Acta Informatica 23, 325–356 (1986)

Acta Informatica
© Springer-Verlag 1986

The Complexity of Combinatorial Problems with Succinct Input Representation*

Klaus W. Wagner

Fakultät für Informatik, Universität Passau, D-8390 Passau, Federal Republic of Germany

Summary. Several languages for the succinct representation of the instances of combinatorial problems are investigated. These languages have been introduced in [20, 2] and [5] where it has been shown that describing the instances by these languages causes a blow-up of the complexities of some problems. In the present paper the descriptional power of these languages is compared by estimating the complexities of some combinatorial problems in terms of completeness in suitable classes of the "counting polynomial-time hierarchy" which is introduced here. It turns out that some of the languages are not comparable, unless $\mathscr{P} = \mathscr{NP}$. Some problems left open in [2] are solved.

1. Introduction

In order to be able to speak about the complexity of a combinatorial problem it is necessary to specify the language used to describe the instances of the problem. In the case that the instances are finite sets they are usually described by a report (i.e. a list) of their elements. As frequently used examples take lists of natural numbers, lists of edges of finite graphs and lists of geometrical objects in the plane. However, if finite sets have a lot of regularities (this appears especially in practically important cases, for example in VLSI design, in computer graphics and architectural design) it can be possible and advantageous to describe these sets more succinctly by special languages which avoid the repeated description of similarly structured subsets. Such languages for the succinct representation of inputs have been developped and investigated recently in several papers. In [2] a hierarchical language for describing finite sets of rectangles in the plane has been specified which allows to refer to a set of rectangles by name, after the set has been defined. In [5] finite graphs have

* A preliminary version of this paper appeared in the proceedings of the Second Frege Conference held in September 1984 in Schwerin (GDR), see [24]

been represented by combinatorial circuits. In [9–11] and [12] finite graphs have been described by "hierarchical graphs", i.e. by a language allowing to replace certain vertices of a graph by other graphs in a hierarchical manner.

Using the above mentioned languages to describe the instances of combinatorial problems succinctly one can make the following observation (see [2, 5, 9–12]): Since the computational complexity of algorithms is measured in terms of the length of the input and since a succinct description of length n can describe finite sets with an exponentially large number of elements we can have an exponential blow-up in the computational complexity of the algorithms. And in some cases the inherent computational complexity of combinatorial problems really has this exponential blow-up when the above mentioned input languages are used. This phenomenon can be interpreted as follows: Though only very regular (in a sense determined by the used input language) inputs can be described succinctly, the difficulties of these problems do not necessarily disappear when considering only such regular inputs. This phenomenon has been studied already much more earlier from a purely theoretical point of view in [20] where it has been shown that the use of integer-expressions to describe finite sets of natural numbers can make some combinatorial problems of very low complexity $\mathcal{NP}$-hard, and that the use of different kinds of regular-like expressions to describe finite automata (i.e. regular languages) results in at least an exponential blow-up of the complexity of the membership problem. In [14] it has even been shown that the use of the weak monadic second-order theory of successor to describe finite automata causes a non-elementary recursive blow-up of the complexity of the membership problem.

The aim of the present paper is to compare some of the languages investigated in [2, 5] and [20] by estimating the complexities of some selected combinatorial problems with respect to each of these languages. Doing so we are able to improve some results in [2]. It turns out that some of these languages can be compared with each other with respect to their computational power and that some others cannot be compared (unless $\mathcal{P}=\mathcal{NP}$). In order to localize the complexities of the problems in terms of completeness in natural complexity classes the classes of the polynomial-time hierarchy (see [21]) do not suffice. We need in addition some classes of the "counting polynomial-time hierarchy" which is introduced here. We give some basic results on the classes of this hierarchy, and we exhibit complete sets for every class of the hierarchy.

The paper is organized as follows. In Sect. 2 we adapt the integer-expressions from [20], the general hierarchic input language from [2] and the combinatorial circuits from [5] to describe arbitrary finite subsets of $\mathbb{N}^n$. This enables us to compare these languages. In Sect. 3 we introduce and investigate the "counting polynomial-time hierarchy", and in Sect. 4 we estimate the complexities of the problems we are interested in with respect to each of the descriptional languages.

2. Input Languages

In this section we adapt four input languages known from the literature to describe arbitrary finite subsets of $\mathbb{N}^n$ ($\mathbb{N}$ being the set of natural numbers and

$n \geqq 1$). This will enable us to compare the descriptional power of these languages.

All natural numbers occuring in the descriptions of these languages are thought to be written in binary notation.

2.1. *Integer-Expressions*

We start with the integer-expression language introduced in [20] to describe finite subsets of $\mathbb{N}$. To describe finite subsets of $\mathbb{N}^n$ for arbitrary $n \geqq 1$ the definition has to be made as follows.

1. $(a_1, \ldots, a_n)$ is an (n-dimensional) integer-expression for $a_1, \ldots, a_n \in \mathbb{N}$.
2. If H_1, H_2 are (n-dimensional) integer-expressions then $(H_1 \cup H_2)$ and $(H_1 + H_2)$ are (n-dimensional) integer-expressions,
3. Nothing else is an (n-dimensional) integer-expression.

Every n-dimensional integer-expression H describes a finite set $L(H) \subseteq \mathbb{N}^n$ which is defined as follows:

1. $L((a_1, \ldots, a_n)) = \{(a_1, \ldots, a_n)\}$,
2. $L((H_1 \cup H_2)) = L(H_1) \cup L(H_2)$,
   $L((H_1 + H_2)) = \{(a_1 + b_1, \ldots, a_n + b_n)\colon (a_1, \ldots, a_n) \in L(H_1)$ and $(b_1, \ldots, b_n) \in L(H_2)\}$.

For example, for $r \geqq 1$, the one-dimensional integer-expression

$$H_r \equiv (\ldots(((0 \cup 1) + (0 \cup 2)) + (0 \cup 4)) + \ldots + (0 \cup 2^{r-1}))$$

describes the set $L(H_r) = \{0, 1, 2, 3, \ldots, 2^r - 1\}$. Since the length of H_r is of order r^2 we observe that an integer-expression of length m can describe a number of elements which is exponentially in m. By induction it can easily be seen that this is also an upper bound. Let $|w|$ denote the length of the word w.

Proposition 1. *For every integer-expression H,*

1. $\operatorname{card} L(H) \leqq 2^{|H|}$,
2. $|x| \leqq |H|$ *for all* $x \in L(H)$. □

The description of the elements of $L(H)$ by the n-dimensional integer-expression H can be represented by the acyclic directed graph $G(H)$ having edges labelled by n-tuples of natural numbers in the following way: $a \in L(H)$ iff there exists a path from the source to the sink of $G(H)$ such that a is the sum of all labels on this path. The graphs $G(H)$ are defined inductively as follows:

1. The graph $G((a_1, \ldots, a_n))$ is shown in Fig. 1a.
2. The graphs $G((H_1 \cup H_2))$ and $G((H_1 + H_2))$ can be constructed from $G(H_1)$ and $G(H_2)$ as shown in Fig. 1b and Fig. 1c, resp.

As an example, for $H_3 \equiv (((0 \cup 1) + (0 \cup 2)) + (0 \cup 4))$ the graph $G(H_3)$ is shown in Fig. 2.

The n-tuple corresponding to the path x of $G(H)$ is denoted by $\varphi_H(x)$. Obviously, if $a \in \mathbb{N}^n$ is described by H exactly k times then there exist exactly k different paths x such that $\varphi_H(x) = a$ and vice versa. We say that k is the multiplicity of the description of a by H, and we define

$$\operatorname{mult}_H(a) = \operatorname{card}\{x\colon \varphi_H(x) = a\}.$$

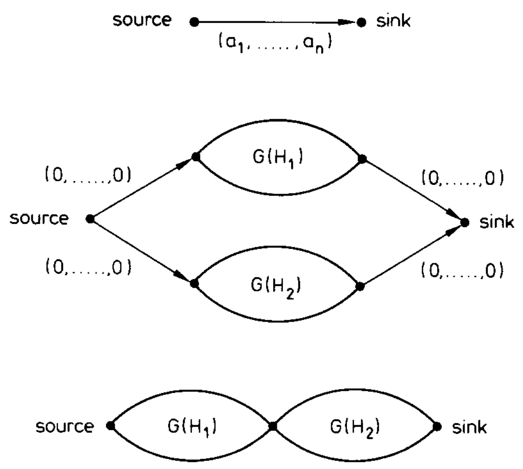

Fig. 1. a The graph $G((a_1, \ldots, a_n))$. **b** The graph $G((H_1 \cup H_2))$. **c** The graph $G((H_1 + H_2))$

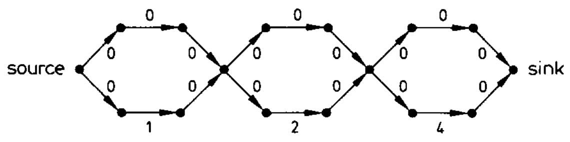

Fig. 2. The graph $G(H_3)$

Evidently, $a \in L(H)$ iff $\mathrm{mult}_H(a) > 0$. Furthermore, the multiplicities of the integer-expression H are bounded in the same way as the cardinality of $L(H)$.

Proposition 2. *For every integer-expression H and every $a \in L(H)$,*

$$\mathrm{mult}_H(a) \leq 2^{|H|}. \quad \square$$

Finally, it is not hard to see that $\varphi_H(x)$ is computable in polynomial time.

Proposition 3. *Given an integer-expression H and an $x \in \{0, 1\}^*$, one can compute $\varphi_H(x)$ in polynomial time (in the size of H and x) where the result is "no" if x is not a path in $G(H)$.* $\square$

We conclude this subsection by constructing some one-dimensional integer-expressions which are needed in Sect. 4.

Lemma 1. *There exists a logarithmic-space bounded machine computing to every input $r \in \mathbb{N}$ (given in binary notation) a one-dimensional integer-expression $H(r)$ such that*

1. $L(H(r)) = \{0, 1, 2, 3, \ldots, r\}$,
2. $\mathrm{mult}_{H(r)}(a) \leq 1$ *for all* $a \in \mathbb{N}$, *and*
3. $|H(r)| \leq c \cdot \log^3 r$, *for suitable* $c > 0$.

Proof. Let $r=\sum_{i=0}^{k}\alpha_i\cdot 2^i$ such that $\alpha_0,\alpha_1,\ldots,\alpha_k\in\{0,1\}$, and let

$$\{i_1,i_2,\ldots,i_s\}=\{i\colon 0\leqq i\leqq k \text{ and } \alpha_i=1\} \qquad \text{where } i_1>i_2>\ldots>i_s.$$

We obtain

$$\{0,1,2,3,\ldots,r\}=L(H(2^{i_1}-1)\cup(2^{i_1}+(H(2^{i_2}-1) \\ \cup(2^{i_2}+(H(2^{i_3}-1)\cup\ldots\cup(2^{i_{s-1}}+(H(2^{i_s}-1)\cup 2^{i_s}))\ldots))))).$$

Because of $L(H(2^t-1))=L(H_t)$ (H_t being defined above) we can define

$$H(r)\equiv(H_{i_1}\cup(2^{i_1}+(H_{i_2}\cup(2^{i_2}+(H_{i_3}\cup\ldots\cup(2^{i_{s-1}}+(H_{i_s}\cup 2^{i_s}))\ldots))))).$$

By the construction of $H(r)$ it is obvious that non of its multiplicities can be greater than one. Furthermore, denoting the binary representation of i by bin i,

$$\begin{aligned}|H(r)|&=6s-3+\sum_{j=1}^{s}|H_{i_j}|+\sum_{j=1}^{s}\operatorname{bin}|2^{i_j}|\\ &\leqq 6(k+1)-3+(k+1)\cdot|H_k|+(k+1)\cdot|\operatorname{bin}2^k|\\ &\leqq c_1\cdot k+c_2\cdot k^3+c_3\cdot k^2 \quad \text{for suitable } c_1,c_2,c_3>0\\ &\leqq c\cdot\log^3 r \quad \text{for suitable } c>0. \quad \square\end{aligned}$$

2.2. *The General Hierarchic Input Language* (GHIL)

This language has been introduced in [2] to describe finite sets of rectangles in the plane which are important for VLSI design and other applications. The idea is to define several finite sets of rectangles, to give them names and to refer to them by name when defining new finite sets of rectangles. Thus large designs can be built by replicating smaller designs. The operations on the finite sets of rectangles are union and translation.

To describe finite subsets of $\mathbb{N}^n$ (for arbitrary $n\geqq 1$) we modify the definitions in [2] as follows:

1. $(a_1,\ldots,a_n)$ is an (n-dimensional) GHIL-expression, for $a_1,\ldots,a_n\in\mathbb{N}$.
2. $\langle i\rangle$ is an (n-dimensional) GHIL-expression for $i\geqq 0$ ($\langle i\rangle$ is interpreted as the name of the i-th GHIL-expression already defined).
3. If H_1, H_2 are (n-dimensional) GHIL-expressions then $(H_1\cup H_2)$ is an (n-dimensional) GHIL-expression.
4. If H is an (n-dimensional GHIL-expression and $a_1,\ldots,a_n\in\mathbb{N}$ then $(H+(a_1,\ldots,a_n))$ is an (n-dimensional) GHIL-expression.
5. Nothing else is an (n-dimensional) GHIL-expression.

Now an (n-dimensional GHIL description is a sequence $D\equiv(H_0, H_1,\ldots,H_r)$ where $r\geqq 0$ and $H_0,H_1,\ldots,H_r$ are (n-dimensional GHIL-expressions satisfying the condition: if $\langle i\rangle$ occurs in H_m then $i<m$. This means: if

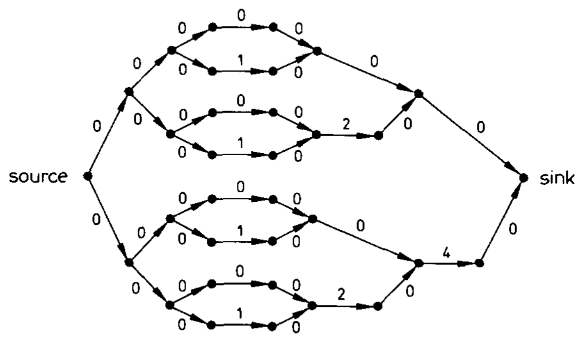

Fig. 3. The graph $G(D_3)$

the name $\langle i\rangle$ is used in the definition of H_m then the i-th GHIL-expression H_i must be already defined.

Every n-dimensional GHIL description $D \equiv (H_0, H_1, \ldots, H_r)$ describes a finite set $L(D) \underset{\mathrm{df}}{=} L(H_r)$ where

1. $L((a_1, \ldots, a_n)) = \{(a_1, \ldots, a_n)\}$,
2. $L(\langle i\rangle) = L(H_i)$,
3. $L((H_1 \cup H_2)) = L(H_1) \cup L(H_2)$,
4. $L((H + (a_1, \ldots, a_n))) = \{(a_1 + b_1, \ldots, a_n + b_n)\colon (b_1, \ldots, b_n) \in L(H)\}$.

For example, for $r > 0$ the one-dimensional GHIL description

$$D_r \equiv ((0 \cup 1), (\langle 0\rangle \cup (\langle 0\rangle + 2)), (\langle 1\rangle \cup (\langle 1\rangle + 4)), \ldots, (\langle r-2\rangle \cup (\langle r-2\rangle + 2^{r-1})))$$

describes the set $L(D_r) = L((\langle r-2\rangle \cup (\langle r-2\rangle + 2^{r-1}))) = \{0, 1, 2, 3, \ldots, 2^r - 1\}$. Since the length of D_r is of order r^2 we observe that a GHIL description of length n can describe a number of elements which is exponentially in n. As in the case of integer-expressions, this is also an upper bound as can be proved by induction.

Proposition 4. *For every* GHIL *description D,*

1. $\operatorname{card} L(D) \leqq 2^{|D|}$,
2. $|a| \leqq |D|$ *for all* $a \in L(D)$. □

In the same way as in the case of integer-expressions, the description of the elements of $L(D)$ by the GHIL description D can be represented by an acyclic directed labelled graph $G(D)$. For example, the graph $G(D_3)$ corresponding to the GHIL description $D_3 \equiv ((0 \cup 1), (\langle 0\rangle \cup (\langle 0\rangle + 2)), (\langle 1\rangle \cup (\langle 1\rangle + 4)))$ is shown in Fig. 3.

Again, we define $\varphi_D(x)$ to be that element of $L(D)$ which corresponds to the path x in the graph $G(D)$, and

$$\operatorname{mult}_D(a) = \operatorname{card}\{x\colon \varphi_D(x) = a\}$$

as the multiplicity of the description of the element a by the GHIL description D. We obtain

Proposition 5. *For every* GHIL *description D and for every $a \in L(D)$,*

$$\mathrm{mult}_D(a) \leqq 2^{|D|}. \quad \square$$

Proposition 6. *Given a* GHIL *description D and an $x \in \{0,1\}^*$, one can compute $\varphi_D(x)$ in polynomial time (in the size of D and x) where the result is "no" if x is not a path in $G(D)$.* $\square$

In [2] it has been shown that the general hierarchic input language is at least as powerful as the integer-expression language. More precisely, it has been shown that there exists a polynomial-time algorithm transforming every one-dimensional integer-expression H to a GHIL description which is in a sense equivalent to H (note that these languages originally describe different objects). In our terms this fact is expressed by the following theorem whose proof is similar to that in [2].

Theorem 1. *There exists a logarithmic-space (and hence polynomial-time) algorithm transforming every integer-expression H to a* GHIL *description D_H such that*

1. $L(D_H) = L(H)$,
2. $\mathrm{mult}_{D_H} = \mathrm{mult}_H$,
3. $|D_H| \leqq c \cdot |H| \cdot \log |H|$ *for some universal constant $c > 0$.*

Sketch of the proof. We describe a logarithmic-space algorithm constructing for every integer-expression H a GHIL description $D_H \equiv (H_0, H_1, \ldots, H_r)$ fulfilling the above conditions and in addition the following Condition 4: each of the GHIL-expressions $H_0, H_1, \ldots, H_r$ has either the form $(a_1, \ldots, a_n)$, or the form $(\langle i \rangle \cup \langle j \rangle)$ or the form $(\langle i \rangle + (a_1, \ldots, a_n))$. For every GHIL-expression F and every $k \geqq 1$, let $F(k)$ be the GHIL-expression arising from F by replacing all $\langle i \rangle$ by $\langle i+k \rangle$. The algorithm works recursively as follows:

1. If $H \equiv (a_1, \ldots, a_n)$ then $D_H \equiv ((a_1, \ldots, a_n))$.
2. Let $H \equiv (H_1 \cup H_2)$, and let $D_{H_1} \equiv (H_{10}, H_{11}, \ldots, H_{1r})$ and $D_{H_2} \equiv (H_{20}, H_{21}, \ldots, H_{2s})$ fulfill the conditions 1–4. Then

$$D_H \equiv (H_{10}, H_{11}, \ldots, H_{1r}, H_{20}(r+1), H_{21}(r+1), \ldots, H_{2s}(r+1), (\langle r \rangle \cup \langle s+r+1 \rangle)).$$

3. Let $H \equiv (H_1 + H_2)$, and let $D_{H_1} \equiv (H_{10}, H_{11}, \ldots, H_{1r})$ and $D_{H_2} \equiv (H_{20}, H_{21}, \ldots, H_{2s})$ fulfill the conditions 1–4. Then $D_H \equiv (H_{10}, H_{11}, \ldots, H_{1r}, H'_{20}, H'_{21}, \ldots, H'_{2s})$ where, for $i = 0, 1, \ldots, s$,

$$H'_{2i} = \begin{cases} (\langle r \rangle + (a_1, \ldots, a_n)) & \text{if } H_{2i} \equiv (a_1, \ldots, a_n) \\ H_{2i}(r+1) & \text{otherwise.} \end{cases}$$

The verification of this algorithm is left to the reader. The crucial point is item 3 of the algorithm. For its understanding it is important to see that adding D_{H_1} to D_{H_2} is the same as adding D_{H_1} to the atoms $(a_1, \ldots, a_n)$ of D_{H_2}. $\square$

The question of whether integer-expressions and GHIL descriptions have the same descriptional power, i.e. whether there exists a logarithmic-space algorithm transforming GHIL descriptions to equivalent integer-expressions, is still unresolved. We conjecture a negative solution.

2.3. Combinatorial Circuit Representation

In [5] combinatorial circuits have been used for the succinct representation of graphs. Informally, this works as follows. To describe a graph G with vertices $v_0, v_1, \ldots, v_{m-1}$ a combinatorial circuit C with $2k$ inputs and one output is used where $m<2^k$. The graph G has the edge (v_i, v_j) iff the output of C is 1 when the binary representations of i and j are given as inputs. In the same manner combinatorial circuits can represent finite subsets of $\mathbb{N}^n$. This will be defined now formally.

Let C be a combinatorial circuit with inputs $x_1, x_2, \ldots, x_{kn}$ $(k, n \geqq 1)$ and one output, and let $C(\alpha_1 \alpha_2 \ldots \alpha_{kn})$ be the output of C when $\alpha_1, \alpha_2, \ldots, \alpha_{kn} \in \{0,1\}$ are given to the inputs $x_1, x_2, \ldots, x_{kn}$. Denote the binary representation of length k of a number $a<2^k$ by $\mathrm{bin}_k(a)$. The finite set

$$L_k^n(C) = \{(a_1, \ldots, a_n)\colon a_1, \ldots, a_n < 2^k \text{ and } C(\mathrm{bin}_k(a_1) \ldots \mathrm{bin}_k(a_n)) = 1\}$$

is said to be represented by the combinatorial circuit C. For combinatorial circuits C not having $k \cdot n$ inputs we define $L_k^n(C) = \emptyset$.

As an example, the combinatorial circuit $C_{r,k}$ representing the set $L_k^1(C_{r,k}) = \{0, 1, 2, 3, \ldots, 2^r - 1\}$ (for $r \leqq k$) is shown in Fig. 4. Note that $C_{r,k}$ has $3k+r-2$ gates.

This example shows that combinatorial circuits with m gates can describe a number of elements which is exponentially in m. This is obviously also an upper bound. Defining $|C|$ as the number of gates of the combinatorial circuit C (including the input nodes) we obtain

Proposition 7. *For every combinatorial circuit C and every $n, k \geqq 1$,*

1. $\mathrm{card}\, L_k^n(C) \leqq 2^{|C|}$,
2. $|a| \leqq |C|$ *for all* $a \in L_k^n(C)$. □

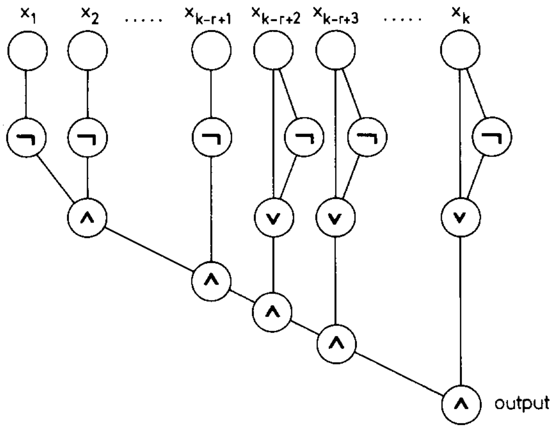

Fig. 4. The combinatorial circuit $C_{r,k}$

2.4. Boolean Expression Representation

Finally we use boolean expressions to describe finite subsets of $\mathbb{N}^n$. This works as for combinatorial circuits. The boolean expression $F(x_1, \ldots, x_{kn})$ with the boolean variables $x_1, \ldots, x_{kn}$ $(k, n \geqq 1)$ represents the finite set $L_k^n(F) \subseteq \mathbb{N}^n$ where

$$L_k^n(F) = \{(a_1, \ldots, a_n)\colon a_1, \ldots, a_n < 2^k \text{ and } F(\mathrm{bin}_k(a_1) \ldots \mathrm{bin}_k(a_n)) = 1\}.$$

For boolean expressions F not having $k \cdot n$ variables we set $L_k^n(F) = \emptyset$.

As an example take the boolean expression $F_{r,k}$ representing the set $L_k^1(F_{r,k}) = \{0, 1, 2, 3, \ldots, 2^r - 1\}$ for $r \leqq k$:

$$F_{r,k}(x_1, \ldots, x_k) \equiv \overline{x_1} \wedge \overline{x_2} \wedge \ldots \wedge \overline{x_{k-r+1}} \wedge (x_{k-r+2} \vee \overline{x_{k-r+2}}) \wedge (x_{k-r+3} \vee \overline{x_{k-r+3}}) \wedge \ldots \wedge (x_k \vee \overline{x_k}).$$

Note that $F_{r,k}$ has $2k + r - 2$ boolean operations and $k + r - 1$ occurences of variables. This examples shows that, as in the case of the other descriptional languages, a boolean expression of length m can describe a number of elements which is exponentially in m. And again this is also an upper bound. Defining $|F|$ as the number of occurences of boolean operations and boolean variables in the boolean expression F we obtain

Proposition 8. *For every boolean expression F and every $n, k \geqq 1$,*

1. $\mathrm{card}\, L_k^n(F) \leqq 2^{|F|}$,
2. $|a| \leqq |F|$ *for all* $a \in L_k^n(F)$. □

We give now a further example for constructing a boolean expression which describes a given finite set of natural numbers. We will make use of this example in Sect. 4.

Lemma 2. *There exists a logarithmic-space bounded machine computing to every input (r, k) such that $r < 2^k$ (r, k given in binary notation) a boolean expression $F(r, k)$ with variables $x_1, \ldots, x_k$ satisfying*

1. $L_k^1(F(r, k)) = \{0, 1, 2, 3, \ldots, r\}$,
2. $|F(r, k)| \leqq c \cdot \log^2 r$, *for suitable* $c > 0$.

Proof. Let $r = \sum_{i=0}^{k-1} \alpha_i \cdot 2^i$ such that $\alpha_0, a_1, \ldots, \alpha_{k-1} \in \{0, 1\}$, and let $I(r, k) = \{i\colon 0 \leqq i \leqq k - 1 \text{ and } \alpha_i = 1\}$. It is not hard to see that $F(r, k)$ can be defined as

$$F(r, k) \equiv \bigwedge_{\substack{i \notin I(r,k) \\ i \in \{0, 1, \ldots, k-1\}}} (\overline{x_{k-i}} \vee \bigvee_{\substack{j \in I(r,k) \\ j > i}} \overline{x_{k-j}})$$

to fulfill the lemma. □

It is obvious that combinatorial circuits to describe finite subsets of $\mathbb{N}^n$ are at least as powerful as boolean expressions because every boolean expression can be transformed in a canonical way to an equivalent combinatorial circuit.

Theorem 2. *There exists a logarithmic-space algorithm transforming every boolean expression F to a combinatorial circuit C_F such that*

1. $L_k^n(C_F) = L_k^n(F)$, for all $k, n \geqq 1$,
2. $|C_F| \leqq |F|$ *for some universal constant* $c > 0$. □

Remark. As an example take $C_{F_{r,k}} = C_{r,k}$.

Finally we mention that Theorem 2 cannot be conversed, i.e. that there does not exist a logarithmic-space algorithm transforming combinatorial circuits to equivalent boolean expressions unless $\mathscr{P} = \mathscr{L}$. This is an immediate consequence of Theorem 9.

2.5. General Remarks on Our Descriptional Languages

The descriptional languages defined above can be subdivided into two groups:

Group A: integer-expressions, GHIL descriptions.

Group B: boolean expressions, combinatorial circuits.

In the preceding subsections we have not compared languages from group A with languages from group B. We will see in Sect. 4 that this is in a sense not possible, i.e. that there do not exist polynomial-time compilers between languages from group A to languages from group B and vice versa, unless $\mathscr{P} = \mathscr{N}\mathscr{P}$. This is a consequence of the fact that

1. the membership problem is $\mathscr{N}\mathscr{P}$-complete for the languages of group A and is $\mathscr{P}$-complete for the languages of group B (see Theorem 9),

2. the non-emptiness problem is solvable in polynomial time for the languages of group A and is $\mathscr{N}\mathscr{P}$-complete for the languages of group B (see Theorem 10).

The main difference between these two groups of languages is the following: the descriptions of the languages of group A *generate* the described elements whereas the descriptions of the languages of group B *accept* the elements described by them. In a sense there is a duality between these groups of languages. If we define the set of possible paths of $G(H)$ to be the set described by the integer-expression (GHIL description) H then the integer-expressions (GHIL descriptions) become a language of group B. On the other hand, if we take a combinatorial circuit with several outputs (a k-tuple of boolean expressions) C and if we define the set of all numbers whose binary description can occur as an output of C to be the set described by C, then the combinatorial circuits (boolean expressions) become a language of group A. For a similar discussion of the duality between generating and accepting see [18].

Finally a remark on how many finite sets can be described succinctly by our languages. We have seen that sets of 2^n elements can have descriptions of length n^k (for suitable $k > 0$) in our languages. By a simple counting argument we can see that only a few sets can have such short descriptions. Take all subsets of $\{0, 1, 2, 3, \ldots, 2^{n+1} - 1\}$ which have at least 2^n elements. There are at least 2^{2^n} such subsets. But for the case of integer-expressions, for example, there are at most 6^{n^k} descriptions of a length not exceeding n^k. Consequently, for growing n, the rate of subsets with at least 2^n elements having a short description tends to zero.

3. The Counting Polynomial-Time Hierarchy

In order to be able to classify the complexities of the problems we are interested in we use the notion of completeness in the classes of the polynomial-time hierarchy introduced and investigated in [20] and [21]. However, for some problems in which counting is involved the classes of the polynomial-time hierarchy are not adequate. For this reason we introduce in this section the counting polynomial-time hierarchy which is an extension of the polynomial-time hierarchy. This extension is caused by adding the "counting quantifier" to the existential and universal quantifiers.

The counting quantifier C is defined as follows. For every formula $H(x, y)$ with the free variables x and y (which can be n-tuples),

$$\overset{k}{\underset{y}{\mathsf{C}}}\, H(x, y) \leftrightarrow \operatorname{card}\{y\colon H(x, y) \text{ is true}\} \geqq k.$$

The polynomially bounded versions of the existential, universal and counting quantifiers give rise to the operators V, $\mathsf{\Lambda}$ and C, resp., which are defined as follows. Let $\mathscr{K}$ be a class of languages.

$A \in \mathsf{V}\mathscr{K}$ iff there exist a $B \in \mathscr{K}$ and a polynomial p such that

$$x \in A \leftrightarrow \bigvee_{\substack{y \\ |y| \leqq p(|x|)}} (x, y) \in B.$$

$A \in \mathsf{\Lambda}\mathscr{K}$ iff there exist a $B \in \mathscr{K}$ and a polynomial p such that

$$x \in A \leftrightarrow \bigwedge_{\substack{y \\ |y| \leqq p(|x|)}} (x, y) \in B.$$

$A \in \mathsf{C}\mathscr{K}$ iff there exist a $B \in \mathscr{K}$, a polynomial-time computable function f and a polynomial p such that

$$x \in A \leftrightarrow \overset{f(x)}{\underset{\substack{y \\ |y| \leqq p(|x|)}}{\mathsf{C}}} (x, y) \in B.$$

Because of

$$\bigvee_{\substack{y \\ |y| \leqq p(|x|)}} (x, y) \in B \leftrightarrow \overset{1}{\underset{\substack{y \\ |y| \leqq p(|x|)}}{\mathsf{C}}} (x, y) \in B$$

and

$$\bigwedge_{\substack{y \\ |y| \leqq p(|x|)}} (x, y) \in B \leftrightarrow \overset{N(x)}{\underset{\substack{y \\ |y| \leqq p(|x|)}}{\mathsf{C}}} (x, y) \in B$$

($N(x)$ being the number of all y such that $|y| \leqq p(|x|)$) the operator C is at least as powerful as the operators V and $\mathsf{\Lambda}$. More formally,

Proposition 9. *For every class $\mathscr{K}$ of languages, $\mathsf{V}\mathscr{K} \cup \mathsf{\Lambda}\mathscr{K} \subseteq \mathsf{C}\mathscr{K}$.* □

Now let $\mathscr{P}$ and $\mathscr{NP}$ be the classes of languages accepted by deterministic and nondeterministic, resp., Turing machines in polynomial time.

Definition. The polynomial-time hierarchy is the smallest family $\mathscr{F}$ of classes of languages satisfying

a) $\mathscr{P} \in \mathscr{F}$,

b) if $\mathscr{K} \in \mathscr{F}$ then $\mathsf{V}\mathscr{K}$, $\mathsf{\Lambda}\mathscr{K} \in \mathscr{F}$.

Definition. The counting polynomial-time hierarchy is the smallest family $\mathscr{F}$ of classes of languages satisfying

a) $\mathscr{P} \in \mathscr{F}$,

b) if $\mathscr{K} \in \mathscr{F}$ then $\mathsf{V}\mathscr{K}$, $\mathsf{\Lambda}\mathscr{K}$, $\mathsf{C}\mathscr{K} \in \mathscr{F}$.

Furthermore, define
$\mathscr{PH} = \bigcup \{\mathscr{K} : \mathscr{K}$ is a class of the polynomial-time hierarchy$\}$,
$\mathscr{CPH} = \bigcup \{\mathscr{K} : \mathscr{K}$ is a class of the counting polynomial-time hierarchy$\}$,
$\Sigma_0^p = \Pi_0^p = \mathscr{P}$,
$\Sigma_{k+1}^p = \mathsf{V}\, \Pi_k^p$, for all $k \geqq 0$, and
$\Pi_{k+1}^p = \mathsf{\Lambda}\, \Sigma_k^p$, for all $k \geqq 0$.

The following theorem concerning the polynomial-time hierarchy can be found in [20] and [21]. For every class $\mathscr{K}$ of languages we define $\text{co}\mathscr{K} = \{\bar{A} : A \in \mathscr{K}\}$, and we denote the boolean closure of $\mathscr{K}$ by $B(\mathscr{K})$.

Theorem 3. 1. *All languages in $\mathscr{PH}$ can be accepted in polynomial space.*

2. $\mathscr{NP} = \Sigma_1^p$ *and* $\text{co}\mathscr{NP} = \Pi_1^p$.

3. $\mathsf{V}\,\Sigma_k^p = \Sigma_k^p$ *and* $\mathsf{\Lambda}\,\Pi_k^p = \Pi_k^p$ *for all* $k \geqq 1$. *Consequently, the polynomial-time hierarchy consists only of the class* $\Sigma_0^p = \Pi_0^p$, Σ_1^p, Π_1^p, Σ_2^p, Π_2^p, Σ_3^p, $\Pi_3^p, \ldots$.

4. $\text{co}\Sigma_k^p = \Pi_k^p$ *for all* $k \geqq 0$.

5. $\Sigma_k^p \cup \Pi_k^p \subseteq B(\Sigma_k^p) = B(\Pi_k^p) \subseteq \Sigma_{k+1}^p \cap \Pi_{k+1}^p$, *for all* $k \geqq 0$.

6. *Every class of the polynomial-time hierarchy is closed under* $\leqq_m^p$ *(polynomial-time bounded m-reducibility).* □

By similar arguments one can prove the following theorem on the counting polynomial-time hierarchy.

Theorem 4. 1. *All languages in $\mathscr{CPH}$ can be accepted in polynomial space.*

2. $\mathscr{PH} \subseteq \mathscr{CPH}$.

3. $\mathsf{V}\mathsf{V}\mathscr{K} = \mathsf{V}\mathscr{K}$ *and* $\mathsf{\Lambda}\mathsf{\Lambda}\mathscr{K} = \mathsf{\Lambda}\mathscr{K}$ *for all classes* $\mathscr{K}$ *of the counting polynomial-time hierarchy.*

4. $\text{co}\,\mathsf{V}\mathscr{K} = \mathsf{\Lambda}\,\text{co}\mathscr{K}$, $\text{co}\,\mathsf{\Lambda}\mathscr{K} = \mathsf{V}\,\text{co}\mathscr{K}$ *and* $\text{co}\,\mathsf{C}\mathscr{K} = \mathsf{C}\,\text{co}\mathscr{K}$ (see [19]) *for all classes* $\mathscr{K}$ *of the counting polynomial-time hierarchy.*

5. $\mathsf{C}\Sigma_k^p = \mathsf{C}\Pi_k^p = \mathsf{C}B(\Sigma_k^p) = \text{co}\,\mathsf{C}\Sigma_k^p \subseteq \mathsf{V}\mathsf{C}\Sigma_k^p \cap \mathsf{\Lambda}\mathsf{C}\Sigma_k^p$, *for all* $k \geqq 0$.

6. *Every class of the counting polynomial-time hierarchy is closed under* $\leqq_m^p$. □

It is interesting to observe the contrast between the known properties of the classes $\mathscr{NP}$ and $\mathsf{C}\mathscr{P}$. Whereas $\mathscr{NP}$ is closed under union and intersection and is not known to be closed under complement, the class $\mathsf{C}\mathscr{P}$ is closed under complement and is not known to be closed under union and intersection.

Note that it is not known whether the inclusions in the statements 5 of Theorem 3 and Theorem 4 are proper. Figure 5 shows the known inclusional relationships between the lowest classes of the counting polynomial-time hierarchy.

The classes of the polynomial-time hierarchy and some classes of the counting polynomial-time hierarchy can be characterized in terms of complexity classes of oracle Turing machines. For every class $\mathscr{K}$ of languages we define

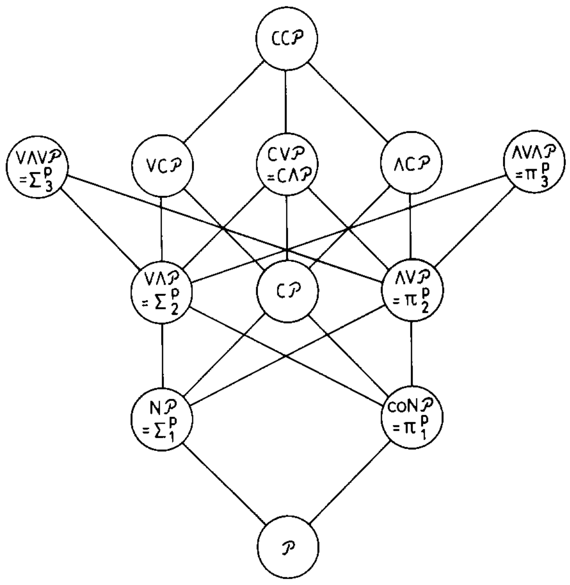

Fig. 5. The lowest classes of the counting polynomial-time hierarchy

$A \in \mathcal{NP}(\mathcal{K})$ iff there exist a $B \in \mathcal{K}$, a nondeterministic oracle Turing machine M and a polynomial p such that: $x \in A \leftrightarrow M$ with oracle B accepts x on at least one computation within $p(|x|)$ steps.

Obviously, $\mathcal{NP} = \mathcal{NP}(\emptyset) = \mathcal{NP}(\mathcal{P})$.

In [19] the class $m\mathcal{NP}$ of all languages acceptable in polynomial-time by counting Turing machines has been introduced. We generalize for every class $\mathcal{K}$ of languages:

$A \in m\mathcal{NP}(\mathcal{K})$ iff there exist a $B \in \mathcal{K}$, a polynomial-time computable function f, a nondeterministic oracle Turing machine M and a polynomial p such that: $x \in A \leftrightarrow M$ with oracle B accepts x on at least $f(x)$ computations within $p(|x|)$ steps.

Note that $m\mathcal{NP} = m\mathcal{NP}(\emptyset) = m\mathcal{NP}(\mathcal{P})$.

The following theorems show some relationships between the operators $\mathcal{NP}(\ )$ and $m\mathcal{NP}(\ )$ and our hierarchies.

Theorem 5 [20, 21]. $\Sigma_{k+1}^p = \mathcal{NP}(\Sigma_k^p) = \mathcal{NP}(\Pi_k^p)$, *for all* $k \geq 0$. □

In a similar manner one can show

Theorem 6. $\mathrm{C}\Sigma_k^p = m\mathcal{NP}(\Sigma_k^p) = m\mathcal{NP}(\Pi_k^p)$, *for all* $k \geq 0$. □

Moreover, in [19] it has been proved that the class $m\mathcal{NP}$ coincides with the class $\mathcal{PP}$ of all languages accepted in polynomial time by probabilistic Turing machines. The class $\mathcal{PP}$ has been introduced in [3].

Besides the obvious relationships $\mathcal{NP} \subseteq \mathrm{C}\mathcal{P}$ and co$\mathcal{NP} \subseteq \mathrm{C}\mathcal{P}$ there is no known relationship between the class $\mathrm{C}\mathcal{P} = m\mathcal{NP} = \mathcal{PP}$ and the classes of the polynomial-time hierarchy (see also Fig. 5). One would be interested in relationships like $\mathrm{C}\mathcal{P} \subseteq \Sigma_k^p$ for some $k \geqq 2$ or $\mathrm{C}\mathcal{P} \nsubseteq \mathcal{PH}$. It is conjectured that the second statement is true. However, there have been made several attempts to compare the class $\mathrm{C}\mathcal{P}$ or the related class of functions $\#\mathcal{P}$ (which is defined in [23]) with the classes of the polynomial-time hierarchy. In [22] and [7] it is proved that the subclass $\mathcal{BPP}$ of $\mathcal{PP}$, which consists of all languages accepted by polynomial-time probabilistic Turing machines with bounded error probability, is contained in $\Sigma_2^p \cap \Pi_2^p$. In [1] it has been shown that there exists an oracle A such that the A-relativization of $\mathrm{C}\mathcal{P}$ is not contained in the A-relativization of $\Sigma_2^p \cap \Pi_2^p$. For further results concerning the relationships between $\mathrm{C}\mathcal{P} = m\mathcal{NP} = \mathcal{PP}$ and the polynomial-time hierarchy see [17, 22, 30] and [32]. For polynomial-time hierarchies based on subclasses of $\mathrm{C}\mathcal{P}$ see [31].

In [20] complete problems for the classes Σ_k^p have been exhibited which are based on the satisfyability problem for the propositional calculus (see also [21] and [28]). In the same manner we can construct complete sets for all classes of the counting polynomial-time hierarchy. For $k \geqq 1$, $Q_1, \ldots, Q_{k-1} \in \{\mathrm{V}, \Lambda, \mathrm{C}\}$ and $Q_k \in \{\mathrm{V}, \mathrm{C}\}$ we define

$(F(\tilde{x}_1, \ldots, \tilde{x}_k), m_1, \ldots, m_k) \in Q_1 \ldots Q_k B_{be} \leftrightarrow$ *F is a boolean expression in conjunctive normal form and* $m_1, \ldots, m_k \in \mathbb{N}$ such that

$$\overset{m_1}{\underset{\tilde{\alpha}_1}{Q_1}} \ldots \overset{m_k}{\underset{\tilde{\alpha}_k}{Q}} F(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_k) = 1,$$

$$(F(\tilde{x}_1, \ldots, \tilde{x}_k), m_1, \ldots, m_k) \in Q_1 \ldots Q_{k-1} \Lambda B_{be}$$
$$\leftrightarrow (F(\tilde{x}_1, \ldots, \tilde{x}_k), m_1, \ldots, m_k) \notin \overline{Q_1} \ldots \overline{Q_{k-1}} \mathrm{V} B_{be}$$

where $\bar{\mathrm{V}} = \Lambda$, $\bar{\Lambda} = \mathrm{V}$ and $\bar{\mathrm{C}} = \mathrm{C}$. Furthermore, the $\tilde{x}_i$ can be sets of variables, and the numbers over the quantifiers V and Λ are to omit, i.e. $\overset{m}{\mathrm{V}} = \mathrm{V}$ and $\overset{m}{\Lambda} = \Lambda$.

The following theorem generalizes the corresponding result for the classes of the polynomial-time hierarchy whose proof (see [21] and [28]) carries over to the general case. Let $\leqq_m^{\log}$ denote the logarithmic-space m-reducibility.

Theorem 7. *For every $k \geqq 1$ and every $Q_1, \ldots, Q_k \in \{\mathrm{V}, \Lambda, \mathrm{C}\}$,*

$$Q_1 \ldots Q_k B_{be} \textit{ is } \leqq_m^{\log}\textit{-complete in } Q_1 \ldots Q_k \mathcal{P}. \quad \square$$

Note that the problem m-SATIFYABILITY which is identically with our $\mathrm{C}B_{be}$ as well as 20 other problems have been shown to be $m\mathcal{NP}$-complete in [19].

Now we exhibit another type of complete sets for the classes of the counting polynomial-time hierarchy. These complete sets are based on integer-expressions rather than on boolean expressions. For $k \geqq 0$ and $Q_1, \ldots, Q_k \in \{\mathrm{V}, \Lambda, \mathrm{C}\}$ we define

$(H,b,m_1,\dots,m_k,d)\in Q_1\dots Q_k\mathsf{V}B_{ie}\leftrightarrow$ H is a $(k+1)$-dimensional integer-expression and $b,m_1,\dots,m_k,\ d\in\mathbb{N}$ such that

$$\underset{a_1\leqq d}{\overset{m_1}{\mathsf{Q}_1}}\dots\underset{a_k\leqq d}{\overset{m_k}{\mathsf{Q}_k}}\,(a_1,\dots,a_k,b)\in L(H),$$

$$(H,b,m_1,\dots,m_k,d)\in Q_1\dots Q_k\mathsf{\Lambda}B_{ie}\leftrightarrow(H,b,m_1,\dots,m_k,d)\notin\overline{Q_1}\dots\overline{Q_k}\mathsf{V}B_{ie}$$

where $\bar{\mathsf{V}}=\mathsf{\Lambda}$, $\bar{\mathsf{\Lambda}}=\mathsf{V}$ and $\bar{\mathsf{C}}=\mathsf{C}$,

$(H,b,m_1,\dots,m_k,m,d)\in Q_1\dots Q_k\mathsf{C}B_{ie}\leftrightarrow$ H is a $(k+1)$-dimensional integer-expression and $b,m_1,\dots,m_k,\ m,\ d\in\mathbb{N}$ such that

$$\underset{a_1\leqq d}{\overset{m_1}{\mathsf{Q}_1}}\dots\underset{a_k\leqq d}{\overset{m_k}{\mathsf{Q}_k}}\,\mathrm{mult}_H(a_1,\dots,a_k,b)\geqq m.$$

Theorem 8. *For every $k\geqq 1$ and every $Q_1,\dots,Q_k\in\{\mathsf{V},\mathsf{\Lambda},\mathsf{C}\}$,*

$$Q_1\dots Q_kB_{ie}\ \textit{is}\ \leqq_m^{\log}-\textit{complete for}\ Q_1\dots Q_k\mathscr{P}.$$

Proof. To prove this theorem we describe a logarithmic-space reduction from $Q_1\dots Q_kB_{be}$ to $Q_1\dots Q_kB_{ie}$. Let $F(\tilde{x}_1,\dots,\tilde{x}_k)\equiv C_0\wedge C_1\wedge\dots\wedge C_m$ be a boolean expression in conjunctive normal form such that the C_j are disjunctions of literals (i.e. variables or their negations). Without loss of generality we assume that there exists an $n\geqq 1$ such that every $\tilde{x}_1$ consists of n boolean variables: $\tilde{x}_1=(x_{l1},\dots,x_{ln})$ for $l=1,\dots,k$. Furthermore we assume that no C_j is empty and that no C_j contains a variable and its negation.

Modifying a method in [20] we set $r=\min\{s\colon 2^s\geqq 2kn\}$, $b=2^r$, and we define

$$I(x_{li})=(\underbrace{0,0,\dots,0}_{l-1},2^{i-1},\underbrace{0,0,\dots,0}_{k-l-1},\sum_{x_{li}inC_j}b^j),\quad\text{for } l=1,\dots,k-1;\ i=1,\dots,n,$$

$$I(x_{ki})=(\underbrace{0,0,\dots,0}_{k-1},\sum_{x_{ki}inC_j}b^j),\quad\text{for } i=1,\dots,n,$$

$$I(\overline{x_{li}})=(\underbrace{0,0,\dots,0}_{k-1},\sum_{\overline{x_{li}}inC_j}b^j),\quad\text{for } l=1,\dots,k;\ i=1,\dots,n,$$

and

$$H_F=\sum_{l=1}^{k}\sum_{i=1}^{n}(I(x_{li})\cup I(\overline{x_{li}})).$$

An assignment $\alpha_{11},\dots,\alpha_{kn}\in\{0,1\}$ to the variables $x_{11},\dots,x_{kn}$ of F can be interpreted as a path of the generation of elements by H_F in the following manner:

$$\alpha_{li}=1\text{ means: choose }I(x_{li})\text{ in }(I(x_{li})\cup I(\bar{x}_{li})),$$

$$\alpha_{li}=0\text{ means: choose }I(\bar{x}_{li})\text{ in }(I(x_{li})\cup I(\bar{x}_{li})).$$

Thus an assignment $\alpha_{11}, \dots, \alpha_{kn}$ leads to the element

$$\left(\sum_{i=1}^{n} \alpha_{1i} \cdot 2^{i-1}, \sum_{i=1}^{n} \alpha_{2i} \cdot 2^{i-1}, \dots, \sum_{i=1}^{n} \alpha_{k-1,i} \cdot 2^{i-1}, \sum_{j=0}^{m} \beta_j \cdot b^j\right)$$

of $L(H_F)$ where β_j tells us how many literals in C_j become true under this assignment. Consequently, an assignment $\alpha_{11}, \dots, \alpha_{kn}$ to the variables $x_{11}, \dots, x_{kn}$ satisfies F iff $\beta_j \geqq 1$ for all $j = 0, 1, \dots, m$. According to the pattern of H_{r-1} defined in Subsection 2.1 one can define the k-dimensional integer-expression $H_{r-1}^{k,j}$ in such a way that $L(H_{r-1}^{k,j}) = \{(\underbrace{0, \dots, 0}_{k-1}, t \cdot b^j)\colon t = 0, 1, \dots, 2^{r-1} - 1\}$ and that no multiplicity of $H_{r-1}^{k,j}$ is greater than one. Defining $H'_F \equiv H_F + \sum_{j=0}^{m} H_{r-1}^{k,j}$ we obtain: the assignment $\alpha_{11}, \dots, \alpha_{kn}$ to the variables $x_{11}, \dots, x_{kn}$ satisfies F iff

$$\left(\sum_{i=1}^{n} \alpha_{1i} \cdot 2^{i-1}, \sum_{i=1}^{n} \alpha_{2i} \cdot 2^{i-1}, \dots, \sum_{i=1}^{n} \alpha_{k-1,i} \cdot 2^{i-1}, \sum_{j=0}^{m} 2^{r-1} \cdot b^j\right) \in L(H'_F).$$

Since different $\tilde{\alpha}_l = (\alpha_{l1}, \dots, \alpha_{ln})$ yield different $\sum_{i=1}^{n} \alpha_{li} \cdot 2^{i-1}$ we conclude

$$\overset{m_1}{\underset{\tilde{\alpha}_1}{\mathsf{C}}} \dots \overset{m_k}{\underset{\tilde{\alpha}_k}{\mathsf{C}}} F(\tilde{\alpha}_1, \dots, \tilde{\alpha}_k) = 1 \leftrightarrow \overset{m_1}{\underset{a_1 < 2^n}{\mathsf{C}}} \dots \overset{m_{k-1}}{\underset{a_{k-1} < 2^n}{\mathsf{C}}} \operatorname{mult}_{H'_F}(a_1, \dots, a_{k-1}, c) \geqq m_k$$

where $c = \sum_{j=0}^{m} 2^{r-1} \cdot b^j$.

Now let $Q_1, \dots, Q_{k-1} \in \{\mathsf{V}, \mathsf{\Lambda}, \mathsf{C}\}$. We obtain

$$\begin{aligned}
&(F(\tilde{x}_1, \dots, \tilde{x}_k), m_1, \dots, m_k) \in Q_1 \dots Q_{k-1} \mathsf{V} B_{be} \\
&\leftrightarrow \overset{m_1}{\underset{\tilde{\alpha}_1}{\mathsf{Q}_1}} \dots \overset{m_{k-1}}{\underset{\tilde{\alpha}_{k-1}}{\mathsf{Q}_{k-1}}} \underset{\tilde{\alpha}_k}{\mathsf{V}} F(\tilde{\alpha}_1, \dots, \tilde{\alpha}_{k-1}, \tilde{\alpha}_k) = 1 \\
&\leftrightarrow \overset{m'_1}{\underset{\tilde{\alpha}_1}{\mathsf{C}}} \dots \overset{m'_{k-1}}{\underset{\tilde{\alpha}_{k-1}}{\mathsf{C}}} \overset{1}{\underset{\alpha_k}{\mathsf{C}}} F(\tilde{\alpha}_1, \dots, \tilde{\alpha}_{k-1}, \tilde{\alpha}_k) = 1 \quad \text{where } m'_l = \begin{cases} 1 & \text{if } Q_l = \mathsf{V} \\ 2^n & \text{if } Q_l = \mathsf{\Lambda} \\ m_l & \text{if } Q_l = \mathsf{C} \end{cases} \\
&\leftrightarrow \overset{m'_1}{\underset{a_1 < 2^n}{\mathsf{C}}} \dots \overset{m'_{k-1}}{\underset{a_{k-1} < 2^n}{\mathsf{C}}} \operatorname{mult}_{H'_F}(a_1, \dots, a_{k-1}, c) \geqq 1 \\
&\leftrightarrow \overset{m_1}{\underset{a_1 < 2^n}{\mathsf{Q}_1}} \dots \overset{m_{k-1}}{\underset{a_{k-1} < 2^n}{\mathsf{Q}_{k-1}}} \operatorname{mult}_{H'_F}(a_1, \dots, a_{k-1}, c) \geqq 1 \\
&\leftrightarrow \overset{m_1}{\underset{a_1 < 2^n}{\mathsf{Q}_1}} \dots \overset{m_{k-1}}{\underset{a_{k-1} < 2^n}{\mathsf{Q}_{k-1}}} (a_1, \dots, a_{k-1}, c) \in L(H'_F) \\
&\leftrightarrow (H'_F, c, m_1, \dots, m_{k-1}, 2^n - 1) \in Q_1 \dots Q_{k-1} \mathsf{V} B_{ie}.
\end{aligned}$$

Consequently, $Q_1, \dots, Q_{k-1} \mathsf{V} B_{be} \leqq_m^{\log} Q_1 \dots Q_{k-1} \mathsf{V} B_{ie}$. In the same manner we obtain

$$\begin{aligned}
&(F(\tilde{x}_1, \dots, \tilde{x}_k), m_1, \dots, m_k) \in Q_1 \dots Q_{k-1} \mathsf{C} B_{be} \\
&\quad \leftrightarrow (H'_F, c, m_1, \dots, m_k, 2^n - 1) \in Q_1 \dots Q_{k-1} \mathsf{C} B_{ie},
\end{aligned}$$

and consequently $Q_1 \dots Q_{k-1} \mathsf{C} B_{be} \leq_m^{\log} Q_1 \dots Q_{k-1} \mathsf{C} B_{ie}$. Finally, because of $\overline{Q_1} \dots \overline{Q_{k-1}} \vee B_{be} \leq_m^{\log} \overline{Q_1} \dots \overline{Q_{k-1}} \vee B_{ie}$ we have also

$$Q_1 \dots Q_{k-1} \wedge B_{be} \leq_m^{\log} \overline{\overline{Q_1} \dots \overline{Q_{k-1}} \vee B_{be}} \leq_m^{\log} \overline{\overline{Q_1} \dots \overline{Q_{k-1}} \vee B_{ie}}$$
$$\leq_m^{\log} Q_1 \dots Q_{k-1} \wedge B_{ie}. \quad \square$$

At the end of this section we notice that instead of investigating the operator C we could have investigated with the same right the operator $\mathsf{C}_=$ based on the polynomially bounded version of the quantifier $\mathsf{C}_=$ which is defined as follows: $\overset{k}{\underset{y}{\mathsf{C}_=}} H(x,y)$ iff $\operatorname{card}\{y\colon H(x,y) \text{ is true}\} = k$. In contrast to $\mathsf{C}\mathcal{P}$, the class $\mathsf{C}_=\mathcal{P}$ is closed under intersection and it is not known to be closed under union and complement. It is not hard to see that $\mathsf{C}_=\mathcal{K} \subseteq B(\mathsf{C}\mathcal{K}) \subseteq \mathsf{V}\,\mathsf{C}_=\mathcal{K}$ for all classes $\mathcal{K}$ of languages. Consequently, the class $\mathcal{CPH}$ does not depend on whether the operator C or the operator $\mathsf{C}_=$ is used. The completeness results in this section and in Sect. 4 carry over to the case that the classes are defined with $\mathsf{C}_=$ instead of C and that the problems are defined with "exactly" instead of "at least".

Finally we mention that in [26] the class $\mathcal{CPH}$ has been characterized by recursion-theoretic means in such a way that $\mathcal{CPH}$ appears as an analogue of the class $\mathcal{P}$ as well as the class of all sets which are elementary recursive in the sense of KALMAR.

4. Complexity Results

In this paper we are concerned with problems having very low complexities when the instances are given as usually by a report of the elements. More exactly, using this input representation our problems can be solved in space $\log n$ and, alternatively, in time $n \cdot \log n$ for inputs of length n. Since our descriptional languages can compress the inputs at most exponentially (see Propositions 1, 4, 7 and 8) all these problems can be solved in polynomial space when the descriptional languages are used. In fact we can show that each of the 19 problems defined below is $\leq_m^{\log}$-complete in some class of the counting polynomial-time hierarchy.

Let $A(\mathrm{IE})$, $A(\mathrm{GHIL})$, $A(\mathrm{BE})$ and $A(\mathrm{CC})$ be the problem A when its instances are given by integer-expressions, GHIL descriptions, boolean expressions and combinatorial circuits, resp. To prove that $A(X)$ is $\leq_m^{\log}$-complete in the class $\mathcal{K}$ of the counting polynomial-time hierarchy we have to show that

1. $A(X) \in \mathcal{K}$, and
2. every problem in $\mathcal{K}$ is $\leq_m^{\log}$-reducible to $A(X)$.

To this end we will use Theorem 1 which yields $A(\mathrm{IE}) \leq_m^{\log} A(\mathrm{GHIL})$ and Theorem 2 which yields $A(\mathrm{BE}) \leq_m^{\log} A(\mathrm{CC})$. Since in most cases (i.e. with the exception of the Theorems 9 and 13) the problems $A(\mathrm{IE})$ and $A(\mathrm{GHIL})$ have the same complexity, and the problems $A(\mathrm{BE})$ and $A(\mathrm{CC})$ have the same complexity, we can restrict ourselves to prove in these cases 1. for $X = \mathrm{GHIL}$ and $X = \mathrm{CC}$ and 2. for $X = \mathrm{IE}$ and $X = \mathrm{BE}$.

Though our problems are of the first glance of more theoretical nature, there are a lot of connections to practically important problems. For example, some of our problems can be considered as special cases of problems in [2] concerning finite sets of rectangles in the plane and having the same complexities as our special cases. For a more detailed discussion of the practical relevance of these problems the reader is referred to [2].

In the proofs below we will use frequently the integer-expressions H_r defined in Subsection 2.1 to fulfill $L(H_r)=\{0,1,2,\ldots,2^r-1\}$ and the integer-expressions $H(r)$ from Lemma 1 to fulfill $L(H(r))=\{0,1,2,\ldots,r\}$. Furthermore, for one-dimensional integer-expressions H we will use the one-dimensional integer-expressions $r\cdot H$ which arise from H by replacing every "atom" a in H by $r\cdot a$. Consequently, $L(r\cdot H)=\{r\cdot a\colon a\in L(H)\}$.

Now we discuss our problems. In order to be able to formulate the problems in a unified manner we use, for boolean expressions and for combinatorial circuits, instead of $L_k^n(D)$ also $L(D)$. The context will ensure that n and k are self-evident.

Membership Problem

Given: A description D of a finite set $L(D)\subseteq\mathbb{N}$, $a\in\mathbb{N}$.
Question: $a\in L(D)$?

Theorem 9. *The membership problem is*

1. *$\mathcal{NP}$-complete for integer-expressions and* GHIL *descriptions,*
2. *$\mathcal{P}$-complete for combinatorial circuits,*
3. *in $\mathcal{L}$ (the class of problems solvable in logarithmic space by deterministic machines) for boolean expressions.*

Proof. The first statement is proved in [20] for integer-expressions. That the problem is in $\mathcal{NP}$ follows from the equivalence

$$a\in L(D)\leftrightarrow \bigvee_x \varphi_D(x)=a$$

and the fact that $\varphi_D(x)$ can be computed in polynomial time. The second statement is proved in [8], the third statement is proved in [13]. □

A further basic problem is the non-emptiness problem. Here the complexities show another behaviour as for the membership problem. From this fact we can conclude that the two groups of descriptional languages (integer-expressions and GHIL descriptions on the one side, and boolean expressions and combinatorial circuits on the other side) cannot be compared efficiently with each other unless $\mathcal{P}=\mathcal{NP}$.

Non-emptiness Problem

Given: A description D of a finite set $L(D)\subseteq\mathbb{N}$.
Question: $L(D)\neq\emptyset$?

Theorem 10. *The non-emptiness problem is*

1. *in $\mathscr{L}$ for integer-expressions and* GHIL *descriptions,*
2. *$\mathscr{NP}$-complete for boolean expressions and combinatorial circuits.*

Proof. Statement 1 is obvious since every nonempty integer-expression or GHIL description describes a nonempty set. For the second statement we observe that the non-emptiness problem for boolean expressions is identically with the satisfyability problem for the propositional calculus which is known to be $\mathscr{NP}$-complete (see [4]). Thus it remains to show that the problem is in $\mathscr{NP}$ for combinatorial circuits. This follows from

$$L(D) \neq \emptyset \leftrightarrow \bigvee_x x \in L(D)$$

and Theorem 9.2. $\square$

The next problem is in a sense a combination of the membership problem and the non-emptiness problem.

Critical Element Problem

Given: A description D of a finite set $L(D) \subseteq \mathbb{N}$, $a \in \mathbb{N}$.
Question: Does $a \notin L(D)$ imply $L(D) = \emptyset$?

Theorem 11. *The critical element problem is*

1. *$\mathscr{NP}$-complete for integer-expressions and* GHIL *descriptions,*
2. co*$\mathscr{NP}$-complete for boolean expressions and combinatorial circuits.*

Proof. Statement 1. The problem is in $\mathscr{NP}$ because of

$$(a \notin L(D) \to L(D) = 0) \leftrightarrow \bigvee_x (\varphi_D(x) = a \vee L(D) = \emptyset).$$

On the other hand, the $\mathscr{NP}$-complete membership problem (see Theorem 9.1) is log-space reducible to the critical element problem by

$$a \in L(D) \leftrightarrow ((a \notin L(D \cup (a+1)) \to L(D \cup (a+1)) = \emptyset)).$$

Statement 2. The problem is in co$\mathscr{NP}$ because of

$$(a \notin L(D) \to L(D) = \emptyset) \leftrightarrow \bigwedge_x (a \notin L(D) \to x \notin L(D)).$$

Furthermore, the emptiness problem (which is co$\mathscr{NP}$-complete because of Theorem 10.2) is log-space reducible to the critical element problem. This can be seen as follows. Let F be a boolean expression with k variables. Then no element of $L(F)$ can be greater than $2^k - 1$. Consequently

$$L(F) = \emptyset \leftrightarrow (2^k \notin L(F) \to L(F) = \emptyset). \quad \square$$

Intersection Problem

Given: Descriptions D_1 and D_2 of finite sets $L(D_1)$, $L(D_2) \subseteq \mathbb{N}$.
Question: $L(D_1) \cap L(D_2) \neq \emptyset$?

Theorem 12. *The intersection problem is*

1. *$\mathcal{NP}$-complete for integer-expressions and* GHIL *descriptions.*
2. *$\mathcal{NP}$-complete for boolean expressions and combinatorial circuits.*

Proof. Statement 1. The problem is in $\mathcal{NP}$ because of

$$L(D_1) \cap L(D_2) \neq \emptyset \leftrightarrow \bigvee_x \bigvee_y \varphi_{D_1}(x) = \varphi_{D_2}(y).$$

On the other hand, the membership problem can be reduced to the intersection problem by

$$a \in L(D) \leftrightarrow L(D) \cap L(a) \neq \emptyset.$$

Statement 2. The problem is in $\mathcal{NP}$ because of

$$L(D_1) \cap L(D_2) \neq \emptyset \leftrightarrow \bigvee_x (x \in L(D_1) \wedge x \in L(D_2)).$$

On the other hand, the $\mathcal{NP}$-complete non-emptiness problem (see Theorem 10.2) can be reduced to the intersection problem as follows. Let F be a boolean expression with k variables. Then no element of $L(F)$ can be greater than $2^k - 1$. Let $F_{k,k}$ be the boolean expression constructed in Subsection 2.4 such that $L(F_{k,k}) = \{0, 1, 2, 3, \ldots, 2^k - 1\}$. Then

$$L(F) \neq \emptyset \leftrightarrow L(F) \cap L(F_{k,k}) \neq \emptyset. \quad \square$$

Note that our intersection problem for GHIL descriptions is a special case of the intersection problem in [2] which is shown to be $\mathcal{NP}$-complete there.

Isolated Element Problem

Given: A description D of a finite set $L(D) \subseteq \mathbb{N}$, $a \in \mathbb{N}$.
Question: Is a an isolated element of $L(D)$, i.e. $a \in L(D)$, $a + 1 \notin L(D)$ and $(a > 0 \rightarrow a - 1 \notin L(D))$?

The next theorem uses the notion of completeness in the class $D^P = \{A \cap B: A \in \mathcal{NP} \wedge B \in \mathrm{co}\mathcal{NP}\}$. It is obvious that $\mathcal{NP} \cup \mathrm{co}\mathcal{NP} \subseteq D^P \subseteq B(\mathcal{NP})$, and from Theorem 3.5 we know $B(\mathcal{NP}) \subseteq \Sigma_2^p \cap \Pi_2^p$. Complete problems in D^P have already been found in [16] and [15].

Theorem 13. *The isolated element problem is*

1. *D^P-complete for integer-expressions and* GHIL *descriptions,*
2. *$\mathcal{P}$-complete for combinatorial circuits,*
3. *in $\mathcal{L}$ for boolean expressions.*

Proof. Statement 1. The problem is in D^P because of the equivalence:
a is an isolated element of $L(D) \leftrightarrow$

$$\leftrightarrow \bigvee_x \varphi_D(x) = a \wedge \bigwedge_x (\varphi_D(x) \neq a + 1 \wedge (a > 0 \rightarrow \varphi_D(x) \neq a - 1)).$$

On the other hand, an arbitrary set $C \in D^P$ can be reduced to the isolated element problem as follows. Let $C = A \cap B$ where $A \in \mathcal{NP}$ and $\bar{B} \in \mathcal{NP}$. By Theorem 9.1 there are log-space computable functions f and g reducing A and $\bar{B}$, resp., to the membership problem. Let $f(x) = (H, a)$ and $g(x) = (H', a')$. We conclude

$$\begin{aligned} x \in C = A \cap B &\leftrightarrow a \in L(H) \text{ and } a' \notin L(H') \\ &\leftrightarrow 3aa' \in L(3a'H) \text{ and } (3aa'+1) \notin L(3aH'+1) \\ &\leftrightarrow 3aa' \in L(3a'H \cup (3aH'+1)) \text{ and } (3aa'+1) \notin L(3a'H \cup (3aH'+1)) \\ &\leftrightarrow 3aa' \text{ is an isolated element in } L(3a'H \cup (3aH'+1)). \end{aligned}$$

Statement 2. It is obvious that the problem belongs to P. On the other hand, the membership problem (which is $\mathcal{P}$-complete because of Theorem 9.2) can be reduced to the isolated element problem by the following equivalence for boolean expressions F:

$$x \in L(F(x_1, \ldots, x_k)) \leftrightarrow 2x \text{ is an isolated element of } L(F(x_1, \ldots, x_k) \wedge \overline{x_{k+1}}).$$

Statement 3 follows immediately from Theorem 9.3. $\square$

Maximum Element Below a Boundary Problem

Given: A description D of a finite set $L(D) \subseteq \mathbb{N}$, $a, b \in \mathbb{N}$.
Question: Is a the maximum element in $L(D)$ below b, i.e. $a \leqq b$, $a \in L(D)$ and $c \notin L(D)$ for all $c \in \{a+1, a+2, \ldots, b\}$?

Theorem 14. *The maximum element below a boundary problem is*
1. *D^P-complete for integer-expressions and* GHIL *descriptions,*
2. co$\mathcal{NP}$*-complete for boolean expressions and combinatorial circuits.*

Proof. Statement 1. The problem is in D^P because of the equivalence:
a is the maximum element in $L(D)$ below $b \leftrightarrow$

$$\leftrightarrow a \leqq b \wedge \bigvee_x \varphi_D(x) = a \wedge \bigwedge_x \varphi_D(x) \notin \{a+1, a+2, \ldots, b\}.$$

On the other hand, for an arbitrary set $C \in D^P$ we can conclude as in the proof of Theorem 13:

$$\begin{aligned} x \in C &\leftrightarrow 3aa' \in L(3a' \cdot H \cup (3aH'+1)) \text{ and } (3aa'+1) \notin L(3a'H \cup (3aH'+1)) \\ &\leftrightarrow 3aa' \text{ is the maximum element in } L(3a' \cdot H \cup (3aH'+1)) \text{ below } 3aa'+1. \end{aligned}$$

Statement 2. The problem is in co$\mathcal{NP}$ because of the equivalence:
a is the maximum element in $L(D)$ below $b \leftrightarrow$

$$\leftrightarrow a \leqq b \wedge a \in L(D) \wedge \bigwedge_c (a < c \leqq b \rightarrow c \notin L(D)).$$

On the other hand, the emptiness problem (which is co$\mathcal{NP}$-complete because of Theorem 10.2) can be log-space reduced to the maximum element below a boundary problem as follows: Let F be a boolean expression with k variables. We can conclude

$$\begin{aligned} L(F)=\emptyset &\leftrightarrow 0 \text{ is the maximum element in } L((\overline{x_1}\wedge\overline{x_2}\wedge\ldots\wedge\overline{x_{k+1}})\vee F(x_2,\ldots,x_{k+1}))\\ &\leftrightarrow 0 \text{ is the maximum element in } L((\overline{x_1}\wedge\overline{x_2}\wedge\ldots\wedge\overline{x_{k+1}})\vee F(x_2,\ldots,x_{k+1}))\\ & \text{ below } 2^{k+2}. \quad \square \end{aligned}$$

Note that Theorem 14, improves a result in [2] on the northernmost rectangle below a line problem which is shown there to be $\mathcal{NP}$-hard and co$\mathcal{NP}$-hard. Our problem for GHIL descriptions can be considered as a special case of this problem. Hence, the northernmost rectangle below a line problem is D^P-hard. On the other hand, it is obviously in D^P.

Subset Problem

Given: Descriptions D_1, D_2 of finite sets $L(D_1)$, $L(D_2)\subseteq\mathbb{N}$.
Question: $L(D_1)\subseteq L(D_2)$?

Equality Problem

Given: Descriptions D_1, D_2 of finite sets $L(D_1)$, $L(D_2)\subseteq\mathbb{N}$.
Question: $L(D_1)=L(D_2)$?

Theorem 15. *The subset problem and the equality problem are*

1. *Π_2^p-complete for integer-expressions and* GHIL *descriptions,*
2. *co$\mathcal{NP}$-complete for boolean expressions and combinatorial circuits.*

Proof. Statement 1. That the problems are in Π_2^p follows from the equivalences

$$L(D_1)\subseteq L(D_2)\leftrightarrow \bigwedge_x \bigvee_y \varphi_{D_1}(x)=\varphi_{D_2}(y)$$

and

$$L(D_1)=L(D_2)\leftrightarrow L(D_1)\subseteq L(D_2)\wedge L(D_2)\subseteq L(D_1).$$

On the other hand, it is shown in [20] that the problems are Π_2^p-complete for integer-expressions.

Statement 2. That the problems are in co $\mathcal{NP}$ follows from

$$L(D_1)\subseteq L(D_2)\leftrightarrow \bigwedge_x (x\in L(D_1)\rightarrow x\in L(D_2))$$

and

$$L(D_1)=L(D_2)\leftrightarrow L(D_1)\subseteq L(D_2)\wedge L(D_2)\subseteq L(D_1).$$

On the other hand, the co$\mathcal{NP}$-complete emptiness problem (see Theorem 10.2) can be reduced to these problems by

$$L(D)=\emptyset\leftrightarrow L(D)\subseteq L(x_1\wedge\overline{x_1})\leftrightarrow L(D)=L(x_1\wedge\overline{x_1}). \quad \square$$

Note that Theorem 15.1 can already be found in [2] for GHIL descriptions of finite sets of rectangles in the plane. For this case the connectedness problem dealt with in the next theorem is shown to be $\mathcal{NP}$-hard in [2]. This is improved by Theorem 16.

Connectedness Problem

Given: A description D of a finite set $L(D) \subseteq \mathbb{N}$.
Question: Is $L(D)$ connected, i.e. $\bigwedge_x \bigwedge_y \bigwedge_z (x<y<z \wedge x, z \in L(D) \rightarrow y \in L(D))$?

Theorem 16. *The connectedness problem is*

1. *Π_2^p-complete for integer-expressions and* GHIL *descriptions,*
2. co$\mathcal{NP}$*-complete for boolean expressions and combinatorial circuits.*

Proof. Statement 1. The problem is in Π_2^p because of the equivalence:

$$L(D) \text{ is connected} \leftrightarrow \bigwedge_x \bigwedge_y \bigvee_z (\varphi_D(x)<\varphi_D(y) \rightarrow \varphi_D(z)=\varphi_D(x)+1).$$

On the other hand, the Π_2^p-complete problem $\wedge\vee B_{ie}$ (see Theorem 8) can be reduced to the connectedness problem as follows. Let H be a two-dimensional integer-expression and let $N=2^{|H|}$. By Proposition 1.2 we have $a, b<N$ for all $(a,b) \in L(H)$. Now let $\hat{H}$ be that one-dimensional integer-expression which arises from H by replacing every (a,b) by $b \cdot N+a$. Hence $(a,b) \in L(H)$ iff $b \cdot N+a \in L(\hat{H})$, and we can conclude

$$\begin{aligned}(H,b,m,d) \in \wedge\vee B_{ie} &\leftrightarrow \bigwedge_{a \leq d} (a,b) \in L(H)\\ &\leftrightarrow \{bN, bN+1, \ldots, bN+d\} \subseteq L(\hat{H})\\ &\leftrightarrow L(\hat{H} \cup H(bN-1) \cup ((bN+d+1)+H(N^2))) \text{ is connected.}\end{aligned}$$

Statement 2. That the problem is in co$\mathcal{NP}$ follows from its representation in the definition. On the other hand, the co$\mathcal{NP}$-complete emptiness problem (see Theorem 10.2) can be reduced to the connectedness problem as follows. Let F be a boolean expression with k variables. Evidently we have

$$L(F)=\emptyset \leftrightarrow L(F(x_z, \ldots, x_{k+1}) \vee (x_1 \wedge \ldots \wedge x_k \wedge x_{k+1})) \text{ is connected.} \quad \square$$

The subset A of $B \subseteq \mathbb{N}$ is said to be a component of B iff A is connected, $(\max(A)+1) \notin B$ and $(\min(A)>0 \rightarrow (\min(A)-1) \notin B)$.

Component Length Problem

Given: A description D of a finite set $L(D) \subseteq \mathbb{N}$, $m \in \mathbb{N}$.
Question: Does there exist a component of $L(D)$ having length m or more.

Theorem 17. *The component length problem is*

1. *Σ_3^p-complete for integer-expressions and* GHIL *descriptions,*
2. *Σ_2^p-complete for boolean expressions and combinatorial circuits.*

Proof. Statement 1. The problem is in Σ_3^p because of the equivalence:

$$L(D) \text{ has a component of length } m \text{ or more} \leftrightarrow \bigvee_x \bigwedge_y \bigvee_z (x \leq y < x+m \rightarrow \varphi_D(z) = y).$$

On the other hand, the Σ_3^p-complete problem $\mathsf{V}\Lambda\mathsf{V}\, B_{ie}$ (see Theorem 8) is reducible to the component length problem as follows. For a three-dimensional integer-expression H we define $N = 2^{|H|+1}$. By Proposition 1.2 we have $a, b, c < \frac{N}{2}$ for all $(a, b, c) \in L(H)$. Now let $\hat{H}$ be that one-dimensional integer-expression which arises from H by replacing every (a, b, c) by $cN^2 + aN + 2b$. Consequently, $(a, b, c) \in L(H)$ iff $cN^2 + aN + 2b \in L(\hat{H})$, and we can conclude

$$\begin{aligned}
(H, b, m_1, m_2, d) \in \mathsf{V}\Lambda\mathsf{V}\, B_{ie} &\leftrightarrow \bigvee_{a_1 \leq d} \bigwedge_{a_2 \leq d} (a_1, a_2, b) \in L(H) \\
&\leftrightarrow \bigvee_{a_1 \leq d} \bigwedge_{a_2 \leq d} bN^2 + a_1 N + 2a_2 \in L(\hat{H}) \\
&\leftrightarrow \bigvee_{a_1 \leq d} (bN^2 + a_1 N + \{0, 2, 4, \ldots, 2d\}) \subseteq L(\hat{H}) \\
&\leftrightarrow \bigvee_{a_1 \leq d} (bN^2 + a_1 N + \{0, 1, 2, \ldots, 2d\}) \subseteq L(\hat{H}) \cup \\
&\qquad \cup (bN^2 + N \cdot \{0, 1, 2, \ldots, d\} + \{1, 3, \ldots, 2d-1\}) \\
&\leftrightarrow \bigvee_{a_1 \leq d} (bN^2 + a_1 N + \{0, 1, 2, \ldots, 2d\}) \subseteq \\
&\qquad \subseteq L(\hat{H} \cup (bN^2 + N \cdot H(d) + 2H(d-1) + 1)) \\
&\leftrightarrow L(\hat{H} \cup (bN^2 + N \cdot H(d) + 2H(d-1) + 1)) \\
&\qquad \text{has a component of length } 2d+1 \text{ or more.}
\end{aligned}$$

Statement 2. The problem is in Σ_2^p because of the equivalence:

$$L(D) \text{ has a component of length } m \text{ or more} \leftrightarrow \bigvee_x \bigwedge_y (x \leq y < x+m \rightarrow y \in L(D)).$$

On the other hand, the Σ_2^p-complete problem $\mathsf{V}\Lambda B_{be}$ (see Theorem 7) is reducible to the component length problem as follows. For the boolean expression $F(x_1, \ldots, x_r, y_1, \ldots, y_s)$ we define

$$F'(x_1, \ldots, x_r, z, y_1, \ldots, y_s) \equiv F(x_1, \ldots, x_r, y_1, \ldots, y_s) \wedge z.$$

We conclude

$$\begin{aligned}
(F, m_1, m_2) \in \mathsf{V}\Lambda B_{be} &\leftrightarrow \bigvee_{(\alpha_1, \ldots, \alpha_r)} \bigwedge_{(\beta_1, \ldots, \beta_s)} F(\alpha_1, \ldots, \alpha_r, \beta_1, \ldots, \beta_s) = 1 \\
&\leftrightarrow \bigvee_{(\alpha_1, \ldots, \alpha_r)} \bigwedge_{(\beta_1, \ldots, \beta_s)} F'(\alpha_1, \ldots, \alpha_r, 1, \beta_1, \ldots, \beta_s) = 1 \\
&\leftrightarrow L(F') \text{ has a component of length } 2^s \text{ or more.} \quad \square
\end{aligned}$$

Now we deal with some problems in which counting is involved.

Cardinality Problem

Given: A description D of a finite set $L(D) \subseteq \mathbb{N}$, $m \in \mathbb{N}$.
Question: Does $L(D)$ have at least m elements?

Number of Components Problem

Given: A description D of a finite set $L(D)\subseteq \mathbb{N}$, $m\in\mathbb{N}$.
Question: Does $L(D)$ have at least m components?

Theorem 18. *The cardinality problem and the number of components problem are*

1. $\mathsf{C}\vee\mathscr{P}$-complete for integer-expressions and GHIL descriptions,
2. $\mathsf{C}\mathscr{P}$-complete for boolean expressions and combinatorial circuits.

Proof. Statement 1. The problems are in $\mathsf{C}\vee\mathscr{P}$ because of the equivalence:

$$L(D) \text{ has at least } m \text{ elements} \leftrightarrow \overset{m}{\underset{a}{\mathsf{C}}} \bigvee_x \varphi_D(x)=a,$$

and the equivalence:

$$L(D) \text{ has at least } m \text{ components} \leftrightarrow \overset{m}{\underset{a}{\mathsf{C}}} (\bigvee_x \varphi_D(x)=a \wedge \bigwedge_x \varphi_D(x)\neq a+1).$$

The latter equivalence verifies only that the number of components problem is in $\mathsf{C}B(\vee\mathscr{P})$. However, by Theorem 4.5 we have $\mathsf{C}B(\vee\mathscr{P})=\mathsf{C}\vee\mathscr{P}$. On the other hand, the $\mathsf{C}\vee\mathscr{P}$-complete problem $\mathsf{C}\vee B_{ie}$ (see Theorem 8) is reducible to the cardinality problem as follows. For the two-dimensional integer-expression H let $N=2^{|H|}$. Now let $\hat{H}$ be that one-dimensional integer-expression which arises from H by replacing every (a,b) by $bN+a$. Hence $(a,b)\in L(H)$ iff $(bN+a)\in L(\hat{H})$, and we can conclude

$$\begin{aligned}(H,b,m,d)\in\mathsf{C}\vee B_{ie} &\leftrightarrow \overset{m}{\underset{a\leqq d}{\mathsf{C}}} (a,b)\in L(H)\\
&\leftrightarrow \overset{m}{\underset{a\leqq d}{\mathsf{C}}} (bN+a)\in L(\hat{H})\\
&\leftrightarrow \operatorname{card}(\{bN, bN+1,\dots,bN+d\}\cap L(\hat{H}))\geqq m\\
&\leftrightarrow \operatorname{card}(\{0,1,2,\dots,bN+d\}\cap L(\hat{H}\cup H(bN-1))\geqq bN+m\\
&\leftrightarrow \operatorname{card}(\{0,1,2,\dots,bN+d+N^2+1\}\cap L(\hat{H}\cup H(bN-1)\\
&\qquad\qquad \cup((bN+d+1)+H(N^2))))\geqq bN+m+N^2+1\\
&\leftrightarrow \operatorname{card} L(\hat{H}\cup H(bN-1)\cup((bN+d+1)+H(N^2)))\\
&\qquad \geqq bN+m+N^2+1.\end{aligned}$$

Finally, the cardinality problem can be reduced to the number of components problem by:

$$\operatorname{card} L(H)\geqq m \leftrightarrow \text{number of components in } L(2H) \text{ is at least } m.$$

Statement 2. The problems are in $\mathsf{C}\mathscr{P}$ because of the equivalence:

$$L(D) \text{ has at least } m \text{ elements} \leftrightarrow \overset{m}{\underset{a}{\mathsf{C}}}\, a\in L(D),$$

and the equivalence:

$$L(D) \text{ has at least } m \text{ components} \leftrightarrow \overset{m}{\underset{a}{\mathsf{C}}} (a\in L(D) \wedge a+1\notin L(D)).$$

On the other hand, the cardinality problem for boolean expressions has been shown to be $\mathsf{C}\mathscr{P}$-complete in [19] (the problem has been called m-SATISFYABILITY there). Finally, the cardinality problem is reducible to the number of components problem by

$$\begin{aligned}&\operatorname{card} L(F(x_1, \ldots, x_k)) \geqq m \leftrightarrow L(F(x_1, \ldots, x_k) \wedge x_{k+1})\\ &\text{has at least } m \text{ components.} \quad \square\end{aligned}$$

Note that Theorem 18.1 improves a result in [2] where it has been shown for the more general case of GHIL descriptions of finite sets of rectangles in the plane that the cardinality problem is $\mathscr{NP}$-hard.

Cardinality of Projection Problem

Given: A description D of a finite set $L(D) \subseteq \mathbb{N}^2$, $m \in \mathbb{N}$.
Question: Does the projection of $L(D)$ to the first place have at least m elements, i.e. $\overset{m}{\underset{a}{\mathsf{C}}} \bigvee_b (a, b) \in L(D)$?

Theorem 19. *The cardinality of projection problem is*

1. $\mathsf{C}\bigvee\mathscr{P}$*-complete for integer-expressions and* GHIL *descriptions,*
2. $\mathsf{C}\bigvee\mathscr{P}$*-complete for boolean expressions and combinatorial circuits.*

Proof. Statement 1. The problem is in $\mathsf{C}\bigvee\mathscr{P}$ because of the equivalence:

$$\overset{m}{\underset{a}{\mathsf{C}}} \bigvee_b (a, b) \in L(D) \leftrightarrow \overset{m}{\underset{a}{\mathsf{C}}} \bigvee_b \bigvee_x \varphi_D(x) = (a, b).$$

On the other hand, the $\mathsf{C}\bigvee\mathscr{P}$-complete cardinality problem (see Theorem 18.1) can be reduced to the cardinality of projection problem as follows. For a one-dimensional integer-expression H let $\hat{H}$ that two-dimensional integer-expression which arises from H by replacing every a by $(a, 0)$. Hence $(a, b) \in L(\hat{H})$ iff $(a \in L(H)$ and $b = 0)$. We conclude

$$\operatorname{card} L(H) \geqq m \leftrightarrow \overset{m}{\underset{a}{\mathsf{C}}}\, a \in L(H) \leftrightarrow \overset{m}{\underset{a}{\mathsf{C}}}\, (a, 0) \in L(\hat{H}) \leftrightarrow \overset{m}{\underset{a}{\mathsf{C}}} \bigvee_b (a, b) \in L(\hat{H}).$$

Statement 2. The representation in the definition shows that the problem is in $\mathsf{C}\bigvee\mathscr{P}$. On the other hand, for boolean expressions the cardinality of projection problem is identically with the $\mathsf{C}\bigvee\mathscr{P}$-complete problem $\mathsf{C}\bigvee B_{be}$ (see Theorem 7). $\square$

Number of Components of a Given Length Problem

Given: A description D of a finite set $L(D) \subseteq \mathbb{N}$, m, $k \in \mathbb{N}$.
Question: Does $L(D)$ have at least m components of length at least k?

Theorem 20. *The number of components of a given length problem is*

1. $\mathsf{C}\bigwedge\bigvee\mathscr{P}$*-complete for integer-expressions and* GHIL *descriptions,*
2. $\mathsf{C}\bigwedge\mathscr{P}$*-complete for boolean expressions and combinatorial circuits.*

Proof. As for the component length problem but using $\overset{m_1}{\mathsf{C}}$ instead of V. □

The following problems are connected with the multiplicity of the description by integer-expressions and GHIL descriptions. Therefore the corresponding results are only on these two descriptional languages.

Unambiguity Problem

Given: A description of a finite set $L(D) \subseteq \mathbb{N}$.
Question: Is every element of $L(D)$ described only once by D, i.e.

$$\bigwedge_x \bigwedge_y (\varphi_D(x) = \varphi_D(y) \rightarrow x = y)?$$

Theorem 21. *The unambiguity problem is* co$\mathcal{NP}$*-complete for integer-expressions and* GHIL *descriptions.*

Proof. That the problem is in co$\mathcal{NP}$ follows from its representation in the definition. On the other hand, we can show that the idea in [2] to reduce the $\mathcal{NP}$-complete sum of subset problem (see [4]) to the multioccurence problem for the more general case of GHIL descriptions of finite sets of rectangles in the plane can also be used to reduce the complement of the sum of subset problem to our unambiguity problem. The sum of subset problem is defined as follows: Given natural numbers $k, a_1, \ldots, a_k, b$, can one find a subset of $\{a_1, \ldots, a_k\}$ whose sum is b? Obviously, there exists such a subset iff $b \in L\left(\sum_{i=1}^{k} (0 \cup a_i)\right)$ (this was the idea in [20] to show that the membership problem for integer-expressions is $\mathcal{NP}$-complete). Unfortunately, the integer-expression $\sum_{i=1}^{k} (0 \cup a_i)$ might be not unambiguious. Therefore we set $N = \sum_{i=1}^{k} a_i + 1$ and use the unambigious integer-expression

$$H \equiv \sum_{i=1}^{k} (0 \cup (2^i \cdot N + a_i))$$

for which we can conclude:

there does not exist a subset of $\{a_1, \ldots, a_k\}$ whose sum is $b \leftrightarrow$
$\leftrightarrow$ there does not exist a $c < 2^{k-1}$ such that $cN + b \in L(H)$
$\leftrightarrow H \cup (N \cdot H_{k-1} + b)$ is unambigious. □

Multiplicity Problem

Given: A description D of a finite set $L(D) \subseteq \mathbb{N}$, $a, m \in \mathbb{N}$.
Question: Is a described by D in at least m different ways, i.e. $\mathrm{mult}_D(a) \geq m$?

For integer-expressions this problem is identically with the $\mathsf{C}\mathcal{P}$-complete problem $\mathsf{C}B_{ie}$ (see Theorem 8). Therefore we have

Theorem 22. *The multiplicity problem is* $\mathsf{C}\mathscr{P}$*-complete for integer-expressions and* GHIL *descriptions.* □

Maximum Multiplicity Problem

Given: A description D of a finite set $L(D)\subseteq\mathbb{N}$, $m\in\mathbb{N}$.
Question: Is the maximum multiplicity of D at least m, i.e.

$$\bigvee_{a} \overset{m}{\underset{x}{\mathsf{C}}}\ \varphi_D(x)=a?$$

Theorem 23. *The maximum multiplicity problem is* $\vee\mathsf{C}\mathscr{P}$*-complete for integer-expressions and* GHIL *descriptions.*

Proof. That the problem is in $\vee\mathsf{C}\mathscr{P}$ follows from its representation in the definition. On the other hand, the $\vee\mathsf{C}\mathscr{P}$-complete problem $\vee\mathsf{C}B_{ie}$ (see Theorem 8) can be reduced to the maximum multiplicity problem as follows. For the two-dimensional integer-expression H define $N=2^{|H|+1}$ and let $\hat{H}$ be that one-dimensional integer-expression which arises from H by replacing every (a,b) by $bN+a$. Consequently, $(a,b)\in L(H)$ iff $(bN+a)\in L(\hat{H})$, and we conclude

$$\begin{aligned}(H,b,m_1,m,d)\in\vee\mathsf{C}B_{ie}&\leftrightarrow\bigvee_{a\leq d}\operatorname{mult}_H(a,b)\geq m\\&\leftrightarrow\bigvee_{a\leq d}\operatorname{mult}_{\hat{H}}(bN+a)\geq m\\&\leftrightarrow\bigvee_{bN\leq a\leq bN+d}\operatorname{mult}_{\hat{H}}(a)\geq m\\&\leftrightarrow\bigvee_{a}\operatorname{mult}_{H_1}(a)\geq N+m,\end{aligned}$$

where $H_1\equiv\hat{H}\cup\left(bN+H(d)+\sum_{i=1}^{|H|+1}(0\cup 0)\right)$, i.e. the elements of $\{bN,bN+1,bN+2,\ldots,bN+d\}$ are added to $L(\hat{H})$ with multiplicity N. The elements of $L(\hat{H})$ do not have such a high multiplicity (see Proposition 2). □

Minimum Multiplicity Problem

Given: A description D of a finite set $L(D)\subseteq\mathbb{N}$, $m\in\mathbb{N}$.
Question: Is the minimum multiplicity of D at least m, i.e.

$$\bigwedge_{a}(a\in L(D)\rightarrow\operatorname{mult}_D(a)\geq m).$$

Theorem 24. *The minimum multiplicity problem is* $\wedge\mathsf{C}\mathscr{P}$*-complete for integer-expressions and* GHIL *descriptions.*

Proof. The problem is in $\wedge\mathsf{C}\mathscr{P}$ because of the equivalence

$$\bigwedge_{a}(a\in L(D)\rightarrow\operatorname{mult}_D(a)\geq m)\leftrightarrow\bigwedge_{x}\overset{m}{\underset{y}{\mathsf{C}}}\ \varphi_D(y)=\varphi_D(x).$$

Table 1. Summary of complexity results

Problem	Complexity for	
	IE/GHIL	BE/CC
Membership	$\mathcal{NP}$-complete	in $\mathcal{L}/\mathcal{P}$-complete
Non-emptiness	in $\mathcal{L}$	$\mathcal{NP}$-complete
Critical element	$\mathcal{NP}$-complete	co $\mathcal{NP}$-complete
Intersection	$\mathcal{NP}$-complete	$\mathcal{NP}$-complete
Isolated element	D^P-complete	in $\mathcal{L}/\mathcal{P}$-complete
Maximum element below a boundary	D^P-complete	co $\mathcal{NP}$-complete
Subset, equality	Π_2^p-complete	co $\mathcal{NP}$-complete
Connectedness	Π_2^p-complete	co $\mathcal{NP}$-complete
Component length	Σ_3^p-complete	Σ_2^p-complete
Cardinality, number of components	$\mathrm{C}\vee\mathcal{P}$-complete	$\mathrm{C}\mathcal{P}$-complete
Cardinality of projection	$\mathrm{C}\vee\mathcal{P}$-complete	$\mathrm{C}\vee\mathcal{P}$-complete
Number of components of a given length	$\mathrm{C}\wedge\vee\mathcal{P}$-complete	$\mathrm{C}\vee\mathcal{P}$-complete
Unambiguity	co $\mathcal{NP}$-complete	–
Multiplicity	$\mathrm{C}\mathcal{P}$-complete	–
Maximum multiplicity	$\vee\mathrm{C}\mathcal{P}$-complete	–
Minimum multiplicity	$\wedge\mathrm{C}\mathcal{P}$-complete	–
Number of elements of a given multipl.	$\mathrm{C}\mathrm{C}\mathcal{P}$-complete	–

On the other hand, the $\wedge\mathrm{C}\mathcal{P}$-complete problem $\wedge\mathrm{C}B_{ie}$ (see Theorem 8) can be reduced to the minimum multiplicity problem as follows. For the two-dimensional integer-expression H we define N and $\hat{H}$ as in the proof of Theorem 23, and we conclude

$$\begin{aligned}(H,b,m_1,m,d)\in\wedge\mathrm{C}B_{ie} &\leftrightarrow \bigwedge_{a\leqq d}\ \mathrm{mult}_H(a,b)\geqq m\\ &\leftrightarrow \bigwedge_{a\leqq d}\ \mathrm{mult}_{\hat{H}}(bN+a)\geqq m\\ &\leftrightarrow \bigwedge_{bN\leqq a\leqq bN+d}\ \mathrm{mult}_{\hat{H}}(a)\geqq m\\ &\leftrightarrow \bigwedge_{a}\ \mathrm{mult}_{H_2}(a)\geqq m,\end{aligned}$$

where

$$H_2\equiv\hat{H}\cup\left((H(bN-1)\cup(bN+d+1+H(N^2)))+\sum_{i=1}^{|H|+1}(0\cup 0)\right),$$

i.e. the elements of $\{0,1,2,\ldots,bN-1,bN+d+1,bN+d+2,\ldots,bN+d+1+N^2\}$ are added to $L(\hat{H})$ with multiplicity N. And if $\mathrm{mult}_{\hat{H}}(a)\geqq m$ for some a we have $N\geqq m$ by Proposition 2. □

Number of Elements of a Given Multiplicity

Given: A description D of a finite set $L(D)\subseteq\mathbb{N}$, k, $m\in\mathbb{N}$.
Question: Do there exist at least k elements in $L(D)$ with multiplicity at least m, i.e.

$$\overset{k}{\underset{a}{\mathrm{C}}}\ \overset{m}{\underset{x}{\mathrm{C}}}\ \varphi_D(x)=a?$$

Theorem 25. *The number of elements of a given multiplicity problem is* $\mathsf{C}\mathcal{C}\mathcal{P}$-*complete for integer-expressions and* GHIL *descriptions.*

Proof. As for the maximum multiplicity problem but using $\overset{m_1}{\mathsf{C}}$ instead of V. □

We conclude this section by a summary of results given by Table 1.

5. Conclusions

The research of the present paper is continued in [25] and [27]. In [25] we investigate the complexity of graph problems when the graphs are described by descriptional languages. We mention only two results which are characteristically for the type of results proved in this paper.

1. It is known that the graph accessibility problem is $\leq_m^{\log}$-complete in nondeterministic logarithmic space for directed graphs, whereas this completeness is not known for undirected graphs. When the graphs are described by our descriptional languages both problems become complete in polynomial space.

2. The $\mathcal{NP}$-complete colourability problem becomes complete in nondeterministic exponential time when the graphs are described by our descriptional languages.

In [27] we investigate in a broader context of languages describing finite subsets of $\mathbb{N}$ the complexity of questions about the maximum or the minimum of the described subset. Typical results of this paper are:

1. The problem of whether $\max L(H)$ is odd for a given boolean expression H is $\leq_m^{\log}$-complete in Δ_2^p (the closure of $\mathcal{NP}$ with respect to the polynomially bounded Turing reducibility).

2. The problem of whether the chromatic number of a given graph is odd is $\leq_m^p$-complete in the closure of $\mathcal{NP}$ with respect to a special kind of polynomially bounded tt-reducibility.

The ideas discussed above are connected with the notion of Kolmogorov complexity. A string D of our descriptional language describes (succinctly) the finite set $L(D)$ where the function L is recursive, i.e. there is a machine M_i such that $L(D)=M_i(D)$. For a given problem A we have considered the derived problem $A_i=\{D\colon M_i(D)\in A\}$ and its complexity. Obviously, the complexity of A_i is influenced by those instances D which describe large sets, i.e. for which $|M_i(D)|$ is not far from the upper bound $2^{|D|}$. However, for arbitrary $i\in\mathbb{N}$ the length of $M_i(x)$ cannot be bounded effectively in terms of $|x|$ and, as a consequence, the set A_i might be undecideable. Therefore we define for every $t(n)\geq n$:

$$A_i^t=\{D\colon M_i(D)\in A \text{ and } M_i \text{ runs at most } t(|x|)^2 \text{ steps on input } x\}.$$

It is not hard to see that for "good" bounding functions t there exist (universal) machines M_u such that, the $\mathcal{NP}$-completeness of the set A implies that A_u^t is complete in $\bigcup_{k\geq 1}$ NTIME$(t(n^k))$. However, this does not mean that the set

$$A \cap \{y \colon \bigvee_x (t(|x|) \leqq |y| \text{ and } M_u(x) = y \text{ in at most } |y|^2 \text{ steps})\}$$

remains $\mathcal{NP}$-complete if A is $\mathcal{NP}$-complete. This question has been discussed in [6].

Acknowledgements. I am grateful to Jörg Vogel (Jena), Burkhard Monien (Paderborn), and Uwe Schöning (Stuttgart) for some interesting discussions on this topic, and the unknown referees for their helpful hints.

References

1. Angluin, D.: On counting problems and the polynomial-time hierarchy. Theor. Comput. Sci. **12**, 161–173 (1980)
2. Bentley, J.L., Ottmann, T., Widmayer, P.: The complexity of manipulating hierarchically defined sets of rectangles. Adv. Comput. Res. **1**, 127–158 (1983)
3. Gill, J.: Computational complexity of probabilistic Turing machines. SIAM J. Comput. **6**, 675–695 (1977)
4. Garey, M.R., Johnson, D.S.: Computers and Intractibility: A Guide to the Theory of *NP*-Completeness. San Francisco: Freeman 1979
5. Galperin, H., Wigderson, A.: Succinct representations of graphs. Inf. Control. **56**, 183–198 (1983)
6. Hartmanis, J.: Generalized Kolmogorov complexity and the structure of feasible computations. Proc. 24th Ann. Symp. Found. of Comp. Sci. 439–445 (1983)
7. Lautemann, C.: BPP and the polynomial hierarchy. IPL **17**, 215–217 (1983)
8. Ladner, R.E.: The circuit value problem is logspace complete for *P*. SIGACT News **7**, 18–20 (1975)
9. Lengauer, T.: The complexity of compacting hierarchically specified layouts of integrated circuits. Proc. 23rd Ann. Symp. on Found. of Comp. Sci. 358–368 (1982)
10. Lengauer, T.: Hierarchical planarity testing algorithms. TR-SFB 124, FB 10 Univ. d. Saarlandes, Saarbrücken 1984
11. Lengauer, T.: Hierarchical graph algorithms, TR-SFB 124, 15/84, FB 10 Univ. d. Saarlandes, Saarbrücken 1984
12. Lengauer, T.: Efficient solution of connectivity problems on hierarchically defined graphs. Rep. No. 24, Series Theoretische Informatik, Universität Paderborn 1985
13. Lynch, N.A.: Log space recognition and translation of parenthesis languages. J. Assoc. Comput. Mach. **24**, 583–590 (1977)
14. Meyer, A.R.: Weak monadic second order theory of successor is not elementary recursive. Proc. 14th Ann. Symp. on Switch and Autom Theory 190–196 (1973)
15. Papadimitriou, C.H.: On the complexity of unique solution. Proc. 23rd Ann. Symp. on Found. of Comp. Sci. 14–20 (1982)
16. Papadimitriou, C.H., Yannakakis, M.: The complexity of facets (and some facets of complexity). Proc. 14th ACM Symp. on Theory of Comp. 255–260 (1982)
17. Papadimitriou, C.H., Zachos, S.K.: Two remarks on the power of counting. (Preprint 1982)
18. Schöning, U.: On small generators. Theor. Comput. Sci. **34**, 337–341 (1984)
19. Simon, J.: On the difference between one and many. Proc. 14th Intern. Coll. on Autom., Lang. and Progr., Lect. Notes Comput. Sci. **52**, 480–491 (1977)
20. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time. Proc. 5th ACM Symp. on Theory of Comp. 1–9 (1973)
21. Stockmeyer, L.J.: The polynomial-time hierarchy, Theor. Comput. Sci. **3**, 1–22 (1977)
22. Stockmeyer, L.J.: The complexity of approximate counting. Proc. 15th ACM Symp. on Theory of Comp. 118–126 (1983)
23. Valiant, L.G.: The complexity of computing the permanent. Theor. Comput. Sci. **8**, 189–201 (1979)

24. Wagner, K.: Compact descriptions and the counting polynomial-time hierarchy. Proc. 2nd Frege Conf. 383–392 (1984)
25. Wagner, K.: The complexity of problems concerning graphs with regularities. Proc. 11th Symp. on Math. Found. of Comp. Sci., Lect. Notes Comput. Sci. **176**, 544–552 (1984)
26. Wagner, K.: Some observations on the connection between counting and recursion. (To appear in Theor. Comput. Sci.)
27. Wagner, K.: More complicated questions about maxima and minima, and some closures of *NP*. (Manuscript 1985)
28. Wrathall, C.: Complete sets and the polynomial-time hierarchy. Theor. Comput. Sci. **3**, 23–33 (1977)
29. Wagner, K., Wechsung, G.: Computational Complexity. Berlin: Deutscher Verlag der Wissenschaften 1985
30. Yao, A.C.: Seperating the polynomial-time hierarchy by oracles. (To appear in the Proc. of 26th Ann. Symp. on Found. of Comp. Sci. 1985)
31. Zachos, S.K.: Collapsing polynomial hierarchies. Proc. of a Conf. on Comp. Compl. Theory, Santa Barbara 75–81 (1983)
32. Zachos, S.K., Heller, H.: A decisive characterization of BPP. (Manuscript 1985)

Received September 10, 1984/Dezember 4, 1985