string a *potential input*, iff the computation stops after exactly n steps and for all $1 \leq i, j \leq n$ $q_i = q_j$ implies that $\overline{x}(i) = \overline{x}(j)$. We say that a potential input $\overline{x}$ is *consistent* with an input x, iff $\overline{x}(i) = x(q_i)$, for all $1 \leq i \leq |\overline{x}|$ and input positions q_i that are associated to $\overline{x}$. (Note, that we use a special symbol $\perp$ for $\overline{x}(i)$, iff $q_i > n$.)

This notion easily extends to random access Turing machines with more that one index tape:
(-) If there are two inputs x and y, then we use two strings $\overline{x}$ and $\overline{y}$ s. t. $\overline{x}$ is a potential input and we check, whether $\overline{x}$ is consistent with x, and whether $\overline{y}$ is consistent with y.
(-) For multi-dimensional input, only the computation of the input position changes. That is, the positions p_i are determined using the contents of all the index tapes as stated in Definition 5.1 rather than being the content of an index tape. But we can still associate to each bit of a logarithmically length-bounded string an input position as described above.

Theorem 6.1

$$\# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_0 LOGTIME \subseteq \# \cdot \textit{1-1-}\Sigma_0 L$$

Proof.

Let f be a function in $\# \cdot i^m$-i^m_{pol}-Σ_0LOGTIME and $R \in i^m$-i^m_{pol}-Σ_0LOGTIME such that for all strings x it holds that $f(x) = ||\{y : (x,y) \in R\}||$. Let R be of arity k and M_R be the deterministic logtime i^k-i^k-Σ_0-TM that accepts R within the time bound $t \in O(log)$. Consider the following relation R': $(x, \langle \overline{x}, \overline{y}, y\rangle) \in R' \quad \Leftrightarrow \quad (x,y) \in R$ and $\overline{x}, \overline{y}$ are potential inputs of M_R that are consistent with x and y, resp. Since M_R is deterministic, there exists exactly one potential input $\overline{x}, \overline{y}$ that is consistent with x, y, for all inputs x, y. It follows that $f(x) = ||\{y : (x,y) \in R\}|| = ||\{\langle \overline{x}, \overline{y}, y\rangle : (x, \langle \overline{x}, \overline{y}, y\rangle) \in R'\}||$. Thus, it is sufficient to show that $R' \in$ 1-1-Σ_0L.

Let $M_{R'}$ be the deterministic 1-1-Σ_0-TM that accepts R' as follows: First $M_{R'}$ copies $\overline{x}, \overline{y}$ on its worktapes and checks parallel to that, whether $|\overline{x}| = |\overline{y}| \leq t(|x|)$. After that, $M_{R'}$ simulates M_R and checks, that M_R stops after exactly $|\overline{x}|$ steps. If this is true, $M_{R'}$ simulates M_R $|\overline{x}|$-times, to check whether $q_i = q_j$ implies that $\overline{x}(i) = \overline{x}(j)$, for all $1 \leq i, j \leq |\overline{x}|$. $M_{R'}$ uses $|\overline{x}|^2$ simulations of M_R to check the consistency.

$M_{R'}$ accepts, iff M_R accepts. □

For nondeterministic and universally branching 1-1-Turing machines, we get a stronger result. Since it is possible to guess the potential inputs, we do not need to change the relation.

Theorem 6.2

(1) i^m-i^m_{pol}-$\Sigma_1 LOGTIME \subseteq$ *1-1*-$\Sigma_1 L$

(2) i^m-i^m_{pol}-$\Pi_1 LOGTIME \subseteq$ *1-1*-$\Pi_1 L$

Proof.

(1) Let $R \in i^m$-i^m_{pol}-Σ_1LOGTIME be a relation of arity k and M_R be the nondeterministic logtime i^k-i^k-TM that accepts R within the time bound $t \in O(log)$. A logspace 1-1-NTM $M_{R'}$ accepts R as follows:
On input x, y $M_{R'}$ first guesses a protocol p of length less than $t(|x|)$ and a two strings $\overline{x}, \overline{y}$, s. t. $|p| = |\overline{x}| = |\overline{y}|$. For the rest of the computation $M_{R'}$ uses p to simulate M_R deterministically. $M_{R'}$ checks whether $\overline{x}$ and $\overline{y}$ are potential inputs of M_R that are consistent with x and y using the algorithm that is described in the proof of Theorem 6.1. $M_{R'}$ accepts, iff M_R accepts.

(2) A universally branching logspace 1-1-Π_1-TM M_R accepts a relation $R \in$ 1-1-Π_1L using repeated simulations of an i^k-i^k-Π_1-TM M_R similar as above, guessing the protocol p and $\overline{x}, \overline{y}$ universally. If $\overline{x}$ and $\overline{y}$ are not potential inputs of M_R that are consistent with x and y, then $M_{R'}$ accepts. Otherwise $M_{R'}$ accepts, iff M_R accepts. (Note, that there exists a consistent potential input for each protocol p. Thus, if M_R has a non-accepting computation, then there exists a non-accepting computation of $M_{R'}$ too.) □

By the definition of the counting operator we get the results for the counting classes as a corollary.

Corollary 6.3

(1) $\# \cdot i^m$-i^m_{pol}-$\Sigma_1 LOGTIME \subseteq \# \cdot$ *1-1*-$\Sigma_1 L$

(2) $\# \cdot i^m$-i^m_{pol}-$\Pi_1 LOGTIME \subseteq \# \cdot$ *1-1*-$\Pi_1 L$

We get Theorem 3.9 that $\# \cdot$ 1-1-Π_1L $= \# \cdot$ 1-1-Π_1L $= \#$P using Theorem 5.6 and Theorem 6.3.(2) as a corollary.

Corollary 6.4

$$\#P \subseteq \# \cdot i^m\text{-}i^m_{pol}\text{-}\Pi_1 LOGTIME \subseteq \# \cdot \textit{1-1-}\Pi_1 L \subseteq \# \cdot \textit{2-1-}\Pi_1 L \subseteq \#P$$

Deterministic and nondeterministic logtime Turing machines read only logarithmically many input bits in accepting computations. Using this observation, it is easy to separate the logtime and the corresponding one-way logspace counting classes:

Theorem 6.5

(1) $\quad \# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_0 LOGTIME \subsetneq \# \cdot 1\text{-}1\text{-}\Sigma_0 L$

(2) $\quad \# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_1 LOGTIME \subsetneq \# \cdot 1\text{-}1\text{-}\Sigma_1 L$

Proof.

Consider the set $A = \{w \; : \; w \in \{0,1\}^* \quad \wedge \quad |w|_1 > \frac{|w|}{2}\}$ that is, A contains all strings $w \in \{0,1\}^*$ that have more "ones" that "zeroes". It is easy to see that $\chi_A \in$ F1L $\setminus \# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_1$LOGTIME. □

It is obvious to ask, whether it is possible to get a result similar to Theorem 6.2 also in the case of deterministic computations (that is, to prove $i^m\text{-}i^m_{pol}\text{-}\Sigma_0$LOGTIME $\subseteq$ 1-1-Σ_0L). But this is not possible.

Proposition 6.6

$$i^m\text{-}i^m_{pol}\text{-}\Sigma_0 LOGTIME \not\subseteq 1\text{-}1\text{-}\Sigma_0 L$$

Proof.

Let $last(w)$ be the suffix of w of length $\lfloor \sqrt{log(|w|)} \rfloor$, for all $w \in \Sigma^*$.Let $bin^{-1}(v)$ be the number, for which $v = bin(|v|, bin^{-1}(v))$ (each $v \in \Sigma^*$ may be seen as the binary representation of some natural number, which we denote by $bin^{-1}(v)$).

Consider the following relation $R \subseteq \Sigma^* \times \Sigma$, for $\Sigma = \{0,1\}$ defined as follows: $(w,b) \in R \Leftrightarrow b = w(bin^{-1}(last(w)))$.

It is easy to see that $R \in i^m\text{-}i^m_{pol}\text{-}\Sigma_0$LOGTIME: A logtime i-i-DTM copies the last $\lfloor \sqrt{log(|w|)} \rfloor$ of its first input to the first index tape using $log(|w|)$ time and checks afterwards, whether the accessed bit on the first input and the bit of the second input are the same.

Suppose that there exists a logspace 1-1-DTM M_R that accepts R. $2^{\lfloor \sqrt{log(|w|)} \rfloor}$ can not be logarithmically bounded in the length of w. For each v, the number of different strings u such that $|u| = \lfloor \sqrt{log(|uv|)} \rfloor$can not be polynomially bounded in $|uv|$. Thus, there are two strings $u \neq u'$ ($|u| = |u'| = \lfloor \sqrt{log(|uv|)} \rfloor$) such that M_R gets in the same configuration for both inputs. Let v a suffix that encodes a position i, such that $u(i) \neq u'(i)$, then M_R will accept both uv and $u'v$ which contradicts the assumption. □

We thus have the interesting situation, that $i^m\text{-}i^m_{pol}\text{-}\Sigma_0$LOGTIME $\not\subseteq$ 1-1-Σ_0L but $\# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_0$LOGTIME $\subseteq \# \cdot$ 1-1-Σ_0L.

7 Summary

Comparing logically counting classes and Turing machine based counting classes, we get the following results:

(1.) The interesting result in [SST92] that the hierarchy of first-order counting classes collapse to the Π_2-Level depends on the chosen ordering. Using a successor relation (or a plus relation) we get a collapse to the Π_1-Level.

Although it is interesting to distinguish #P-complete counting functions with respect to their descritpional complexity, it seems to be more reasonable to use the successor relation when dealing with Turing machine-based counting classes.

(2.) Using the above result we get a characterization of #P via universally branching logtime Turing machines, and thus

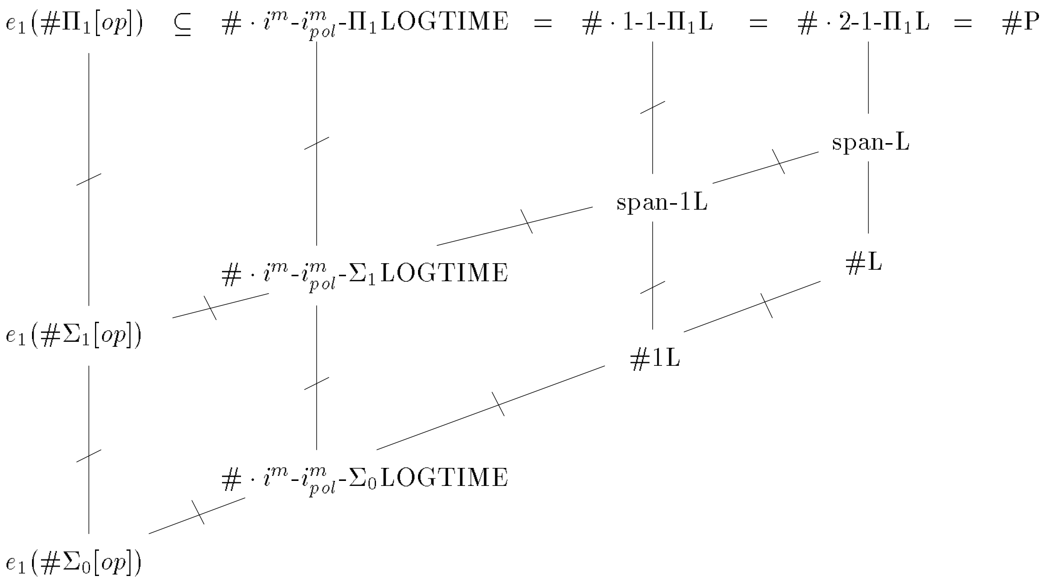
$$\#\mathrm{P} \quad = \quad \# \cdot i\text{-}i_{pol}\text{-}\Pi_1\mathrm{LOGTIME} \quad = \quad \# \cdot 1\text{-}1\text{-}\Pi_1\mathrm{L} \quad = \quad \# \cdot 2\text{-}1\text{-}\Pi_1\mathrm{L}$$

It was first proven in Georgenthal [Bur92] that $\#\mathrm{P} = \# \cdot 2\text{-}1\text{-}\Pi_1\mathrm{L}$. A close inspection of the proof yields also $\#\mathrm{P} = \# \cdot 1\text{-}1\text{-}\Pi_1\mathrm{L}$. A related result may be found also in [AJ93].

(3.) There is a strict hierarchy of one-way logspace counting classes within #P:

$$\#1\mathrm{L} \quad \subsetneq \quad \text{span-1L} \quad \subsetneq \quad \#\mathrm{P}$$

(4.) We get the following inclusions (for $op = SUCC$ or $op = +$):



(5.) The classes of (encoded) first-order definable counting functions are strictly contained in the corresponding logtime counting functions, since the encoding is not onto. But concerning one-way logspace counting classes it is possible to separate the classes of first-order counting functions from the one-way logspace counting classes even within the range of the encoding. That is,

$$\#1\mathrm{L} \not\subseteq \#\Sigma_1[<] \quad \text{and} \quad \text{span-1L} \not\subseteq \#\Pi_1[<]$$

.

It follows from [SST92] that $e_1(\#DEG\_1\_NGB) \notin e_1(\#\Pi_1[<])$ and $e_1(\#DIST_2) \notin e_1(\#\Sigma_0[<])$. It is easy to construct the one-way logspace Turing machines such that $e_1(\#DEG\_1\_NGB) \in$ span-1L and $e_1(\#DIST_2) \in \#1\mathrm{L}$.

(6.) For proving theorems concerning counting classes it is an often-used techique to code some information (that is uniquely determined) into the strings, that are counted. (To prove that

$\# \cdot \mathrm{UP} = \#\mathrm{P}$, for example.) One may ask, whether this is always necessary or if it is possible to prove the thoerem without this method. For proving that $\# \cdot i^m\text{-}i^m_{pol}\text{-}\Sigma_0\mathrm{LOGTIME} \subseteq \# \cdot 1\text{-}1\text{-}\Sigma_0\mathrm{L}$, we need this technique, since $i^m\text{-}i^m_{pol}\text{-}\Sigma_0\mathrm{LOGTIME} \not\subseteq 1\text{-}1\text{-}\Sigma_0\mathrm{L}$!

Acknowledgments. I would like to thank Martin Dietzfelbinger, Arfst Nickelsen, Dirk Siefkes, Heribert Vollmer, Klaus Wagner, for helpful comments and fruitful discussion.

References

[AJ93] C. Àlvarez and B. Jenner. A Very Hard Log Space Counting Class. *Theoretical Computer Science*, 107:3–30, 1993.

[Ajt83] M. Ajtai. Σ^1_1 formulae an finite structures. *Ann. of Pure and Applied Logic*, 24:1–48, 1983.

[BDG89] J. L. Balcázar, J. Diaz, and J. Gabarró. *Structural complexity I.* EATCS Monographs on Theoretical Computer Science. Springer, 1989.

[BIS90] David A. Mix Barrington, Neil Immerman, and H. Straubing. On Uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[Bur89] H.-J. Burtschick. Vergleich von logarithmischer Platzbschränkung und polynomieller Zeitbeschränkung für Konstruktionsprobleme und Belegmengen. Diplomarbeit, 1989.

[Bur92] H.-J. Burtschick. Dokumentation der ergebnisse des arbeitstreffens komplexitätstheorie in georgenthal/Thür. vom 24.2.1991-1.3.1991. Unpublished Notes, September 1992.

[Bus87] S. R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proc. 19th. Ann. ACM Symp. on Theory of Computing*, pages 123 – 131, 1987.

[CG94] Kevin J. Compton and Erich Grädel. Logical Definability of Counting Functions. In *9th Ann. Conf. Structure in Complexity Theory*, pages 255–266, 1994.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial time recognizable sets. In R. Karp, editor, *Complexity of Computations*, pages 43–73, 1974.

[HIM78] J. Hartmanis, N. Immerman, and S. Mahaney. One-way log-tape reductions. In *Proc. 19th IEEE Symp. Foundations of Computer Science*, pages 65 – 71. IEEE Computer Society Press, 1978.

[HU69] J. E. Hopcroft and J. D. Ullman. Some Results on Tape-Bounded Turing Machines. *Journal of the ACM*, 16(1):168–177, 1969.

[HV95] L. Hemaspaandra and H. Vollmer. The satanic notations: counting classes beyond #P and other definitional adventures. *Complexity Theory Column 8, ACM SIGACT-Newsletter*, 26(1):2–13, 1995.

[Lan86] Klaus Jörn Lange. Two characterizations of the logarithmic alternation hierarchy. In *Proc. of the 12th Symp. on Mathematical Foundations of Computer Science*, volume 233 of *LNCS*, pages 518–526. Springer Verlag, August 1986.

[Lyn90] J. F. Lynch. The Quantifier Structure of Sentences that Characterize Nondeterministic Time Complexity. In *5th Ann. Conf. Structure in Complexity Theory*, pages 210–222, 1990.

[MS82] K. Mehlhorn and E. M. Schmidt. Las vegas is better than determinism in vlsi and distributed computing. In *Proc. 14th. Ann. ACM Symp. on Theory of Computing*, pages 330–337, 1982.

[Pap94] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[SST92] S. Saluja, K. V. Subrahmanyam, and M. N. Thakur. Descriptive complexity of #P functions. In *7th Ann. Conf. Structure in Complexity Theory*, 1992.

[Str94] H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, 1994.

[Val79] Leslie G. Valiant. The complexity of enumeration and reliabilty problems. *SIAM Journal of Computing*, 8(3):411–421, August 1979.

[Vol94a] H. Vollmer. *Komplexitätsklassen von Funktionen.* PhD thesis, Universität Würzburg, Institut für Informatik, Germany, 1994.

[Vol94b] H. Vollmer. On different reducibility notions for function classes. *Proccedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS(775):449–460, February 1994.