Recursion Theoretic Characterizations of Complexity Classes of Counting Functions

Heribert Vollmer[*] *and* *Klaus W. Wagner*[†]

Theoretische Informatik
Universität Würzburg
Am Exerzierplatz 3
D-97072 Würzburg, Germany

Abstract

There has been a great effort in giving machine independent, algebraic characterizations of complexity classes, especially of functions. Astonishingly, no satisfactory characterization of the prominent class # P has been known up to now. Here, we characterize # P as the closure of a set of simple arithmetical functions under summation and weak product.

Building on that result, the hierarchy of counting functions, which is the closure of # P under substitution, is characterized; remarkably without using the operator of substitution, since we can show that in the context of this hierarchy the operation of modified subtraction is as powerful as substitution.

This leads us to a number of consequences concerning closure of # P under certain arithmetical operations.

Analogous results are achieved for the class Gap-P which is the closure of # P under subtraction.

1. Introduction

Ever in the history of computing, there has been a great effort to give machine independent characterizations of problem classes that first had been defined by putting restrictions on resources of some computation models. The two best known ways of doing so are the characterization of complexity classes with logical means (see for example the seminal work of Ronald Fagin, Neil Immerman and many others [7, 13, 14]) and the algebraic approach to complexity theory.

A very well known example from this second area is the characterization of all computable functions as the closure of a small set of simple functions under operations like substitution and some kinds of recursion, for example primitive recursion and minimization. In a seminal paper [5] Alan Cobham was the first who isolated polynomial time as a complexity class (see the historic overview by Michael Sipser [19]), and characterized the polynomial time computable functions using a restricted form of recursion, called bounded recursion on notation. Using this and other restrictive kinds of recursion, characterizations of many complexity classes within PSPACE have been given, for an overview see [26, §10] and [4].

[*]This work was supported by an Alexander von Humboldt fellowship, while the author held a visiting position at the University of California at Santa Barbara.

[†]Supported by DFG grant no. Wa 847/3-1.

Using substitution, bounded primitive recursion and the operators of summation and product (which also can be considered as very restricted forms of recursion), the class of functions computable in polynomial time (FP), the hierarchy of counting functions (FCH, a hierarchy based on Valiant's class $\#\mathrm{P}$), and the class of functions computable in polynomial space (FPSPACE) have been characterized in very similar ways, where the only difference lies in the question whether the process of summation or product is allowed to range over exponentially many values or is restricted to a polynomial number [25].

However, in these characterizations (and to our knowledge in all other similar results given up to now), the obtained class of functions is closed under substitution (simply since substitution is one of the defining operators). Therefore, it was of course not possible to characterize the classes $\#\mathrm{P}$ [22] and Gap-P [8], since these are most likely not closed under substitution. Thus, no similar characterization of $\#\mathrm{P}$ was known up to now.

Using the method of arithmetization of boolean formulae well known from Shamir's famous result [18, 1], we show that the functions from $\#\mathrm{P}$ are exactly those functions which can be obtained from a set of simple arithmetic base functions under the operators summation (of exponentially many values) and weak product (weak means: only of polynomially many values).

Building on that, and showing that substitution in the context of the examined classes can surprisingly be simulated by summation and modified subtraction, we then obtain a characterization of the hierarchy of counting functions, which is the closure of $\#\mathrm{P}$ under substitution, but remarkably, in that characterization the operator of substitution does not appear. It was already known since [16] that modified subtraction (defined as $a \dot{-} b =_{\mathrm{def}} \max\{0, a-b\}$) is a very powerful operation in the context of counting functions; but here we show that this operation is indeed as powerful as substitution. Thus, we get that the hierarchy of counting functions, which is the closure of $\#\mathrm{P}$ under substitution, is exactly the closure of the set of arithmetic base functions mentioned above under summation, weak product, and modified subtraction. This can even be simplified and we get the astonishing result that the hierarchy of counting functions is equal to the closure of all polynomials with nonnegative integer coefficients under summation and modified subtraction.

As an immediate consequence of the just given characterization, we see that $\#\mathrm{P}$ is closed under modified subtraction if and only if the hierarchy of counting functions collapses to $\#\mathrm{P}$, which in turn is equivalent to a collapse of the counting hierarchy to UP [22]. An analogous consequence concerning division of $\#\mathrm{P}$ functions is given. To prove such a result for an operation, say $\circ$, it is because of the above described characterization only necessary to show how modified subtraction can be simulated using the operation $\circ$ and other operations which appear in the recursion theoretic characterization of $\#\mathrm{P}$.

Similar results are given for the class Gap-P (the closure of $\#\mathrm{P}$ under subtraction [8]), and the here introduced hierarchy of gap functions.

It should be remarked that it is common folklore that the $\#$ operator applied to the class AC^0 (or, in logical terms, FO [14]) yields the class $\#\mathrm{P}$. This shows that $\#\mathrm{P}$ is the closure of AC^0 under summation. It is even known that the subclass $\#\Pi_2$ already contains $\#\mathrm{P}$ [17]. However, this result does not imply our characterization of $\#\mathrm{P}$ (or Gap-P). A direct arithmetization of the class $\#\Pi_2$ yields a more complicated operator structure than simply first applying weak product and then applying summation (see our Theorem 3.1). Since the result from [17] directly uses Fagin's logical characterization of NP [7], there is no hope to do with a simpler quantifier structure. To obtain our result, we arithmetize the behaviour of Turing machines such that we can take advantage of the properties of the $\#$ operator. This cannot be achieved using Fagin's result in a black-box manner.

2. Preliminaries

We assume the reader to be familiar with standard complexity theory notions, see e.g. [2, 12].

The class FP_+ (FP, resp.) is the class of all nonnegative integer functions (integer functions, resp.), computable in polynomial time by a deterministic Turing machine.

The class $\#\mathrm{P}$, introduced by Valiant [23], consists of those functions f for which there exists a nondeterministic polynomial time Turing machine M, such that for all x, $f(x)$ is equal to the number of accepting paths of the computation of M on input x.

The class Gap-P, introduced by Fenner, Fortnow, and Kurtz [8], consists of those functions f for which there exists a nondeterministic polynomial time Turing machine M, such that for all x, $f(x)$ is equal to the number of accepting paths of the computation of M on input x minus the number of rejecting paths of M on input x. It follows more or less directly from these definitions, that Gap-P is the closure of $\#\mathrm{P}$ under subtraction.

The hierarchy of counting functions was defined by Wagner [25] to consist of the classes $0\#\mathrm{P} =_{\mathrm{def}} \mathrm{FP}_+$ and $i\#\mathrm{P} =_{\mathrm{def}} \#\mathrm{P}^{(i-1)\#\mathrm{P}}$ for $i \geq 1$, where for a relativizable function class $\mathcal{F}$ and another function class $\mathcal{G}$, $\mathcal{F}^{\mathcal{G}}$ denotes the class of all functions which can be computed with $\mathcal{F}$ resources but allowing oracle access to functions from $\mathcal{G}$. (We adopt the convention that a Turing machine with oracle function g is equipped with an oracle tape, on which it writes the intended oracle query x. When this is done, it changes into a special query state and then receives $g(x)$ as the oracle answer in one time step. The oracle answer replaces the query on the oracle tape.) Let $\mathrm{FCH} =_{\mathrm{def}} \bigcup_{i\geq 0} i\#\mathrm{P}$.

The counting hierarchy (of sets) [25] is the hierarchy $\mathrm{CH} =_{\mathrm{def}} \mathrm{PP} \cup \mathrm{PP}^{\mathrm{PP}} \cup \mathrm{PP}^{\mathrm{PP}^{\mathrm{PP}}} \cup \ldots$, where PP denotes Gill's class of probabilistically polynomial time decidable sets [9].

The class UP [22] consists of all sets whose characteristic function is in $\#\mathrm{P}$. The class SPP [8] (also known under the name XP [16]) consists of all sets whose characteristic function is in Gap-P. It is known that $\text{Gap-P}^{\mathrm{SPP}} = \text{Gap-P}$, which makes SPP low for all so called gap definable classes; for an exact statement see [8].

We use the following operators on classes of functions. Let $\mathcal{F}$ be any class of functions. We say that $h \in \mathrm{Sum}\,\mathcal{F}$ ($h \in \mathrm{WSum}\,\mathcal{F}$, $h \in \mathrm{Prod}\,\mathcal{F}$, $h \in \mathrm{WProd}\,\mathcal{F}$, resp.) if there exist $f \in \mathcal{F}$ and a polynomial p such that $h(x) = \sum_{y=0}^{2^{p(|x|)}} f(x,y)$ ($h(x) = \sum_{y=0}^{p(|x|)} f(x,y)$, $h(x) = \prod_{y=0}^{2^{p(|x|)}} f(x,y)$, $h(x) = \prod_{y=0}^{p(|x|)} f(x,y)$, resp.). We also consider the simple arithmetic operations addition, multiplication, subtraction, integer division (denoted by ":"), and exponentiation as operators on functions. Additionally, we consider modified subtraction: For integers x, y, let $x \dot{-} y =_{\mathrm{def}} \max\{0, x-y\}$. For any arithmetical operation $\circ$ and any function classes $\mathcal{F}_1$ and $\mathcal{F}_2$, let $\mathcal{F}_1 \circ \mathcal{F}_2 =_{\mathrm{def}} \{\, f_1 \circ f_2 \mid f_1 \in \mathcal{F}_1, f_2 \in \mathcal{F}_2 \,\}$. The operation of substitution (i. e. composition) is denoted by Sub. Later in Section 5, we will consider any number theoretic function f as a functional operator and thus, it should lead to no confusion if we say that some class of functions $\mathcal{F}$ is closed under function f.

If $\mathcal{O}_1, \ldots, \mathcal{O}_k$ are operators as above, and $\mathcal{F}$ is a class of functions, then $[\mathcal{F}]_{\mathcal{O}_1,\ldots,\mathcal{O}_k}$ denotes the algebraic closure of $\mathcal{F}$ under the operations $\mathcal{O}_1, \ldots, \mathcal{O}_k$ and the operations of identification of variables, restriction (i. e. substitution of constants), and introduction of variables (i. e. composition with one of the identity functions i_k^n defined by $i_k^n(x_1, \ldots, x_n) =_{\mathrm{def}} x_k$). These latter operations are included for technical reasons to obtain "smoother" classes.

3. Characterizations of $\#\mathrm{P}$ and Gap-P

Obvious recursion theoretic characterizations of $\#\mathrm{P}$ and Gap-P are of course the following: Take as base functions the class FP_+ (FP, resp.) and take as recursion operator the summation operator, i.e.

$$\#\mathrm{P} = [\mathrm{FP}_+]_{\mathrm{Sum}} \quad \text{and} \quad \text{Gap-P} = [\mathrm{FP}]_{\mathrm{Sum}}.$$

Of course, this characterization is not satisfactory since the class of base functions is much too powerful and unstructured. Inspired by Immerman's BIT predicate [13], we define the following function:

$$\mathrm{bit}(\xi, i) =_{\mathrm{def}} \text{the } i\text{-th bit in the binary representation of } \xi \text{ (possibly with leading zeros)};$$

that is, if the binary representation of ξ is $b_\ell, b_{\ell-1}\cdots b_0$, then $\mathrm{bit}(\xi, i) = b_i$ if $i \leq \ell$, and $\mathrm{bit}(\xi, i) = 0$ if $i > \ell$. Additionally, we will have occasion to use the function lg defined by $\lg(x) =_{\mathrm{def}} |x|$, i. e. the length of x in binary.

3.1 Theorem.

1. $$\begin{aligned}\#\mathrm{P} &= \mathrm{Sum}(\mathrm{WProd}([+, \dot{-}, \cdot, \lg, \mathrm{bit}]_{\mathrm{Sub}})) \\ &= \Big[[+, \dot{-}, \cdot, \lg, \mathrm{bit}]_{\mathrm{Sub}}\Big]_{\mathrm{WProd,Sum}}\end{aligned}$$
2. $$\begin{aligned}\text{Gap-P} &= \mathrm{Sum}(\mathrm{WProd}([+, \dot{-}, \cdot, \lg, \mathrm{bit}]_{\mathrm{Sub}} - [+, \dot{-}, \cdot, \lg, \mathrm{bit}]_{\mathrm{Sub}})) \\ &= \Big[[+, -, \dot{-}, \cdot, \lg, \mathrm{bit}]_{\mathrm{Sub}}\Big]_{\mathrm{WProd,Sum}}\end{aligned}$$

Proof. We start by giving a general outline of how to simulate Turing machines using arithmetical functions, and then proceed with different modifications leading to the different equalities of the Theorem.

Let $f \in \#\mathrm{P}$ be witnessed by the nondeterministic Turing machine M. Let r be the polynomial that bounds the running time of M. Let M operate over the alphabet $A = \{b_0, b_1, \ldots, b_\ell\}$. Suppose without loss of generality that $b_0 = 0$, $b_1 = 1$, and $b_2 = \Box$ (the blank symbol). Let $Q = \{q_0, q_1, q_2 \ldots, q_m\}$ be the set of states of M, where q_0 is the initial state, q_1 is the only accepting state, and q_2 is the only rejecting state. Suppose further that M never moves its head to a tape square to the left of the head position in the initial configuration, and that M accepts and rejects with an empty tape and head position as in the initial configuration.

Then obviously, every accepting computation of M on input $x = a_1 a_2 \cdots a_n$ can be described by a sequence

$$\bar{z} =_{\mathrm{def}} c_{0,0}c_{0,1}c_{0,2}\cdots c_{0,r}c_{1,0}c_{1,1}\cdots c_{r,r},$$

where r abbreviates $r(n)$, every c_{ij} $(0 \leq i, j \leq r)$ is a symbol from $A \cup A \times Q$, and the following equations hold:

1. $c_{i,0} = c_{i,r} = \Box$ for $0 \leq i \leq r$.
2. $c_{0,1} = (a_1, q_0)$, $c_{0,j} = a_j$ for $2 \leq j \leq n$, and $c_{0,j} = \Box$ for $n < j \leq r$.
3. $c_{r,1} = (\Box, q_1)$, and $c_{r,j} = \Box$ for $2 \leq j \leq r$.
4. $\delta_M(c_{i-1,j-1}, c_{i-1,j}, c_{i-1,j+1}, c_{i,j-1}, c_{i,j}, c_{i,j-1})$ for $1 \leq i \leq r$ and $1 \leq j \leq r-1$. Here δ_M describes, according to the transition function of M, how in the configuration at step i the symbol on tape square j depends on the position of the tape head, the state of the machine, and the contents of the tapes $j-1$, j, and $j+1$ of the configuration at step $i-1$.

Item 1. requires that the first and the last symbol of the part of the tape that we consider is always the blank symbol, 2. describes the initial configuration, 3. describes the accepting configuration, and 4. ensures that the sequence of configurations encoded by $\bar{z}$ really corresponds to the transition function of M.

Let $s =_{\text{def}} (\ell+1)\cdot(m+2)$. From now on, we think of every string $\bar{z}$ as above as the s-ary representation of some natural number z. For every $k \geq 2$, we define the following function:

$\text{bit}_k(\xi, i) =_{\text{def}}$ the i-th symbol in the k-ary representation of ξ (possibly with leading zeros).

Encoding b_i by i (for $i = 1,\dots,\ell$), and (b_i, q_j) by $i + (\ell+1)(j+1)$ (for $i = 0,\dots,\ell$ and $j = 0,\dots,m$), and abbreviating $\text{bit}_s(z, i(r+1)+j)$ by $z_{i,j}$ (for $0 \leq i,j \leq r$) and $\text{bit}(x, i-1)$ by x_i (for $1 \leq i \leq n$), we have

$$\begin{aligned} f(x) = \#\{\, z \mid 0 \leq z < s^{(r+1)^2} & \wedge \textstyle\bigwedge_{i=0}^{r} z_{i,0} = 2 \wedge \bigwedge_{i=1}^{r} z_{i,r} = 2 \\ & \wedge z_{0,1} = x_1 + \ell + 1 \wedge \textstyle\bigwedge_{j=2}^{n} z_{0,j} = x_j \wedge \bigwedge_{j=n+1}^{r} z_{0,j} = 2 \\ & \wedge z_{r,1} = 2\ell + 4 \wedge \textstyle\bigwedge_{j=2}^{r} z_{r,j} = 2 \\ & \wedge \textstyle\bigwedge_{i=1}^{r} \bigwedge_{j=1}^{r-1} \delta'_M(z_{i-1,j-1}, z_{i-1,j}, z_{i-1,j+1}, z_{i,j-1}, z_{i,j}, z_{i,j-1}) \end{aligned}$$

where δ'_M denotes the number theoretic predicate that corresponds to the word theoretic predicate δ_M.

We now arithmetize the above given formula. For a boolean expression α, let $[\alpha]$ be 1, if α is true, and 0 otherwise. Then we obtain the equation $(\star)$:

$$\begin{aligned} f(x) = \sum_{z=0}^{s^{(r+1)^2}-1} \prod_{i=0}^{r} \prod_{j=0}^{r} \quad & [z_{i,0} = 2] \cdot [z_{i,r} = 2] \\ & \cdot [z_{0,1} = x_1 + \ell + 1] \cdot [2 \leq j \leq n \Rightarrow z_{0,j} = x_j] \cdot [n+1 \leq j \leq r \Rightarrow z_{0,j} = 2] \\ & \cdot [z_{r,1} = 2\ell + 4] \cdot [j \geq 2 \Rightarrow z_{r,j} = 2] \\ & \cdot [(i \geq 1 \wedge 1 \leq j < r) \Rightarrow \delta'_M(z_{i-1,j-1}, z_{i-1,j}, z_{i-1,j+1}, z_{i,j-1}, z_{i,j}, z_{i,j-1})] \end{aligned}$$

For the different parts ot the Theorem, we now proceed as follows:

Statement 1, first equality: The two products of $(\star)$ can be merged as follows:

$$\begin{aligned} f(x) = \sum_{z=0}^{s^{(r+1)^2}-1} \prod_{t=0}^{r^2-1} \quad & [t \leq r \Rightarrow \text{bit}_s(z, t(r+1)) = 2] \cdot [t \leq r \Rightarrow \text{bit}_s(z, t(r+1)+r) = 2] \\ & \cdot [\text{bit}_s(z,1) = \text{bit}(x,0) + \ell + 1] \cdot [2 \leq t \leq |x| \Rightarrow \text{bit}_s(z,t) = \text{bit}(x, t-1)] \\ & \qquad \cdot [|x| + 1 \leq t \leq r \Rightarrow \text{bit}_s(z,t) = 2] \\ & \cdot [\text{bit}_s(z, (r+1)r + 1) = 2\ell + 4] \\ & \qquad \cdot [(r+1)r + 2 \leq t \leq (r+1)r + r + 1 \Rightarrow \text{bit}_s(z,t) = 2] \\ & \cdot [t \leq (r+1)r - 3 \Rightarrow \\ & \qquad \delta'_M(\text{bit}_s(z,t), \text{bit}_s(z,t+1), \text{bit}_s(z,t+2), \\ & \qquad\quad \text{bit}_s(z,t+r+1), \text{bit}_s(z,t+r+2), \text{bit}_s(z,t+r+3))] \end{aligned}$$

The remaining boolean conditions are then transformed according to the following rules (where α and β are boolean expressions):

$$\begin{array}{rcl} [u \leq v] & \rightsquigarrow & 1 \dot{-} (u \dot{-} v) \\ [u = v] & \rightsquigarrow & 1 \dot{-} ((u \dot{-} v) + (v \dot{-} u)) \\ [\alpha \wedge \beta] & \rightsquigarrow & [\alpha] \cdot [\beta] \\ [\neg\alpha] & \rightsquigarrow & 1 \dot{-} [\alpha] \\ [\alpha \vee \beta] & \rightsquigarrow & 1 \dot{-} (1 \dot{-} [\alpha])(1 \dot{-} [\beta]) \\ [\alpha \Rightarrow \beta] & \rightsquigarrow & [\neg\alpha \vee \beta] \end{array}$$

It now remains only to show that every use of a function bit_k can be replaced by a suitable use of bit. But this can be achieved using a simple block coding scheme. Finally, it is easy to see that δ'_M which is a finite predicate can be expressed using the just given expressions for boolean conditions.

Thus, we showed $\#\mathrm{P} \subseteq \mathrm{Sum}(\mathrm{WProd}([+, \dot{-}, \cdot, \lg, \mathrm{bit}]_{\mathrm{Sub}}))$. Since the base functions are in $\mathrm{FP}_+ \subseteq \#\mathrm{P}$, and since $\#\mathrm{P}$ is closed under summation and weak product, we proved the first equality of Statement 1.

Statement 2, first equality: Observe that in the proof of Statement 1, first equality, we constructed a sum, where every accepting path contributes 1 and every rejecting path contributes 0. We now have to change the above proof such that every rejecting path contributes -1. This is simply done as follows: Let $T(x, z, i, j)$ be the term under the sum and products in equation ($\star$). Replace $T(x, z, i, j)$ by $T(x, z, i, j) - T'(x, z, i, j)$, where $T'(x, z, i, j)$ is obtained from $T(x, z, i, j)$ by replacing $z_{r,1} = 2\ell + 4$ by $z_{r,1} = 3\ell + 5$. Now complete the arithmetization as in the above proof.

Statements 1 and 2, second equality: Immediate from the above, since $\#\mathrm{P}$ and Gap-P possess the closure properties under examination.

$\square$

3.2 Remark. The proof of Theorem 3.1 shows, that it is even sufficient to restrict oneself to zero-one-valued functions from the "inner" class of functions, i. e. that subclass defined using the substitution operator.

Strictly speaking, the functions lg and bit from the preceeding proof are not purely arithmetical. But one can show:

3.3 Corollary.

1. $\#\mathrm{P} = \mathrm{Sum}([+, \dot{-}, \cdot, :]_{\mathrm{Sub},\mathrm{WProd}}) = \Big[[+, \dot{-}, \cdot, :]_{\mathrm{Sub},\mathrm{WProd}}\Big]_{\mathrm{Sum}}$.

2. $\text{Gap-P} = \mathrm{Sum}([+, -, \dot{-}, \cdot, :]_{\mathrm{Sub},\mathrm{WProd}}) = \Big[[+, -, \dot{-}, \cdot, :]_{\mathrm{Sub},\mathrm{WProd}}\Big]_{\mathrm{Sum}}$.

Proof. Follows immediately from Theorem 3.1, taking into account that $\mathrm{bit}(z, i) = (z : 2^i) \dot{-} 2 \cdot (z : 2^{i+1})$, $2^i = \prod_{j=1}^{|x|} 2 \cdot (1 \dot{-} (j \dot{-} i))$ for $i \leq |x|$, and $|x| = \prod_{i=0}^{|x|} f(x, i)$, where

$$f(x, i) =_{\mathrm{def}} \left\{ \begin{array}{ll} 1, & \text{if } i < |x| \\ i, & \text{if } i \geq |x| \end{array} \right\} = (i - 1) \cdot (1 \dot{-} (1 \dot{-} (2^i \dot{-} x))) + 1$$

$\square$

In the just given results, we had to introduce the operation of integer division to simulate bit access operations. Naturally, the question arises whether we can do without it. Our next theorem answers this question positively, but it seems that now, we are required to have weak product as an outer operator. Additionally, we show that in the characterization of Gap-P, we can even replace modified subtraction by decrement (i. e. modified subtraction of 1).

3.4 Theorem.

1. $\#\mathrm{P} = \Big[[+, \dot{-}, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{WProd},\mathrm{Sum}}$

2. $\text{Gap-P} = \Big[[+, -, \dot{-}, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{WProd},\mathrm{Sum}} = \Big[[+, -, \dot{-}1, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{WProd},\mathrm{Sum}}$

Proof. Our proof heavily depends on an arithmetical characterization of NP, given in [15, 11]. Building on results of Martin Davis [6] about r.e. sets, Kent and Hodgson give a normal form for arithmetical represenations of NP sets. The proof of their result reveals the following:

For every nondeterministic polynomial time machine M, there exist a number $k \geq 0$ and polynomials $q_0, q_1, \ldots, q_k, p, p_1, p_2$ with nonnegative integer coefficients such that M accepts some input x, $|x| = n$, if and only if

$$(\exists z \leq 2^{p(n)})(\forall u \leq q_0(n))(\exists v_1 \leq 2^{q_1(n)}) \cdots (\exists v_k \leq 2^{q_k(n)})(p_1(x, z, u, \bar{v}) = p_2(x, z, u, \bar{v})),$$

where $\bar{v}$ abbreviates $(v_1, \ldots, v_k)$. Moreover, the proof given in [15] shows that for every accepting computation path of M on x, there exists exactly one z as above, and vice versa, and for every (x, z, u), there exists at most one $\bar{v}$ wich fulfills the equality. Thus, we immediately have that the number of accepting computation paths of M on input x is

$$\sum_{z \leq 2^{p(n)}} \prod_{u \leq q_0(n)} \sum_{v_1 \leq 2^{q_1(n)}} \cdots \sum_{v_k \leq 2^{q_k(n)}} [p_1(x, z, u, \bar{v}) = p_2(x, z, u, \bar{v})].$$

Taking into consideration that $[p_1 = p_2] = (2p_1p_2 + 1) \dot{-} (p_1^2 + p_2^2)$, we get

$$\#\mathrm{P} \subseteq \Big[[+, \cdot]_{\mathrm{Sub}} \dot{-} [+, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{Sum,WProd}} \subseteq \Big[[+, \dot{-}, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{Sum,WProd}}$$

and Statement 1 follows immediately.

To prove Statement 2, we recall that every function from Gap-P is the difference of a $\#$P function and 2 raised to the power of a suitable polynomial [8]. Such a sum

$$\Big(\sum_{z \leq 2^{p(n)}} \prod_{u \leq q_0(n)} \sum_{v_1 \leq 2^{q_1(n)}} \cdots \sum_{v_k \leq 2^{q_k(n)}} [p_1(x, z, u, \bar{v}) = p_2(x, z, u, \bar{v})] \Big) - 2^{t(n)}$$

for a polynomial t, can however be written as

$$\sum_{a \leq 2^{t(n)}} \sum_{z \leq 2^{p(n)}} \prod_{u \leq q_0(n)} \sum_{v_1 \leq 2^{q_1(n)}} \cdots \sum_{v_k \leq 2^{q_k(n)}} ([p_1(x, z, u, \bar{v}) = p_2(x, z, u, \bar{v})][a = 0] - [a > 0][z = 0][\bar{v} = \bar{0}])$$

Taking into consideration that $[a > 0] = 1 \dot{-} (1 \dot{-} a))$, $[a = b] = 1 \dot{-} (a - b)^2$, and $1 \dot{-} a = ((a \dot{-} 1) + 1) - a$ for $a \geq 0$, we obtain Gap-P $\subseteq \Big[[+, \dot{-} 1, -, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{Sum,WProd}}$, and thus Statement 2 is immediate. □

The question which now arises naturally is, whether we can even get rid of modified subtraction, or equivalently, wether the closure of all polynomials with integer coefficients (nonnegative integer coefficients) under summation and weak product is already equal to Gap-P ($\#$P, resp.). Next, we show that this is very unlikely.

3.5 Theorem. *If* $\#\mathrm{P} = \Big[[+, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{Sum,WProd}}$ *or* Gap-P $= \Big[[+, -, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{Sum,WProd}}$ *then* $\mathrm{P} = \oplus\mathrm{P}$, *and hence the polynomial time hierarchy collapses to its second level.*

Proof. Both proofs are completely analogous. We prove the second statement.

Let $A \in \oplus\mathrm{P}$, then there exists a function $f \in$ Gap-P such that $x \in A$ if and only if $f(x)$ is odd. If now $f \in \Big[[+, -, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{Sum,WProd}}$, then we can answer the question whether $f(x)$ is odd deterministically in polynomial time, since we can use the structure of the recursive definition of f to evaluate $f(x)$ modulo 2 inductively as follows:

Let q be a polynomial with variables $(x, \bar{z})$ (where $\bar{z}$ abbreviates $(z_1, z_2, \cdots, z_k)$ for some k), i.e. $q \in \mathbb{Z}[x, \bar{z}]$. Whether $q(x, \bar{z}) \equiv 0 \pmod 2$ depends only on whether the $x, \bar{z}$ are even or odd, that is there exists a polynomial $p \in \mathrm{GF}_2[x, \bar{z}]$ such that $q(x, \bar{z}) \equiv p(x, \bar{z}) \pmod 2$. The evaluation of $f(x)$ modulo 2 now consists of repeated applications of the following steps: Let $p \in \mathrm{GF}_2[x, \bar{y}, z]$. Then

1. $\prod_z p(x, \bar{y}, z) \equiv p(x, \bar{y}, 0) \cdot p(x, \bar{y}, 1) \pmod 2$, since a product is odd if and only if all its factors are odd.
2. $\sum_z^{2^s} p(x, \bar{y}, z) \equiv p(x, \bar{y}, 0) \pmod 2$, if $s \neq 0$, since in this case we have an even number of terms for z odd (which always sum up to an even number) and an odd number of terms for z even (which sum up to an odd number if $p(x, \bar{y}, 0)$ is odd). If $s = 0$, we have two terms, and obviously $\sum_z^{2^s} p(x, \bar{y}, z) \equiv p(x, \bar{y}, 0) + p(x, \bar{y}, 1) \pmod 2$.

Which of the formulas has to be used in step 2 depends on the length of x and the length of the variables introduced in operators to the left of s. However, all these lengths are bounded by polynomials in the length of x. By repeated application of the above steps, we therefore obtain a formula containing a constant number of $p(\alpha_1, \alpha_2, \ldots, \alpha_k)$ with $\alpha_1, \alpha_2, \ldots, \alpha_k \in \{0, 1\}$, which is easy to evaluate.

Finally, a result by Toda [21] shows that the polynomial time hierarchy is included in the second level of the polynomial time hierarchy over $\oplus\mathrm{P}$. Thus, $\oplus\mathrm{P} = \mathrm{P}$ implies the collapse of the polynomial time hierarchy to its second level. □

4. The hierarchy of counting functions

The hierarchy of counting functions can be characterized as the closure of $\#\mathrm{P}$ under summation and substitution [25]. In this section, we give an alternative algebraic characterization which again as in our previous results doesn't need the operation of substitution.

4.1 Lemma. *For $k \geq 0$, $(k+1)\#\mathrm{P} = \mathrm{Sum}(k\#\mathrm{P} \dot{-} k\#\mathrm{P})$.*

Proof. In [25], it is proved that for every $f \in (k+1)\#\mathrm{P}$, there exist functions $g \in \mathrm{FP}_+$ and $h \in k\#\mathrm{P}$ and a polynomial p such that $f(x) = \sum_{z=0}^{2^{p(|x|)}} g(h(x, z))$.

Let $t(a, b) =_{\mathrm{def}} (2ab + 1) \dot{-} (a^2 + b^2)$. Then $t(a, b) = 1 \iff a = b$ and $t(a, b) = 0 \iff a \neq b$ for $a, b \geq 0$.

Let q be a polynomial such that $h(z) \leq 2^{q(|x|)}$ for $z \leq 2^{p(|x|)}$. Then,

$$f(x) = \sum_{z=0}^{2^{p(|x|)}} \sum_{u=0}^{2^{q(|x|)}} g(u) \cdot t(u, h(x, z)) = \sum_{z=0}^{2^{p(|x|)}} \sum_{u=0}^{2^{q(|x|)}} (g(u) \cdot (2uh(x, z) + 1)) \dot{-} (g(u) \cdot (u^2 + h(x, z)^2)).$$

Hence, $(k+1)\#\mathrm{P} \subseteq \mathrm{Sum}(k\#\mathrm{P} \dot{-} k\#\mathrm{P})$ (since $\mathrm{Sum}\,\mathrm{Sum}\,\mathcal{F} = \mathrm{Sum}\,\mathcal{F}$ for all function classes $\mathcal{F}$ with $\mathcal{F} \circ \mathrm{FP} \subseteq \mathcal{F}$ [24]). On the other hand, $\mathrm{Sum}(k\#\mathrm{P} \dot{-} k\#\mathrm{P}) \subseteq \mathrm{Sum}\,\mathrm{FP}_+^{k\#\mathrm{P}} \subseteq \#\mathrm{P}^{k\#\mathrm{P}} \subseteq (k+1)\#\mathrm{P}$, because the inclusion $\mathrm{Sum}\,\mathrm{FP}_+ \subseteq \#\mathrm{P}$ is relativizable. □

4.2 Corollary. $\mathrm{FCH} = [\#\mathrm{P}]_{\mathrm{Sum}, \dot{-}}$

Building on our results from the previous section, this representation of FCH can be simplified. To generate all functions from FCH using Sum and $\dot{-}$, it is sufficient to start with polynomials with nonnegative integer coefficients rather than $\#\mathrm{P}$ functions.

4.3 Theorem. $\mathrm{FCH} = \Big[[+, \cdot]_{\mathrm{Sub}}\Big]_{\mathrm{Sum}, \dot{-}}$

Proof. From Corollary 4.2 and the proof of Theorem 3.4, we obtain $\mathrm{FCH} = \left[[+,\cdot]_{\mathrm{Sub}}\right]_{\mathrm{Sum},\mathrm{WProd},\dot{-}}$. Since in fact, we use the WProd operator only for 0-1-valued functions f, we have in this case

$$\prod_{u=0}^{p(|x|)} f(x,u) = 1 \dot{-} \sum_{u=0}^{p(|x|)} (1 \dot{-} f(x,u)).$$

From Kent and Hodgson's result $\mathrm{NP} \subseteq \mathcal{E}_3$ (see [11, Section 3]; see also [15, remark on top of p. 261]), we see that we can raise the operator bound $p(|x|)$ to $2^{p'(|x|)}$ for a suitable polynomial p'. □

For use in the next section, we remark that denoting by Var some countable set of variables (representing essentially the set of all identity functions i_k^n), we obtain

4.4 Corollary. $\mathrm{FCH} = [\mathrm{Var}]_{\mathrm{Sum},+,\cdot,\dot{-}}$

The class Gap-P is the integer analogue to the class # P of everywhere nonnegative functions. To get an analogue for the class FCH, we define the *hierarchy of gap functions* to consist of the classes $0 \cdot \text{Gap-P} =_{\mathrm{def}} \mathrm{FP}$ and $k \cdot \text{Gap-P} =_{\mathrm{def}} \text{Gap-P}^{(k-1)\text{Gap-P}}$ for $k \geq 1$. Let $\mathrm{FCH}_{\mathbb{Z}} = \bigcup_{k \geq 0} k \cdot \text{Gap-P}$.

Since the result $\text{Gap-P} = \#\mathrm{P} - \#\mathrm{P} = \#\mathrm{P} - \mathrm{FP}$ relativizes, we obtain

4.5 Proposition.

1. $k \cdot \text{Gap-P} = k\,\#\mathrm{P} - k\,\#\mathrm{P} = k\,\#\mathrm{P} - \mathrm{FP}$.
2. $\mathrm{FCH}_{\mathbb{Z}} = \text{Gap-P}^{\mathrm{FCH}} = \mathrm{FP}^{\mathrm{FCH}} = \mathrm{FCH} - \mathrm{FCH} = \mathrm{FCH} - \mathrm{FP}$.

From Proposition 4.5 and Theorem 4.3, we obtain:

4.6 Corollary.
$$\begin{aligned} \mathrm{FCH}_{\mathbb{Z}} &= [\#\mathrm{P}]_{\mathrm{Sum},\dot{-},-} = [\text{Gap-P}]_{\mathrm{Sum},\dot{-},-} = [\#\mathrm{P}]_{\mathrm{Sum},\dot{-}} - \mathrm{FP} \\ &= \left[[+,\cdot]_{\mathrm{Sub}}\right]_{\mathrm{Sum},\dot{-},-} = \left[[+,\cdot]_{\mathrm{Sub}}\right]_{\mathrm{Sum},\dot{-}} - \mathrm{FP} \\ &= [\mathrm{Var}]_{\mathrm{Sum},+,-,\cdot,\dot{-}} \end{aligned}$$

5. Other Operations and Applications

In this section, we will see how the just obtained results allow us to give a number of interesting consequences, some of which were already proven in [16, 20].

5.1 Corollary. # P *is closed under modified subtraction if and only if the hierarchy of counting functions collapses to* # P *if and only if the counting hierarchy collapses to* UP.

Equivalence of the first and third statement of Corollary 5.1 were already shown in [16].

Proof. The hierarchy of counting functions is closed under every FP_+ operation, therefore the second statement implies the first.

If # P is closed under modified subtraction, then $[\#\mathrm{P}]_{\dot{-},\mathrm{Sum}} = \#\mathrm{P}$; thus the first statement implies the second as a consequence of Corollary 4.2. □

Analogously, we obtain:

5.2 Corollary. Gap-P *is closed under modified subtraction if and only if the hierarchy of gap functions collapses to* Gap-P *if and only if the counting hierarchy collapses to* SPP.

Proof. The hierarchy of gap functions is closed under every FP operation; thus the second statement implies the first.

If Gap-P is closed under modified subtraction, then $[\text{Gap-P}]_{\text{Sum},\dot{-},-} = \text{Gap-P}$; thus the first statement implies the second as a consequence of Corollary 4.6.

If SPP is equal to the counting hierarchy, then the counting hierarchy is low for Gap-P; however, Gap-P with oracles from the counting hierarchy is exactly the hierarchy of gap functions. Thus, the third statement implies the second.

If $\text{FCH}_{\mathbb{Z}} = \text{Gap-P}$, then the class of all sets whose characteristic function is in $\text{FCH}_{\mathbb{Z}}$ (which is the counting hierarchy) is equal to SPP. Thus, the second statement implies the third. □

In the next results, we will see how we can, building on our recursion theoretic characterizations, obtain results analogous to Theorems 4.2 and 4.6 and the just given two Theorems, but for other arithmetical operations. Because of the results from the previous sections, if we want to prove an analogue to, say, Theorem 5.1, for an operation $\circ$, all we have to do is show that an application of $\dot{-}$ can be equivalently replaced by a sequence of operations including those under which $\#\text{P}$ is closed and the operator $\circ$; that is, we in a sense reduce $\dot{-}$ to $\circ$. Thus, if $\#\text{P}$ were closed under $\circ$, then it would be closed under $\dot{-}$. This form of reduction has been defined precisely and a number of other applications have been given recently in [10].

5.3 Theorem. $\text{FCH} = [\#\text{P}]_{\text{Sum},+,\cdot,:} = [\text{Var}]_{\text{Sum},+,\cdot,:}$

Proof. For $x, y \geq 0$, we have

$$\begin{aligned} x \dot{-} y &= \#\{\, i \mid 0 \leq i \leq x \wedge y + 1 + i \leq x \,\} \\ &= \#\{\, i \mid 0 \leq i \leq x \wedge (\exists j, 0 \leq j \leq x)(x + i + j = y) \,\} \\ &= \sum_{0 \leq i \leq y} \sum_{0 \leq j \leq y} [x + i + j = y], \end{aligned}$$

and $[x = y] = ((u+1) : (v+1)) \cdot ((v+1) : (u+1))$. So, the closure of $\#\text{P}$ under summation and modified subtraction is included in the closure of $\#\text{P}$ under summation, addition, multiplication, and integer division. The other direction is obvious, since FCH is closed under summation as well as every FP_+ operation. □

5.4 Corollary. $\#\text{P}$ *is closed under integer division if and only if the hierarchy of counting functions collapses to* $\#\text{P}$ *if and only if the counting hierarchy collapses to* UP.

Proof. Follows directly from the preceding theorem, analogous to the proof of Corollary 5.1. □

5.5 Theorem. $\begin{aligned}[t] \text{FCH}_{\mathbb{Z}} &= [\text{Gap-P}]_{\text{Sum},-,\max} = [\text{Var}]_{\text{Sum},+,-,\cdot,\max} \\ &= [\text{Gap-P}]_{\text{Sum},-,\min} = [\text{Var}]_{\text{Sum},+,-,\cdot,\min} \end{aligned}$

Proof. Modified subtraction can be reduced to maximum by $x \dot{-} y = \max\{0, x - y\}$. Maximum can be reduced to mininum and subtraction by $\max\{a, b\} = b - \min\{0, b - a\}$. The theorem then follows directly from Corollary 4.6 □

5.6 Corollary. Gap-P *is closed under maximum if and only if* Gap-P *is closed under minimum if and only if the hierarchy of gap functions collapses to* Gap-P *if and only if the counting hierarchy collapses to* SPP.

Equivalence of the first, second, and fourth statement was already shown in [20].

5.7 Theorem. $\text{FCH}_{\mathbb{Z}} = [\text{Gap-P}]_{\text{Sum},+,-,\cdot,:} = [\text{Var}]_{\text{Sum},+,-,\cdot,:}$

Proof. Exactly as in the proof of Theorem 5.3. □

5.8 Corollary. Gap-P *is closed under integer division if and only if the hierarchy of gap functions collapses to* Gap-P *if and only if the counting hierarchy collapses to* SPP.

Eventually, we consider the operation decrement ($\dot{-}\,1$), i.e. modified subtraction of 1.

5.9 Theorem. $\mathrm{FCH}_{\mathbb{Z}} = [\text{Gap-P}]_{\mathrm{Sum},+,-,\cdot,\dot{-}1} = [\mathrm{Var}]_{\mathrm{Sum},+,-,\cdot,\dot{-}1}$

Proof. The result follows from the following equations: $a \dot{-} b = \sum_{z=0}^{a}[b \leq z] \cdot [z+1 \leq a]$, $[a \leq b] = \sum_{z=0}^{b}[a + z = b]$, $[a = b] = 1 \dot{-} (a-b)^2$, and $1 \dot{-} a = ((a \dot{-} 1) + 1) - a$ for $a \geq 0$. ☐

5.10 Corollary. Gap-P *is closed under decrement if and only if the hierarchy of gap functions collapses to* Gap-P *if and only if the counting hierarchy collapses to* SPP.

In [20], it was already shown that Gap-P_+, the class of all everywhere nonnegative Gap-P functions, is closed under decrement if and only if the counting hierarchy collapses to SPP.

Acknowledgement. The ideas of this paper developed during a seminar on Complexity Theory on Burg Rothenfels in March 1994. We would like to thank the other participants of the seminar, i.e. Herbert Baier, Gerhard Buntrock, Ulrich Hertrampf, and Diana Rooß, for stimulating and helpful discussions.

References

[1] L. Babai, L. Fortnow, Arithmetization: A new method in structural complexity theory; *Computational Complexity* **1** (1991) 41–66.

[2] J. L. Balcázar, J. Díaz, J. Gabarró, *Structural Complexity I* (Springer Verlag, Berlin – Heidelberg – New York, 2nd edition, 1995).

[3] P. Clote, Sequential, machine independent characterizations of the parallel complexity classes AlogTIME, AC^k, NC^k and NC; in: S.R. Buss and P.J. Scott, eds., *Feasible Mathematics* (Birkhäuser Verlag, Boston – Basel – Berlin, 1990) 49–69.

[4] P. Clote, Bounded arithmetic and computational complexity; in: *Proceedings of the 5th Structure in Complexity Theory Conference* (1990) 186–199.

[5] A. Cobham, The intrinsic computational complexity of functions, in: *Proceedings of the 1964 International Conference on Logic, Methodology and Philosophy of Science* (1964) 24–30.

[6] M. Davis, Arithmetical problems and recursively enumerable predicates; *Journal of Symbolic Logic* **18** (1953), 33–41.

[7] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets; in: R. Karp (ed.), *The Complexity of Computation,* SIAM-AMS Proceedings Vol. 7 (American Mathematical Society, Providence RI, 1974) 43–73.

[8] S. Fenner, L. Fortnow, S. Kurtz, Gap-definable counting classes; *Journal of Computer and System Sciences* **48** (1994) 116–148.

[9] J. Gill, Computational complexity of probabilistic complexity classes; *SIAM Journal on Computing* **6** (1977) 675–695.

[10] U. Hertrampf, H. Vollmer, K.W. Wagner, On the power of number-theoretic operations with respect to counting; in: *Proceedings of the 10th Structure in Complexity Theory Conference* (1995) 299–314.

[11] B. Hodgson, C. Kent, A normal form for arithmetical representations of NP-sets; *Journal of Computer and System Sciences* **27** (1993) 378–388.

[12] J. E. Hopcroft, J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation,* (Addison-Wesley, Reading, Mass., 1979).

[13] N. Immerman, Languages that capture complexity classes; *SIAM J. Computing* **16** (1987) 760–778.

[14] N. Immerman, Expressibility and parallel complexity; *SIAM J. Computing* **18** (1989) 625–638.

[15] C. Kent, B. Hodgson, An arithmetical characterization of NP; *Theoretical Computer Science* **21** (1982) 255–267.

[16] M. Ogiwara, L. Hemachandra, A complexity theory for feasible closure properties; *Journal of Computer and System Sciences* **46** (1993) 295–325.

[17] S. Saluja, K. V. Subrahmanyam, M. N. Thakur, Descriptive complexity of # P functions; in: *Proceedings of the 7th Structure in Complexity Theory Conference* (1992) 169–184.

[18] A. Shamir, IP = PSPACE; in: *Proceedings of the 31st Symposium on Foundations of Computer Science* (1990) 11–15.

[19] M. Sipser, The history and status of the P versus NP question; in: *Proceedings of the 14th Symposium on the Theory of Computing* (1992) 603–618.

[20] T. Thierauf, S. Toda, O. Watanabe, On closure properties of Gap-P; *Computational Complexity* **4** (1994) 242–261.

[21] S. Toda, PP is as hard as the polynomial time hierarchy; *SIAM Journal on Computing* **20** (1991) 865–877.

[22] L. G. Valiant, Relative complexity of checking and evaluation, *Information Processing Letters* **5** (1976) 20–23.

[23] L. G. Valiant, The complexity of computing the permanent; *Theoretical Computer Science* **8** (1979) 189–201.

[24] H. Vollmer, K. W. Wagner, Complexity classes of optimization functions; *Information and Computation* **120** (1995) 198–219.

[25] K. W. Wagner, Some observations on the connection between counting and recursion; *Theoretical Computer Science* **47** (1986) 131–147.

[26] K. W. Wagner, G. Wechsung, *Computational Complexity* (Deutscher Verlag der Wissenschaften, Berlin, 1986).