On Cluster Machines and Function Classes*

Sven Kosub
Lehrstuhl für Theoretische Informatik
Julius-Maximilians-Universität Würzburg
kosub@informatik.uni-wuerzburg.de

May 1997

Abstract

We consider a special kind of non-deterministic Turing machines. Cluster machines are distinguished by a neighbourhood relationship between accepting paths. Based on a formalization using equivalence relations some subtle properties of these machines are proven. Moreover, by abstraction we gain the machine-independend concept of cluster sets which is the starting point to establish cluster operators. Cluster operators map complexity classes of sets into complexity classes of functions where for the domain classes only cluster sets are allowed. For the counting operator c#· and the optimization operators cmax· and cmin· the structural relationships between images resulting from these operators on the polynomial-time hierarchy are investigated. Furthermore, we compare cluster operators with the corresponding common operators #·, max· and min· [Tod90b, HW97].

1 Introduction

Clarifying relationships between determinism and non-determinism is one of the most important subjects in complexity theory. Except for a small number of cases (see e.g. [Sav70, PPST83]) such problems are still unsolved, in particular in the polynomial-time case. In order to throw light upon that problem it is meaningful to analyze intermediate concepts like unambiguity or fewness. On the side of classes of sets these concepts have lead to various complexity classes located between P and NP such as UP [Val76], FewP [All86, AR88] or R [Gil77] (VPP in Gill's notion).

However, the same questions about function classes kept widely unnoticed, function classes between FP and #P [Val79a, Val79b] were not often considered (as an exception: [HV95]). Trying to close the gap a suitable machine concept will be developed. Under polynomial time-restrictions generally being assumed this concept should have some properties, e.g. the class of languages decidable via such machines should be located between P and NP, and counting or other operations should be possible within the model. In that sense we should have a concept easier than non-determinism and harder than determinism.

*Most of this work was done as a master's thesis at Friedrich-Schiller-Universität Jena [Kos96].

We start with an observation: Running a non-deterministic computation on an input it may be that all accepting paths are neighbours. Say that in such special case the principle of neighbouring accepting paths holds. Making a definition from this principle in sense of that for a non-deterministic machine and for every input the neighbourhood relationship must be satisfied we obtain the concept of a cluster machine. A cluster is just the set of all accepting paths.

Note that we use the empty set as a cluster. Why do we remark this? Supposed the empty set is not a cluster, then there exist no clusters on computation trees having no accepting paths. On the other hand, if empty clusters are admitted then we have lost uniqueness of clusters. In order to avoid limits to the existence of clusters we will accept such fuzzieness. Note also that typically for semantic machine concepts cluster machines are probably non-enumerable as pointed out by Hartmanis and Hemachandra for UP-machines [HH88]. Finally note that defining cluster machines makes only sense on balanced trees caused by using an order on computation trees. Supposing balanced trees can be essential as investigated in [HVW96].

Where does clustering locate between determinism and non-determinism? Obviously, every machine witnessing a language in UP suffices for the principle of neighbouring accepting paths. Hence, clustering is at least as hard as unambiguity. Regarding to the acceptance power both concepts even coincide. In typical case this can be seen with a simple thought. If a cluster machine runs along a rejecting path, then a slightly modified machine shall continue with the next path in a quasi-lexikographical order, and accepts if this neighbour-path is accepting. Since we have a cluster machine our new machine has exactly one accepting path if and only if actually accepting paths for the original exist. Consequently, the language decided in this way belongs to UP. To prove these or similar facts exactly we use equivalence relations to formalize our concept.

We want to go one step further. A formalization of clustering by equivalence classes makes it possible to embed investigations about the functional power of cluster machines in a more abstract context. If we neglect that clustering is a concept on computations of machines we can generally introduce cluster sets. Cluster sets are sets of pairs $\langle x, y \rangle$ such that for every fixed x the set of all related second components only contain such polynomially bounded y that are neighbours with respect to quasi-lexicographical order. Clearly, cluster sets in P represent exactly computation trees of cluster machines. But how does clustering work on sets from NP or coNP? This question will be tackled by applying some functional operators to cluster sets.

Since introduced in [Tod90b, Wra77] operators became a great subject in complexity theory. Although first mostly restricted to classes of sets Toda had already defined the functional operator $\#\cdot$ on complexity classes of sets (num$\cdot$ in Toda's notion [Tod90b]) as a generalization of Valiant's class of counting functions #P [Val79a, Val79b] — such that $\# \cdot \mathrm{P} = \#\mathrm{P}$. Subsequently other function classes such as OptP [Kre88, Kre92, GKR95], SpanP [Köb89, KST89], MidP [Tod90a] or GapP [FFK94] were generalized by operators in several manners (see e.g. [VW93, Vol94, HV95, VW95, HW97]).

We will focus our attention to the operators $\#\cdot, \max\cdot$ and $\min\cdot$ [Tod90b, HW97] considered under restriction of cluster sets. Such operators, called cluster operators, will be denoted by adding the prefix letter c to the operators. Note that Vollmer [Vol94, VW95] defined an operator $\mathrm{F}\cdot$ using maximization on (implicitely considered) cluster sets fixed on smallest words of a certain polynomial length (representing the zero path). For that operator $\mathrm{F} \cdot \mathrm{P} = \mathrm{FP}$ holds.

Using similar characterization as for #P we have an inclusion structure $\mathrm{FP} \subseteq \mathrm{c\#} \cdot \mathrm{P} \subseteq \# \cdot \mathrm{P}$. As we will see, further function classes, namely $\mathrm{cmax} \cdot \mathrm{P}$ and $\mathrm{cmin} \cdot \mathrm{P}$, are located between FP and $\# \cdot \mathrm{P}$. Moreover we get some very interesting results inside the structural relations of cluster operators in application to the polynomial-time hierarchy [MS72, Sto77] and in relation to common operators, e.g. only having the concept of cluster sets, for classes $\max \cdot \mathrm{NP}$ and $\min \cdot \mathrm{NP}$ we get better upper bounds $\mathrm{cmin} \cdot \mathrm{coNP}$ and $\mathrm{cmax} \cdot \mathrm{coNP}$ respectively than pointed out in [HW97].

2 Preliminaries

We assume familiarity with general notions of complexity theory [BDG90, BDG95, Pap94]. Throughout this paper, we use the finite alphabet $\Sigma = \{0, 1\}$. Our computational model is the non-deterministic standard Turing machine [BDG95, Pap94]. Sometimes machines with the additional facility to make outputs are considered. We describe a computation of a machine M on input $x \in \Sigma^*$ by a computation tree $M(x)$. To define what is a cluster machine we suppose strict non-determinism, i.e. deterministic computation steps are not allowed. Furthermore, in the case of polynomial time-resources a computation on an input $x \in \Sigma^*$ runs exactly $p(|x|)$ steps where p is the polynomial. Thus, we suppose complete binary computation trees. On computation trees $M(x)$ we define $acc_M(x)$ as the set of all accepting paths of $M(x)$, and $out^+_M(x)$ as the set of all outputs along accepting paths of $M(x)$ whereas let all outputs along rejecting paths of $M(x)$ be contained in $out^-_M(x)$.

Next review the definitions of some common operators we are interested in.

Definition 2.1. [Tod90b, HW97]. Let $\mathcal{C}$ be an arbitrary complexity class.

1. $\# \cdot \mathcal{C}$ is the class defined in the following way: A function f is in $\# \cdot \mathcal{C}$ if there exist a set $A \in \mathcal{C}$ and a polynomial p such that $f(x) = \#\{y \mid |y| = p(|x|) \wedge \langle x, y \rangle \in A\}$ for every $x \in \Sigma^*$.

2. $\max \cdot \mathcal{C}$ is the class defined in the following way: A function f is in $\max \cdot \mathcal{C}$ if there exist a set $A \in \mathcal{C}$ and a polynomial p such that $f(x) = \max\{y \mid |y| = p(|x|) \wedge \langle x, y \rangle \in A\}$ for every $x \in \Sigma^*$. If $\{y \mid |y| = p(|x|) \wedge \langle x, y \rangle \in A\} = \varnothing$ then we define $f(x) = 0$.

3. $\min \cdot \mathcal{C}$ is the class defined in the following way: A function f is in $\min \cdot \mathcal{C}$ if there exist a set $A \in \mathcal{C}$ and a polynomial p such that $f(x) = \min\{y \mid |y| = p(|x|) \wedge \langle x, y \rangle \in A\}$ for every $x \in \Sigma^*$. If $\{y \mid |y| = p(|x|) \wedge \langle x, y \rangle \in A\} = \varnothing$ then we define $f(x) = 0$.

Note that these operators yield classes of total functions (where we set the function value for minimization to 0 on empty set instead $2^{p(|x|)}$ as used in [HW97]). In contrary to this, e.g. so called multivalued functions or several special cases of this function type (see e.g. [BLS84, Sel94, FGH$^+$96]) are generally partial functions.

To the purpose of comparison with later results recall some structural relations between the common operators defined above. The following results completely cover valid inclusions on the lowest stage of the polynomial-time hierarchy.

Proposition 2.2. [HW97].

1. $\mathrm{FP} \subseteq \max \cdot \mathrm{P} \subseteq \max \cdot \mathrm{NP} \subseteq \mathrm{F}\Delta_2^{\mathrm{P}}$.

2. $\max \cdot \mathrm{P} \subseteq \max \cdot \mathrm{coNP} \subseteq \max \cdot \Delta_2^{\mathrm{P}}$.

3. $\max \cdot \mathrm{NP} \subseteq \min \cdot \mathrm{coNP}$.

4. $\mathrm{FP} \subseteq \min \cdot \mathrm{P} \subseteq \min \cdot \mathrm{NP} \subseteq \mathrm{F}\Delta_2^{\mathrm{P}}$.

5. $\min \cdot \mathrm{P} \subseteq \min \cdot \mathrm{coNP} \subseteq \min \cdot \Delta_2^{\mathrm{P}}$.

6. $\min \cdot \mathrm{NP} \subseteq \max \cdot \mathrm{coNP}$.

Proposition 2.3. [Köb89, KST89].

1. $\max \cdot \mathrm{NP} \subseteq \# \cdot \mathrm{NP}$.

2. $\# \cdot \mathrm{coNP} = \# \cdot \Delta_2^{\mathrm{P}}$.

3 Clusters and Equivalence Relations

According to the principle of neighbouring accepting paths we take into a cluster such paths of computation trees yielding an integer interval when interpreted as natural numbers. We formalize this by introducing an equivalence relation.

Definition 3.1. Let M be a non-deterministic Turing machine and let y, z be two paths of the computation tree on $M(x)$ for an input $x \in \Sigma^*$.

1. The paths y and z have the relation $\equiv_{M(x)}$ if and only if (1) $|y - z| = 1$ (i.e. y and z are neighbouring paths) and (2) $M(x)$ along y is accepting if and only if $M(x)$ along z is accepting.
2. For $k \in \mathbb{N}$ and $k > 0$ the paths y and z have the relation $\equiv^k_{M(x)}$ if and only if there exist $k-1$ different paths $y_1, \ldots, y_{k-1} \in M(x)$ such that $y \equiv_{M(x)} y_1, y_1 \equiv_{M(x)} y_2, \ldots, y_{k-1} \equiv_{M(x)} z$. For $k = 0$, we define $y \equiv^0_{M(x)} z \Leftrightarrow y = z$.
3. The paths y and z have the relation $\equiv^*_{M(x)}$ if and only if there exists a $k \in \mathbb{N}$ such that $y \equiv^k_{M(x)} z$.

Generated by the relation $\equiv_{M(x)}$ the relation $\equiv^*_{M(x)}$ is reflexive, symmetric, and transitive and, consequently, an equivalence relation. With respect to this equivalence relation the paths of a computation tree $M(x)$ are disjointly divisible in equivalence classes. Then a cluster is an equivalence class whose representatives are accepting paths of $M(x)$. Recall that empty sets are also clusters. We gain the concept of a cluster machine by requiring for a non-deterministic machine that for every $x \in \Sigma^*$ all accepting paths are always contained in an equivalence class.

Definition 3.2. A non-deterministic Turing machine M is a *cluster machine* if and only if for every $x \in \Sigma^*$ there is a path $y \in M(x)$ such that $\{z | z \equiv^*_{M(x)} y \wedge y \in \mathrm{acc}_M(x)\} = \mathrm{acc}_M(x)$.

In order to prove some properties of cluster machines we need the concept of k-fold (weakly k-fold, resp.) computations of a function.

Definition 3.3. Let $f : \Sigma^* \longrightarrow \mathbb{N}$.

1. A non-deterministic Turing machine M computes the function f k-*fold* if and only if $\mathrm{out}^+_M(x) = \{f(x)\}$ and $\#\mathrm{acc}_M(x) = k$ for every $x \in \Sigma^*$.
2. A non-deterministic Turing machine M computes the function f *weakly* k-*fold* if and only if $f(x) > 0 \Rightarrow \mathrm{out}^+_M(x) = \{f(x)\} \wedge \#\mathrm{acc}_M(x) = k$ and $f(x) = 0 \Rightarrow \mathrm{out}^+_M(x) = \varnothing$ for every $x \in \Sigma^*$.

The following results show the relationship between cluster machines and (weakly) onefold computable functions.

Theorem 3.4. *Let* $f : \Sigma^* \longrightarrow \mathbb{N}$.

1. *If a non-deterministic polynomial-time Turing machine* M *computes the function* f *onefold then there is a polynomial-time cluster machine* N *such that* $\#\mathrm{acc}_N(x) = f(x)$ *for all* $x \in \Sigma^*$.
2. *For every polynomial-time cluster machine* N *fulfilling* $\#\mathrm{acc}_N(x) > 0$ *for every* $x \in \Sigma^*$ *there is a non-deterministic polynomial-time Turing machine* M *that computes the function* $\#\mathrm{acc}_N$ *onefold.*
3. *There is a non-deterministic polynomial-time Turing machine* M *that computes the function* f *weakly onefold if and only if there exists a polynomial-time cluster machine* N *fulfilling* $\#\mathrm{acc}_N(x) = f(x)$ *for every* $x \in \Sigma^*$.

Proof. (1) is easy to verify and thus omitted.

(2) Let N be a polynomial-time cluster machine such that, without loss of generality, the lexicographically smallest computation paths of N as well as the largest are always rejecting, and N is time-bounded by the polynomial p. Define K to be the machine that, on input $x \in \Sigma^*$ (a) guesses a path $y \in \Sigma^{=p(|x|)}$, (b) simulates $N(x)$ along y, (c) simulates $N(x)$ along $z = y+1$, where in the case that $y = 1^{p(|x|)}$ continues with $z = 0^{p(|x|)}$, (d) outputs y and accepts if and only if either y is accepting or z is accepting. Since N runs in polynomial time, K also does. By assumption then we have $\#\mathrm{acc}_K(x) = 2$ and $\mathrm{out}^+_K(x) = \{z_1, z_2\}$ with $|z_1 - z_2| = \#\mathrm{acc}_N(x)$ for every $x \in \Sigma^*$. Define M to be the machine that, on input $x \in \Sigma^*$, (a) guesses paths $w_1, w_2 \in K(x)$, (b) simulates $K(x)$ along w_1 and along w_2 one after the other, (c) accepts if and only if w_1 and w_2 are accepting and $z_1 < z_2$, where z_1 and z_2 are outputs of $K(x)$ along w_1 and along w_2 resp., and outputs $|z_1 - z_2|$. Clearly, M runs in polynomial time, and $\#\mathrm{acc}_M(x) = 1$ and $\mathrm{out}^+_M(x) = \{\#\mathrm{acc}_N(x)\}$ for every $x \in \Sigma^*$. Hence, M computes $\#\mathrm{acc}_N$ onefold.

(3) is similar to (1) and (2). ❑

Corollary 3.5. *For an arbitrary set* $A \in \mathrm{NP}$ *there is a polynomial-time cluster machine accepting* A *in the sense of* NP *if and only if* $\mathrm{NP} = \mathrm{UP}$.

This corollary illustrates the coincidence of unambiguity and clustering in sense of NP-acceptance. Cluster machines only accept sets from UP.

4 Cluster Operators and a cluster functions hierarchy

We want to lift our basic definitions from the machine level to sets. Here, a formalization is also based on the definition of the generator relation $\equiv_x$, of the generated equivalence relation $\equiv_x^*$ and on understanding equivalence classes as clusters.

Definition 4.1. Let $A \subseteq \Sigma^*$, and let $A_x = \{y | \langle x, y \rangle \in A\}$.

1. Two words $y, z \in \Sigma^*$ have the relation $\equiv_x$ if and only if $|y - z| = 1$ and $\langle x, y \rangle \in A \Leftrightarrow \langle x, z \rangle \in A$.
2. The relation $\equiv_x^*$ is defined as the equivalence relation generated by $\equiv_x$.
3. A set A is a *cluster set* if and only if for every $x \in \Sigma^*$ there exists a $y \in \Sigma^*$ such that $A_x = \{z | z \equiv_x^* y \wedge \langle x, y \rangle \in A\}$.

Now we can define some cluster operators we will use.

Definition 4.2. Let $\mathcal{C}$ be an arbitrary complexity class.

1. $c\# \cdot \mathcal{C}$ is the class defined in the following way: A function f is in $c\# \cdot \mathcal{C}$ if and only if there exist a cluster set $A \in \mathcal{C}$ and a polynomial p such that $f(x) = \#\{y | |y| = p(|x|) \wedge \langle x, y \rangle \in A\}$ for every $x \in \Sigma^*$.
2. $\mathrm{cmax} \cdot \mathcal{C}$ is the class defined in the following way: A function f is in $\mathrm{cmax} \cdot \mathcal{C}$ if and only if there exist a cluster set $A \in \mathcal{C}$ and a polynomial p such that $f(x) = \max\{y | |y| = p(|x|) \wedge \langle x, y \rangle \in A\}$ for every $x \in \Sigma^*$. If $\{y | |y| = p(|x|) \wedge \langle x, y \rangle \in A\} = \varnothing$ then we define $f(x) = 0$.
3. $\mathrm{cmin} \cdot \mathcal{C}$ is the class defined in the following way: A function f is in $\mathrm{cmin} \cdot \mathcal{C}$ if and only if there exist a cluster set $A \in \mathcal{C}$ and a polynomial p such that $f(x) = \min\{y | |y| = p(|x|) \wedge \langle x, y \rangle \in A\}$ for every $x \in \Sigma^*$. If $\{y | |y| = p(|x|) \wedge \langle x, y \rangle \in A\} = \varnothing$ then we define $f(x) = 0$.

Clearly, cluster operators are monotonical, and images are a part of the images of the corresponding common operators.

Similar to the characterization $\# \cdot \mathrm{P} = \#\mathrm{P}$ mentioned above we have a result for the operator $c\#\cdot$ that is easy to show.

Proposition 4.3. $c\# \cdot \mathrm{P} = \{\#acc_M | M \text{ is a polynomial time cluster machine}\}$.

The study of structural relationships between function classes produced by the cluster operators in application to certain complexity classes only reflects the behaviour of operators on the lowest level of the polynomial-time hierarchy. Statements about higher levels are easy to derive from the monotonicity of the operators and using relativizations.

From Definition 4.2 and Theorem 3.4 we immediately obtain the following proposition.

Proposition 4.4. $\mathrm{FP} \subseteq (\mathrm{c\#} \cdot \mathrm{P} \cap \mathrm{cmax} \cdot \mathrm{P} \cap \mathrm{cmin} \cdot \mathrm{P})$.

Theorem 4.5 shows remarkable differences in the behaviour of $\mathrm{c\#} \cdot \mathrm{P}, \mathrm{cmax} \cdot \mathrm{P}$ and $\mathrm{cmin} \cdot \mathrm{P}$ related to analogous classes under application of common operators.

Theorem 4.5. *1.* $\mathrm{cmax} \cdot \mathrm{P} = \mathrm{cmin} \cdot \mathrm{P}$.

2. $\mathrm{cmax} \cdot \mathrm{P} \subseteq \mathrm{c\#} \cdot \mathrm{P}$.

Proof. (1) Let $f \in \mathrm{cmax} \cdot \mathrm{P}$ by a cluster set $A \in \mathrm{P}$ and a polynomial p. Define $B = \{\langle x, y\rangle \mid |y| = p(|x|) \wedge (y = 1^{p(|x|)} \rightarrow \langle x, y\rangle \in A) \wedge (y < 1^{p(|x|)} \rightarrow (\langle x, y\rangle \in A \wedge \langle x, y+1\rangle \notin A))\}$. Clearly, $B \in \mathrm{P}$ and for every $x \in \Sigma^*$ there is at most one y with $\langle x, y\rangle \in B$. Hence, B is a cluster set and $f(x) = \min\{y \mid |y| = p(|x|) \wedge \langle x, y\rangle \in B\}$. Analogously, the converse can be proven.

(2) is a simple consequence of the first proof. ❑

Remark 4.6. *In Theorem 4.5 the class* P *can be replaced by every complexity class closed under complementation, intersection and* $\leq_m^P$*-reductions such as by* PP, XP [OH93, FFK94] *or* $\oplus$P.

Using Theorem 3.4 one can conclude a simple statement about the position $\mathrm{c\#} \cdot \mathrm{P}$ with respect to $\mathrm{cmax} \cdot \mathrm{NP}$ and $\mathrm{cmin} \cdot \mathrm{NP}$. The inclusion $\mathrm{c\#} \cdot \mathrm{P} \subseteq \mathrm{c\#} \cdot \mathrm{NP}$ is a direct consequence of the monotonicity of $\mathrm{c\#}\cdot$.

Proposition 4.7. $\mathrm{c\#} \cdot \mathrm{P} \subseteq (\mathrm{c\#} \cdot \mathrm{NP} \cap \mathrm{cmax} \cdot \mathrm{NP} \cap \mathrm{cmin} \cdot \mathrm{NP})$.

The next theorem summarizes the relative positions of the classes $\mathrm{c\#} \cdot \mathrm{NP}, \mathrm{cmax} \cdot \mathrm{NP}$ and $\mathrm{cmin} \cdot \mathrm{NP}$ inside the cluster functions hierarchy. Same results also hold for common function classes. The only deviation is the identity of $\mathrm{cmax} \cdot \mathrm{NP}$ and $\mathrm{c\#} \cdot \mathrm{NP}$ not known there.

Theorem 4.8. *1.* $\mathrm{cmax} \cdot \mathrm{NP} = \mathrm{c\#} \cdot \mathrm{NP}$.

2. $(\mathrm{cmax} \cdot \mathrm{NP} \cup \mathrm{cmin} \cdot \mathrm{NP}) \subseteq \mathrm{F}\Delta_2^P$.

3. $\mathrm{cmax} \cdot \mathrm{NP} \subseteq \mathrm{cmin} \cdot \mathrm{coNP}$.

4. $\mathrm{cmin} \cdot \mathrm{NP} \subseteq \mathrm{cmax} \cdot \mathrm{coNP}$.

5. $\mathrm{cmin} \cdot \mathrm{NP} \subseteq \mathrm{c\#} \cdot \mathrm{coNP}$.

Proof. (1) Let $f \in \mathrm{cmax} \cdot \mathrm{NP}$ by a cluster set $A \in \mathrm{NP}$ and by a polynomial p. Define $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge (\exists z, |z| = p(|x|))(z > y \wedge \langle x, z\rangle \in A)\}$. Obviously, $B \in \mathrm{NP}$ and $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge y < f(x)\}$. Thus, B is a cluster set and $f(x) = \#\{y \| y| = p(|x|) \wedge \langle x, y\rangle \in B\}$. Hence, $f \in \mathrm{c\#} \cdot \mathrm{NP}$. Conversely, let $f \in \mathrm{c\#} \cdot \mathrm{NP}$ by a cluster set $A \in \mathrm{NP}$ and by a polynomial p. Consider a set B defined as $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge (\exists z_1, z_2, |z_1| = |z_2| = p(|x|))(\langle x, z_1\rangle \in A \wedge \langle x, z_2\rangle \in A \wedge y = |z_1 - z_2| + 1)\}$. Since $A \in \mathrm{NP}$, we have $B \in \mathrm{NP}$ and, furthermore, $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge y \le f(x)\}$. Thus, B is a cluster set and $f(x) = \max\{y \| y| = p(|x|) \wedge \langle x, y\rangle \in B\}$. Hence, $f \in \mathrm{cmax} \cdot \mathrm{NP}$.

(2) Let $f \in \mathrm{cmax} \cdot \mathrm{NP}$ by a cluster set $A \in \mathrm{NP}$ and by a polynomial p. If we define a set $B \in \mathrm{NP}$ as $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge (\exists z, |z| = p(|x|))(y \le z \wedge \langle x, z\rangle \in A)\}$, then, for every $x \in \Sigma^*$, the value $f(x)$ can be computed deterministically in polynomial time by binary search using queries to B. Hence, $f \in \mathrm{F}\Delta_2^{\mathrm{P}}$. Analogously, one can show the inclusion for $\mathrm{cmin} \cdot \mathrm{NP}$.

(3) Let $f \in \mathrm{cmax} \cdot \mathrm{NP}$ by a cluster set $A \in \mathrm{NP}$ and by a polynomial p. Define $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge (\forall z, |z| = p(|x|))(\langle x, z\rangle \in A \to z \le y)\}$. Surely, B is a cluster set in coNP with $f(x) = \min\{y \| y| = p(|x|) \wedge \langle x, y\rangle \in B\}$ for every $x \in \Sigma^*$. Hence, $f \in \mathrm{cmin} \cdot \mathrm{coNP}$.

(4) Similar to (3) we can verify this statement using a set B defined as $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge (\forall z, |z| = p(|x|))(\langle x, z\rangle \in A \to y \le z)\}$.

(5) Let $f \in \mathrm{cmin} \cdot \mathrm{NP}$ by a cluster set $A \in \mathrm{NP}$ and by a polynomial p. Define $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge (\forall z, |z| = p(|x|))(\langle x, z\rangle \in A \to y < z)\}$. Clearly, $B \in \mathrm{coNP}$ and $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge y < f(x)\}$. Obviously, $f(x) = \#\{y \| y| = p(|x|) \wedge \langle x, y\rangle \in B\}$. Hence, $f \in \mathrm{c\#} \cdot \mathrm{coNP}$. ❑

From the monotonicity of the operators we obtain the inclusions $\mathrm{c\#} \cdot \mathrm{coNP} \subseteq \mathrm{c\#} \cdot \Delta_2^{\mathrm{P}}$, $\mathrm{cmax} \cdot \mathrm{coNP} \subseteq \mathrm{cmax} \cdot \Delta_2^{\mathrm{P}}$ and $\mathrm{cmin} \cdot \mathrm{coNP} \subseteq \mathrm{cmin} \cdot \Delta_2^{\mathrm{P}}$. Thus next stage of the cluster function hierarchy is reached. An analogous result to $\# \cdot \mathrm{coNP} = \# \cdot \Delta_2^{\mathrm{P}}$ is not known for c#·.

Now we turn towards the question whether we get finer results when more subtle proof techniques are used. We show characterizations that give evidence for non-existence of such results by finding unlike identities of complexity classes. Theorem 4.9 contains inclusions equivalent to UP = P.

Theorem 4.9. *The following statements are equivalent.*

1. UP = P.

2. $\mathrm{c\#} \cdot \mathrm{P} = \mathrm{FP}$.

3. $\mathrm{cmax} \cdot \mathrm{P} = \mathrm{FP}$.

4. $\mathrm{c\#} \cdot \mathrm{P} = \mathrm{cmax} \cdot \mathrm{P}$.

Proof. $(1) \Rightarrow (2)$: Let $f \in \mathrm{c\#} \cdot \mathrm{P}$. By Proposition 4.3 and Theorem 3.4 there exists a nondeterministic polynomial-time Turing machine M computing f weakly onefold. Therefore, consider a set $A \in \mathrm{UP}$ defined as $A = \{\langle x, y\rangle \| y| = p(|x|) \wedge (y = 2^{p(|x|)} \to f(x) = y) \wedge (y < 2^{p(|x|)} \to f(x) > y)\}$. By assumption, $A \in \mathrm{P}$. Using A, for every $x \in \Sigma^*$, the value $f(x)$ can be determined deterministically in polynomial time by binary search. Hence, $f \in \mathrm{FP}$.

$(2) \Rightarrow (3)$ and $(2) \Rightarrow (4)$ are due to $\mathrm{FP} \subseteq \mathrm{cmax} \cdot \mathrm{P} \subseteq \mathrm{c\#} \cdot \mathrm{P}$.

$(3) \Rightarrow (1)$: Let $A \in \mathrm{UP}$, and let M be a machine that accepts A in the sense of UP. Without loss of generality assume that only a path $y > 0$ might be accepting. Define a function f as $f(y) = \max \mathrm{acc}_M(y)$ for every $y \in \Sigma^*$. Clearly, $f \in \mathrm{cmax} \cdot \mathrm{P}$ and, by assumption, $f \in \mathrm{FP}$. Let N be a machine that computes f in deterministic polynomial time. Modify N such that, for every $x \in \Sigma^*$, $N(x)$ is accepting if and only if $f(x) > 0$. Hence, $A \in \mathrm{P}$.

$(4) \Rightarrow (1)$: Let $A \in \mathrm{UP}$, and let M be a machine that accepts A in the sense of UP. Define a function f as $f(x) = \#\mathrm{acc}_M(x)$ for every $x \in \Sigma^*$. Clearly, $f \in \mathrm{c\#} \cdot \mathrm{P}$ and $x \in A \Leftrightarrow f(x) = 1$ and $x \notin A \Leftrightarrow f(x) = 0$. By assumption, $f \in \mathrm{cmax} \cdot \mathrm{P}$. Then, there exist a cluster set $B \in \mathrm{P}$ and a polynomial p such that $f(x) = \max\{y \mid |y| = p(|x|) \wedge \langle x, y\rangle \in B\}$ for every $x \in \Sigma^*$. Define N to be a deterministic polynomial-time Turing machine that checks $\langle x, 0^{p(|x|)-1}1\rangle \in B$, on a given input $x \in \Sigma^*$. Consequently, $x \in A$ if and only if x is accepted by N. Hence, $A \in \mathrm{P}$. ❑

In order to prove Theorem 4.11 we need a characterization of $\mathrm{c\#} \cdot \mathrm{P}$.

Lemma 4.10. $\mathrm{c\#} \cdot \mathrm{P} = \mathrm{cmax} \cdot \mathrm{UP} \cap \mathrm{cmin} \cdot \mathrm{UP}$.

Proof. $\mathrm{c\#} \cdot \mathrm{P} \subseteq (\mathrm{cmax} \cdot \mathrm{UP} \cap \mathrm{cmin} \cdot \mathrm{UP})$ follows directly from Theorem 3.4. Let $f \in \mathrm{cmax} \cdot \mathrm{UP} \cap \mathrm{cmin} \cdot \mathrm{UP}$. Then, there exist cluster sets $C \in \mathrm{UP}$ and $D \in \mathrm{UP}$ and, without loss of generality, a polynomial p such that, for every $x \in \Sigma^*$, $f(x) = \max\{y \mid |y| = p(|x|) \wedge \langle x, y\rangle \in C\} = \min\{y \mid |y| = p(|x|) \wedge \langle x, y\rangle \in D\}$. Since $C \cap D \in \mathrm{UP}$, the function f is weakly onefold computable. Hence, $f \in \mathrm{c\#} \cdot \mathrm{P}$. ❑

Theorem 4.11. *The following statements are equivalent.*

1. $\mathrm{PH} = \mathrm{UP}$.
2. $\mathrm{F}\Delta_2^{\mathrm{P}} = \mathrm{c\#} \cdot \mathrm{P}$.
3. $\mathrm{cmax} \cdot \mathrm{NP} = \mathrm{c\#} \cdot \mathrm{P}$.
4. $\mathrm{cmin} \cdot \mathrm{NP} = \mathrm{c\#} \cdot \mathrm{P}$.

Proof. $(1) \Rightarrow (2)$ can be seen as follows: $\mathrm{c\#} \cdot \mathrm{P} \subseteq \mathrm{F}\Delta_2^{\mathrm{P}} \subseteq \mathrm{cmax} \cdot \Delta_2^{\mathrm{P}} \subseteq (\mathrm{cmax} \cdot \mathrm{UP} \cap \mathrm{cmin} \cdot \mathrm{UP})$. By Lemma 4.10 the latter class coincides with $\mathrm{c\#} \cdot \mathrm{P}$.

$(2) \Rightarrow (3)$ and $(2) \Rightarrow (4)$ are due to $\mathrm{c\#} \cdot \mathrm{P} \subseteq (\mathrm{cmax} \cdot \mathrm{NP} \cup \mathrm{cmin} \cdot \mathrm{NP}) \subseteq \mathrm{F}\Delta_2^{\mathrm{P}}$.

$(3) \Rightarrow (1)$: It is sufficient to show $\mathrm{coNP} \subseteq \mathrm{UP}$. Let $A \in \mathrm{coNP}$, and let M be a machine accepting $\overline{A}$ in the sense of NP. Define N to be the machine that, on input $x \in \Sigma^*$, (a) guesses a path $y \in M(x)$, (b) simulates $M(x)$ along y, (c) outputs 1, if y is rejecting, or outputs 2, if y is accepting, and always accepts. Set $f(x) = \max \mathrm{out}_N^+(x)$ for every $x \in \Sigma^*$. Then, $f \in \mathrm{cmax} \cdot \mathrm{NP}$ and $x \in A \Rightarrow f(x) = 1$ and $x \notin A \Rightarrow f(x) = 2$. By assumption and by Theorem 3.4 there is a non-deterministic polynomial-time Turing machine K that computes f onefold. Modify K such that the only accepting output path is rejecting if $f(x) = 2$ or is accepting if $f(x) = 1$. Thus, K accepts A in the sense of UP.

$(4) \Rightarrow (1)$ can be proven analogously to $(3) \Rightarrow (1)$. Here, we only have to swap the outputs in the definition of the machine M and to define the function f as $f(x) = \min \mathrm{out}_N^+(x)$ for every $x \in \Sigma^*$. ❑

Corollary 4.12. *If* $c\# \cdot \mathrm{P} = c\# \cdot \mathrm{NP}$ *then* $\# \cdot \mathrm{P} = \# \cdot \mathrm{NP}$.

Proof. Consequence of $\# \cdot \mathrm{P} = \# \cdot \mathrm{NP} \Leftrightarrow \mathrm{NP} = \mathrm{UP}$ [KST89] and $\mathrm{cmax} \cdot \mathrm{NP} = c\# \cdot \mathrm{NP}$. ❑

Our next theorem lists collapses equivalent to $\mathrm{NP} = \mathrm{coNP}$.

Theorem 4.13. *The following statements are equivalent.*

1. $\mathrm{NP} = \mathrm{coNP}$.
2. $\mathrm{cmax} \cdot \mathrm{NP} = \mathrm{cmin} \cdot \mathrm{NP} = \mathrm{cmax} \cdot \mathrm{coNP} = \mathrm{cmin} \cdot \mathrm{coNP} = c\# \cdot \mathrm{coNP} = \mathrm{F}\Delta_2^{\mathrm{P}} = \mathrm{cmax} \cdot \Delta_2^{\mathrm{P}} = c\# \cdot \Delta_2^{\mathrm{P}}$.
3. $\mathrm{cmax} \cdot \Delta_2^{\mathrm{P}} = \mathrm{cmax} \cdot \mathrm{coNP}$.
4. $\mathrm{cmin} \cdot \Delta_2^{\mathrm{P}} = \mathrm{cmin} \cdot \mathrm{coNP}$.
5. $\mathrm{F}\Delta_2^{\mathrm{P}} = \mathrm{cmax} \cdot \mathrm{NP}$.
6. $\mathrm{F}\Delta_2^{\mathrm{P}} = \mathrm{cmin} \cdot \mathrm{NP}$.
7. $\mathrm{F}\Delta_2^{\mathrm{P}} \subseteq \mathrm{cmax} \cdot \mathrm{coNP}$.
8. $\mathrm{F}\Delta_2^{\mathrm{P}} \subseteq \mathrm{cmin} \cdot \mathrm{coNP}$.
9. $c\# \cdot \mathrm{coNP} \subseteq \mathrm{cmax} \cdot \mathrm{NP}$.
10. $c\# \cdot \mathrm{coNP} = \mathrm{cmin} \cdot \mathrm{NP}$.
11. $c\# \cdot \mathrm{coNP} \subseteq \mathrm{cmax} \cdot \mathrm{coNP}$.
12. $c\# \cdot \mathrm{coNP} \subseteq \mathrm{cmin} \cdot \mathrm{coNP}$.
13. $\mathrm{cmax} \cdot \mathrm{coNP} \subseteq \mathrm{cmax} \cdot \mathrm{NP}$.
14. $\mathrm{cmax} \cdot \mathrm{coNP} \subseteq \mathrm{cmin} \cdot \mathrm{coNP}$.
15. $\mathrm{cmin} \cdot \mathrm{coNP} = \mathrm{cmax} \cdot \mathrm{NP}$.
16. $\mathrm{cmin} \cdot \mathrm{coNP} \subseteq \mathrm{cmin} \cdot \mathrm{NP}$.
17. $\mathrm{cmin} \cdot \mathrm{coNP} \subseteq \mathrm{cmax} \cdot \mathrm{coNP}$.
18. $\mathrm{cmax} \cdot \mathrm{NP} \subseteq \mathrm{cmin} \cdot \mathrm{NP}$.
19. $\mathrm{cmax} \cdot \mathrm{NP} \subseteq \mathrm{cmax} \cdot \mathrm{coNP}$.
20. $\mathrm{cmin} \cdot \mathrm{NP} \subseteq \mathrm{cmax} \cdot \mathrm{NP}$.
21. $\mathrm{cmin} \cdot \mathrm{NP} \subseteq \mathrm{cmin} \cdot \mathrm{coNP}$.

Proof. $(1) \Rightarrow (2)$: Obviously, from NP = coNP follows cmax · NP = cmax · coNP, cmin · NP = cmin · coNP and cmax · NP = c# · NP = c# · coNP. Moreover, cmax · NP $\subseteq$ (cmin · coNP $\cap$ FΔ_2^P) $\subseteq$ (cmin · coNP $\cup$ FΔ_2^P) $\subseteq$ cmax · $\Delta_2^P \subseteq$ c# · Δ_2^P. Finally, since NP = coNP implies Δ_2^P = NP, we have c# · Δ_2^P = c# · NP = cmax · NP.

Clearly, from (2) we conclude all statements from (3) to (21).

$(3) \Rightarrow (19)$ is due to cmax · NP $\subseteq$ cmax · Δ_2^P.

$(4) \Rightarrow (21)$ is due to cmin · NP $\subseteq$ cmin · Δ_2^P.

$(5) \Rightarrow (21)$ is due to cmin · NP $\subseteq$ FΔ_2^P and cmax · NP $\subseteq$ cmin · coNP.

$(6) \Rightarrow (19)$ and $(7) \Rightarrow (19)$ are due to cmax · NP $\subseteq$ FΔ_2^P.

$(8) \Rightarrow (21)$ are due to cmin · NP $\subseteq$ FΔ_2^P.

$(9) \Rightarrow (21)$ is due to cmin · NP $\subseteq$ c# · coNP and cmax · NP $\subseteq$ cmin · coNP.

$(10) \Rightarrow (11)$, $(14) \Rightarrow (21)$ and $(18) \Rightarrow (19)$ are due to cmin · NP $\subseteq$ cmax · coNP.

$(12) \Rightarrow (21)$ and $(13) \Rightarrow (21)$ are due to cmin · NP $\subseteq$ c# · coNP.

$(16) \Rightarrow (19)$ is due to cmax · NP $\subseteq$ cmin · coNP and cmin · NP $\subseteq$ cmax · coNP.

$(17) \Rightarrow (19)$ and $(20) \Rightarrow (21)$ are due to cmax · NP $\subseteq$ cmin · coNP.

$(11) \Rightarrow (1)$: Let $A \in$ NP, and let M be a machine that accepts A in the sense of NP time-bounded by polynomial p. Modify M such that M outputs the path itself and accepts if M runs along a rejecting path, and outputs non-deterministically each natural number less than the path itself and accepts if M runs along a accepting path. Define $f(x) = \#\{y \mid |y| = p(|x|) \wedge y \notin \mathrm{out}_M^-(x)\}$ for every $x \in \Sigma^*$. Clearly, $f \in$ c# · coNP and thus $f \in$ cmax · coNP by a cluster set $B \in$ coNP and a polynomial q. Define K to be a machine that checks $\langle x, 0^{q(|x|)-1}1\rangle \in B$ on a given input $x \in \Sigma^*$. Thus, $x \in A$ if and only if $f(x) > 0$ if and only if $\#\mathrm{acc}_K(x) = 0$. Hence, $A \in$ coNP.

$(15) \Rightarrow (1)$: Let $A \in$ coNP, and let M be a machine that accepts $\overline{A}$ in the sense of NP. Modify M such that M outputs 0 along rejecting paths and accepts, and, along accepting paths, M outputs non-deterministically both 0 and 1 and accepts always. Define $f(x) = \min\{y \mid |y| = 1 \wedge y \notin \mathrm{out}_M^+(x)\}$ for every $x \in \Sigma^*$. Obviously, $f \in$ cmin · coNP and, thus, $f \in$ cmax · NP by a cluster set $B \in$ NP and by a polynomial q. Define K to be a machine that checks $\langle x, 0^{q(|x|)-1}1\rangle \in B$, on a given input $x \in \Sigma^*$. Thus, if $x \in A$ then $f(x) = 1$ and $\#\mathrm{acc}_K(x) > 0$, and if $x \notin A$ then $f(x) = 0$ and $\#\mathrm{acc}_K(x) = 0$. Hence, $A \in$ NP.

$(19) \Rightarrow (1)$: Let $A \in$ NP, and let M be a machine that accepts A in the sense of NP. Modify M such that M outputs 1 along rejecting paths, and outputs 2 along accepting paths, and accepts always. Define $f(x) = \max \mathrm{out}_M^+(x)$ for every $x \in \Sigma^*$. Trivially, $f \in$ cmax · NP and as supposed $f \in$ cmax · coNP by a cluster set $B \in$ coNP and by a polynomial q. Define K to be a machine that, on input $x \in \Sigma^*$, checks $\langle x, 0^{q(|x|)-2}10\rangle \in B$. Thus, if $x \in A$ then $f(x) = 2$ and $\#\mathrm{acc}_K(x) = 0$, and if $x \notin A$ then $f(x) = 1$ and $\#\mathrm{acc}_K(x) > 0$. Hence, $A \in$ coNP.

$(21) \Rightarrow (1)$: Let $A \in$ coNP, and let M be a machine that accepts $\overline{A}$ in the sense of NP. Modify M such that M outputs 2 along rejecting paths, outputs 1 along accepting paths, and accepts always. Define $f(x) = \min \mathrm{out}_M^+(x)$ for every $x \in \Sigma^*$. Surely, $f \in$ cmin · NP and, thus, $f \in$ cmin · coNP by a cluster set $B \in$ coNP and by a polynomial q. Define K to be a machine that checks $\langle x, 0^{q(|x|)-1}1\rangle \in B$, on a given input $x \in \Sigma^*$. Thus, if $x \in A$ then $f(x) = 2$ and $\#\mathrm{acc}_K(x) > 0$, and if $x \notin A$ then $f(x) = 1$ and $\#\mathrm{acc}_K(x) = 0$. Hence, $A \in$ NP. ❑

The following two theorems show that $c\#\cdot coNP$ probably does not coincide with classes inside $c\# \cdot \Delta_2^P$.

Theorem 4.14. *1. If* $c\# \cdot coNP \subseteq F\Delta_2^P$ *then* $UP^{NP} = \Delta_2^P$.

2. If $c\# \cdot coNP \subseteq cmax \cdot \Delta_2^P$ *then* $UP^{NP} = \Delta_2^P$.

Proof. We will use the monotonical operator technique as applied in [VW93, Vol94, HW97]. Say that for an arbitrary function class $\mathcal{F}$ a set $A \in U \cdot \mathcal{F}$ if and only if there is a function $f \in \mathcal{F}$ such that for every $x \in \Sigma^*$ holds $x \in A \Leftrightarrow f(x) = 1$ and $x \notin A \Leftrightarrow f(x) = 0$. Note that $U \cdot c\# \cdot \mathcal{C} = U \cdot \# \cdot \mathcal{C}$ for every complexity class $\mathcal{C}$. Thus $U \cdot c\# \cdot coNP = U \cdot \# \cdot coNP = U \cdot \# \cdot \Delta_2^P = UP^{NP}$. On the other hand it is easy to see that $U \cdot F\Delta_2^P = U \cdot cmax \cdot \Delta_2^P = \Delta_2^P$. Since $U\cdot$ is monotonical the implications hold. ❑

Theorem 4.15. *The following statements are equivalent:*

1. $c\# \cdot \Delta_2^P = c\# \cdot coNP$.
2. $cmax \cdot \Delta_2^P \subseteq c\# \cdot coNP$.
3. $F\Delta_2^P \subseteq c\# \cdot coNP$.

Proof. $(1) \Rightarrow (2)$ and $(2) \Rightarrow (3)$ are due to $F\Delta_2^P \subseteq cmax \cdot \Delta_2^P \subseteq c\# \cdot \Delta_2^P$.

$(3) \Rightarrow (1)$: Let $f \in c\# \cdot \Delta_2^P$. From relativized Theorem 3.4 we obtain a non-deterministic polynomial-time oracle Turing machine M and an oracle $A \in NP$ such that $\#acc_{M^{(A)}}(x) \leq 1$ and if $\#acc_{M^{(A)}}(x) = 1$ then $f(x) = out^+_{M^{(A)}}(x)$ for every $x \in \Sigma^*$. Define a binary function g as $g(x,y) = out^+_{M^{(A)}}(x)$ if $y \in acc_{M^{(A)}}(x)$ and $g(x,y) = 0$ otherwise. Note that there is a polynomial p such that $\#acc_{M^{(A)}}(x) \leq 2^{p(|x|)}$ for every $x \in \Sigma^*$. Clearly, $g \in F\Delta_2^P$, and hence $g \in c\# \cdot coNP$. Consequently, there exist a cluster set $C \in coNP$ and a polynomial r such that $g(x,y) = \#\{z \mid |z| = r(|x|) \wedge |y| = p(|x|) \wedge \langle x,y,z\rangle \in C\}$ for every $x \in \Sigma^*$. Note that for at most one y holds $g(x,y) > 0$ when x is given. Denote such y by y'. Define a set C' as $C' = \{\langle x, yz\rangle \mid |yz| = r(|x|) + p(|x|) \wedge \langle x,y,z\rangle \in C\}$. Then, $C' \in coNP$, and C' is a cluster set. Define a function g' as $g'(x) = \#\{w \mid |w| = q(|x|) \wedge \langle x,w\rangle \in C'\}$ for every $x \in \Sigma^*$ where $q(|x|) = r(|x|) + p(|x|)$. Then, $g' \in c\#\cdot coNP$ and for every $x \in \Sigma^*$ holds $g'(x) = g(x,y') = f(x)$ if such y' exists or $g'(x) = f(x) = 0$ otherwise. Hence, $f \in c\# \cdot coNP$. ❑

5 Relations to Common Operators

We will compare cluster operators and their corresponding common operators. As already mentioned, images of cluster operators are included in the images of the corresponding common operators. Inverstigations are limited again to the lowest level of the polynomial-time hierarchy.

For our considerations we need two lemmata. These results are easy to conclude from well-known facts, noted in [OH93, Sch89, TO92], by inductive argumentations.

Lemma 5.1. *The following statements are equivalent.*

1. $\mathrm{PP} \subseteq \Sigma_k^{\mathrm{P}}$.
2. $\mathrm{PP}^{\Sigma_{k-1}^{\mathrm{P}}} = \Sigma_k^{\mathrm{P}}$.
3. $\mathrm{CH} = \Sigma_k^{\mathrm{P}}$.

Lemma 5.2. *The following statements are equivalent.*

1. $\mathrm{PP} \subseteq \mathrm{UP}^{\Sigma_k^{\mathrm{P}}}$.
2. $\mathrm{PP}^{\Sigma_k^{\mathrm{P}}} = \mathrm{UP}^{\Sigma_k^{\mathrm{P}}}$.
3. $\mathrm{CH} = \mathrm{UP}^{\Sigma_k^{\mathrm{P}}}$.

Now we can note statements about the behaviour of the operator $\mathrm{c\#}\cdot$ in relation to $\#\cdot$.

Theorem 5.3. 1. $\mathrm{c\#} \cdot \mathrm{P} = \# \cdot \mathrm{P}$ *if and only if* $\mathrm{PP} = \mathrm{UP}$.

2. $\mathrm{c\#} \cdot \mathrm{NP} = \# \cdot \mathrm{NP}$ *if and only if* $\mathrm{PP} = \mathrm{NP}$.
3. *If* $\mathrm{c\#} \cdot \mathrm{coNP} = \# \cdot \mathrm{coNP}$ *then* $\mathrm{PP} \subseteq \mathrm{UP}^{\mathrm{NP}}$.

Proof. (1) Let $A \in \mathrm{PP}$. Then, there exist functions $f \in \# \cdot \mathrm{P}$ and $g \in \mathrm{FP}$ such that $x \in A \Leftrightarrow f(x) \geq g(x)$ for every $x \in \Sigma^*$. Without loss of generality, suppose $f(x) > 0$ for every $x \in \Sigma^*$. By assumption, $f \in \mathrm{c\#} \cdot \mathrm{P}$ and, by Proposition 4.3 and by Theorem 3.4 there exist a non-deterministic polynomial-time Turing machine M that computes f onefold. Modify M such that M accepts along the only accepting path if $f(x) \geq g(x)$. M still runs in polynomial time. Thus, M accepts A in the sense of UP. Conversely, we use the equivalence $\mathrm{PP} = \mathrm{UP} \Leftrightarrow \mathrm{C_{=}P} = \mathrm{UP}$ [OH93]. Let $f \in \# \cdot \mathrm{P}$. Define a set C as $C = \{\langle x, y\rangle \mid |y| = p(|x|) \land f(x) = y\}$. Clearly, $C \in \mathrm{C_{=}P}$ and, by assumption, $C \in \mathrm{UP}$. Define N to be a machine that, on input $x \in \Sigma^*$, (a) guesses a word $y \in \Sigma^*$ with $|y| = p(|x|)$, (b) checks $\langle x, y\rangle \in C$, (c) outputs y and accepts if and only if checking is successful. Obviously, N computes f onefold. From Proposition 4.3 and Theorem 3.4 we have $f \in \mathrm{c\#} \cdot \mathrm{P}$.

(2) Let $A \in \mathrm{PP}^{\mathrm{NP}}$. Thus there exist functions $f \in \#\cdot\mathrm{NP}$ and $g \in \mathrm{FP}$ such that $x \in A \Leftrightarrow f(x) \geq g(x)$ for every $x \in \Sigma^*$ [Vol94, Wag86]. Without loss of generality assume that $f(x) > 0$ for every $x \in \Sigma^*$. By supposition, $f \in \mathrm{c\#} \cdot \mathrm{NP}$ and, by Theorem 4.8, consequently $f \in \mathrm{cmax} \cdot \mathrm{NP}$. Now, there exist a cluster set $B \in \mathrm{NP}$ accepted by a machine M in the sense of NP and a polynomial p such that $f(x) = \max\{y \mid |y| = p(|x|) \land \langle x, y\rangle \in B\}$ for every $x \in \Sigma^*$. Define N to be a machine that, on input $x \in \Sigma^*$, (a) guesses a word $y \in \Sigma^*$ with $|y| = p(|x|)$, (b) simulates M on input $\langle x, y\rangle$, (c) computes deterministically $g(x)$, and (d) accepts if and only if $M(\langle x, y\rangle)$ is accepting along the simulation path and it holds that $y \geq g(x)$. Thus, if $x \in A$ then $f(x) \geq g(x)$ and, consequently, $\#acc_N(x) > 0$, and if $x \notin A$ then $f(x) < g(x)$ and, consequently, $\#acc_N(x) = 0$. Hence, $A \in \mathrm{NP}$. In order to prove the converse consider the following inclusions: $\mathrm{c\#} \cdot \mathrm{NP} \subseteq \# \cdot \mathrm{NP} \subseteq \# \cdot \Delta_2^{\mathrm{P}} \subseteq \mathrm{c\#} \cdot \mathrm{P}^{\#\cdot\Delta_2^{\mathrm{P}}} = \mathrm{c\#} \cdot \mathrm{P}^{\mathrm{PP}^{\mathrm{NP}}} \subseteq \mathrm{c\#} \cdot \mathrm{CH}$. From Lemma 5.1 we have $\mathrm{c\#} \cdot \mathrm{CH} = \mathrm{c\#} \cdot \mathrm{NP}$ and, hence, $\# \cdot \mathrm{NP} = \mathrm{c\#} \cdot \mathrm{NP}$.

(3) From $\# \cdot \Delta_2^{\mathrm{P}} = \# \cdot \mathrm{coNP}$ and from supposition follows $\# \cdot \Delta_2^{\mathrm{P}} = \mathrm{c\#} \cdot \Delta_2^{\mathrm{P}}$. Due to the relativizibility of the proof of (1) we conclude $\mathrm{PP}^{\mathrm{NP}} = \mathrm{UP}^{\mathrm{NP}}$. Lemma 5.2 gives $\mathrm{PP} \subseteq \mathrm{UP}^{\mathrm{NP}}$. ❑

Before we explicitely compare optimization operators we point out an interesting result.

Lemma 5.4. $\mathrm{cmax} \cdot \mathrm{P} = \max \cdot \mathrm{P} \cap \min \cdot \mathrm{P}$.

Proof. Obviously, by Theorem 4.5, $\mathrm{cmax} \cdot \mathrm{P} \subseteq (\max \cdot \mathrm{P} \cap \min \cdot \mathrm{P})$. Let $f \in \max \cdot \mathrm{P} \cap \min \cdot \mathrm{P}$. Then, there exist sets $A \in \mathrm{P}$ and $B \in \mathrm{P}$ and, without loss of generality, one polynomial p such that, for every $x \in \Sigma^*$, $f(x) = \max\{y \| y| = p(|x|) \wedge \langle x, y\rangle \in A\} = \min\{y \| y| = p(|x|) \wedge \langle x, y\rangle \in B\}$. Define $C = \{\langle x, y\rangle \| y| = p(|x|) \wedge \langle x, y\rangle \in A \wedge \langle x, y\rangle \in B\}$. Since $A \in \mathrm{P}$ and $B \in \mathrm{P}$ we have $C \in \mathrm{P}$. Moreover, for every $x \in \Sigma^*$, there is at most one $y \in \Sigma^*$ with $|y| = p(|x|)$ such that $\langle x, y\rangle \in C$. Thus, C is a cluster set in P and $f(x) = \max\{y \| y| = p(|x|) \wedge \langle x, y\rangle \in C\}$ for every $x \in \Sigma^*$. Hence, $f \in \mathrm{cmax} \cdot \mathrm{P}$. ❑

Remark 5.5. *In Lemma 5.4 the class* P *can be replaced by every complexity class closed under complementation, intersection and* $\leq_m^{\mathrm{P}}$*-reductions such as by* PP, XP [OH93, FFK94] *or* $\oplus$P.

The next theorem shows the relation between cmax· and max·.

Theorem 5.6. *1.* $\mathrm{cmax} \cdot \mathrm{P} = \max \cdot \mathrm{P}$ *if and only if* NP = P.

2. $\mathrm{cmax} \cdot \mathrm{NP} = \max \cdot \mathrm{NP}$.

3. If $\mathrm{cmax} \cdot \mathrm{coNP} = \max \cdot \mathrm{coNP}$ *then* $\Sigma_2^{\mathrm{P}} = \Pi_2^{\mathrm{P}}$.

Proof. (1) Let $A \in \mathrm{coNP}$, and let M be a machine that accepts A in the sense of coNP and that has a polynomial runtime p. Define a set B as $B = \{\langle x, 1y\rangle \| y| = p(|x|) \wedge y \in acc_M(x)\} \cup \{\langle x, 0^{p(|x|)}1\rangle | x \in \Sigma^*\}$. Clearly, $B \in \mathrm{P}$. Define a function f as $f(x) = \max\{z \| z| = p(|x|) + 1 \wedge \langle x, z\rangle \in B\}$ for every $x \in \Sigma^*$. Obviously, $f \in \max \cdot \mathrm{P}$ and, by assumption, $f \in \mathrm{cmax} \cdot \mathrm{P}$ by a cluster set $C \in \mathrm{P}$ and a polynomial q. Consider a set $D = \{x | \langle x, 0^{q(|x|)-1}1\rangle \in C \wedge \langle x, 0^{q(|x|)-2}10\rangle \notin C\}$. Surely, $D \in \mathrm{P}$ and $x \in A$ if and only $x \in D$. Hence, $A \in \mathrm{P}$. Conversely, let $f \in \max \cdot \mathrm{P}$ by a set $A \in \mathrm{P}$ and by a polynomial p. Define $B = \{\langle x, y\rangle \| y| = p(|x|) \wedge (\exists z, |z| = p(|x|))(y \leq z \wedge \langle x, z\rangle \in A)\}$. Obviously, $B \in \mathrm{NP}$ and, by assumption, $B \in \mathrm{P}$. Using B, for every $x \in \Sigma^*$ the value $f(x)$ can be determined deterministically in polynomial time by binary search. Hence, $f \in \mathrm{cmax} \cdot \mathrm{P}$.

(2) Let $f \in \max \cdot \mathrm{NP}$ by a set $A \in \mathrm{NP}$ and by a polynomial p. Define a set C as $C = \{\langle x, y\rangle \| y| = p(|x|) \wedge (\exists y_1, y_2, |y_1| = |y_2| = p(|x|))(y_1 \leq y \leq y_2 \wedge \langle x, y_1\rangle \in A \wedge \langle x, y_2\rangle \in A)\}$. Clearly, C is a cluster set in NP. Furthermore, note that $f(x) = \max\{y \| y| = p(|x|) \wedge \langle x, y\rangle \in C\}$ for every $x \in \Sigma^*$. Hence, $f \in \mathrm{cmax} \cdot \mathrm{NP}$.

(3) From the monotonicity of cmax· and from Lemma 5.4 we get $\mathrm{cmax} \cdot \mathrm{coNP} \subseteq \mathrm{cmax} \cdot \Delta_2^{\mathrm{P}} = \max \cdot \Delta_2^{\mathrm{P}} \cap \min \cdot \Delta_2^{\mathrm{P}}$. By assumption, $\max \cdot \mathrm{coNP} \subseteq \min \cdot \Delta_2^{\mathrm{P}}$ and by a result proven in [HW97] this gives $\Sigma_2^{\mathrm{P}} = \Pi_2^{\mathrm{P}}$. ❑

When proving $\mathrm{P} = \mathrm{NP} \Rightarrow \max \cdot \mathrm{P} = \mathrm{cmax} \cdot \mathrm{P}$ we have shown more than necessary. It is easy to see that $\mathrm{P} = \mathrm{NP}$ implies $\max \cdot \mathrm{P} = \mathrm{FP}$. Therefore, and from Lemma 5.4 we obtain immediately the following result (see also [HW97]).

Corollary 5.7. *The following statements are equivalent.*

1. $\mathrm{cmax} \cdot \mathrm{P} = \max \cdot \mathrm{P}$.

2. $\max \cdot \mathrm{P} = \mathrm{FP}$.

3. $\max \cdot \mathrm{P} \subseteq \min \cdot \mathrm{P}$.

With respect to minimization operators same results as Theorem 5.6 and Corollary 5.7 hold.

Theorem 5.8. *1.* $\mathrm{cmin} \cdot \mathrm{P} = \min \cdot \mathrm{P}$ *if and only if* $\mathrm{NP} = \mathrm{P}$.

2. $\mathrm{cmin} \cdot \mathrm{NP} = \min \cdot \mathrm{NP}$.

3. *If* $\mathrm{cmin} \cdot \mathrm{coNP} = \min \cdot \mathrm{coNP}$ *then* $\Sigma_2^{\mathrm{P}} = \Pi_2^{\mathrm{P}}$.

Proof. Similar to Theorem 5.6. For statement (3) we use that $\min \cdot \mathrm{coNP} \subseteq \max \cdot \Delta_2^{\mathrm{P}}$ implies $\Sigma_2^{\mathrm{P}} = \Pi_2^{\mathrm{P}}$ [HW97]. ❑

Corollary 5.9. *The following statements are equivalent.*

1. $\mathrm{cmin} \cdot \mathrm{P} = \min \cdot \mathrm{P}$.

2. $\min \cdot \mathrm{P} = \mathrm{FP}$.

3. $\min \cdot \mathrm{P} \subseteq \max \cdot \mathrm{P}$.

Finally, we note an interesting fact easy to verify from the proof of Lemma 4.10.

Proposition 5.10. $\mathrm{cmax} \cdot \mathrm{UP} \cap \mathrm{cmin} \cdot \mathrm{UP} = \max \cdot \mathrm{UP} \cap \min \cdot \mathrm{UP}$.

Remark 5.11. *In Proposition 5.10 the class* UP *can be replaced by every complexity class closed under intersection.*

6 Open Questions

Open problems particularly concern the completion of results in the cluster function hierarchy. Especially the class $\mathrm{c\#}\cdot\mathrm{coNP}$ seems to be very difficult to deal with. The main open problem should be whether $\mathrm{c\#} \cdot \mathrm{coNP} = \mathrm{c\#} \cdot \Delta_2^{\mathrm{P}}$ is valid. The technique applied in [Köb89, KST89] when proving the analogous result for common operators has no direct counterpart in terms of clusters because this technique does not preserve a given cluster structure in computation trees. There are also a lot of other inclusions not known to be true: (1) $\mathrm{cmax} \cdot \mathrm{NP} \subseteq \mathrm{c\#} \cdot \mathrm{coNP}$, (2) $\mathrm{cmax} \cdot \mathrm{coNP} \subseteq \mathrm{cmin} \cdot \mathrm{NP}$, (3) $\mathrm{cmax} \cdot \mathrm{coNP} \subseteq \mathrm{c\#} \cdot \mathrm{coNP}$, (4) $\mathrm{cmax} \cdot \mathrm{coNP} \subseteq \mathrm{F}\Delta_2^{\mathrm{P}}$, (5) $\mathrm{cmin} \cdot \mathrm{coNP} \subseteq \mathrm{c\#} \cdot \mathrm{coNP}$, and (6) $\mathrm{cmin} \cdot \mathrm{coNP} \subseteq \mathrm{F}\Delta_2^{\mathrm{P}}$. The intuition is that these statements do not hold, i.e. we hope to find structural consequences that make these statements improbable. Moreover, we are interested in characterizations where we only have got implications as yet.

Another interesting direction for research topics could be an exact investigation of cluster sets in some complexity classes. Therefore, one could define the set operator cluster$\cdot$ selecting all cluster sets from a given complexity class. Doubtless this may be successful in order to get finer relations on the side of classes of sets. Then some upward collapses are provable. As an example, if NP $\subseteq$ cluster $\cdot$ NP then PP $=$ NP. This follows from Theorem 5.3. But, how does cluster $\cdot$ NP really look like?

Generally we can detach from the cluster idea and only consider equivalence relations in computation trees. Under which conditions complexity classes (of sets as well as functions) defined by an arbitrary equivalence relation coincide with classes obtained from the cluster relation? Or, what are the connections to the leaf language approach [BCS92, Ver93]? In any case this could also be one way in better understanding the world inside non-determinism.

Acknowledgments. I am very grateful to Gerd Wechsung for supervising my master's thesis, and Harald Hempel for some helpful hints.

References

[All86] E. Allender. The complexity of sparse sets in P. In *Proceedings 1st Structure in Complexity Theory Conference*, volume 223 of *Lecture Notes in Computer Science*, pages 1 – 11. Springer-Verlag, 1986.

[AR88] E. Allender and R. Rubinstein. P-printable sets. *SIAM Journal on Computing*, 17(6):1193 – 1202, 1988.

[BCS92] D. P. Bovet, P. Crescenzi, and R. Silvestri. A uniform approach to define complexity classes. *Theoretical Computer Science*, 104:263–283, 1992.

[BDG90] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II*. Springer-Verlag, 1990.

[BDG95] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, 2nd edition, 1995.

[BLS84] R. V. Book, T. Long, and A. Selman. Quantitative relativizations of complexity classes. *SIAM Journal on Computing*, 13:461–487, 1984.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48:116–148, 1994.

[FGH$^+$96] S. Fenner, F. Green, S. Homer, A. L. Selman, T. Thierauf, and H. Vollmer. Complements of multivalued functions. In *Proceedings 11th Conference on Computational Complexity (1996)*, pages 260 – 269, 1996.

[Gil77] J. Gill. Computational complexity of probabilistic complexity classes. *SIAM Journal on Computing*, 6:675–695, 1977.

[GKR95] W. I. Gasarch, M. W. Krentel, and K. J. Rappoport. OptP as the normal behaviour of NP-complete problems. *Mathematical Systems Theory*, 28:487–514, 1995.

[HH88] J. Hartmanis and L. A. Hemachandra. Complexity classes without machines: on complete languages for UP. *Theoretical Computer Science*, 58:129 – 142, 1988.

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

[HVW96] U. Hertrampf, H. Vollmer, and K. W. Wagner. On balanced vs. unbalanced computation trees. *Mathematical Systems Theory*, 29:411–421, 1996.

[HW97] H. Hempel and G. Wechsung. The operators min and max on the polynomial hierarchy. In *Proceedings 14th Symposium on Theoretical Aspects of Computer Science*, Springer Lecture Notes in Computer Science, pages 93 – 104, 1997.

[Köb89] J. Köbler. *Strukturelle Komplexität von Anzahlproblemen*. PhD thesis, Universität Stuttgart, Fakultät für Informatik, 1989.

[Kos96] S. Kosub. Clustermaschinen. Master's thesis, Friedrich-Schiller-Universität Jena, Fakultät für Mathematik und Informatik, 1996.

[Kre88] M. W. Krentel. The complexity of optimization functions. *Journal of Computer and System Sciences*, 36:490–509, 1988.

[Kre92] M. W. Krentel. Generalizations of OptP to the polynomial hierarchy. *Theoretical Computer Science*, 97:183–198, 1992.

[KST89] J. Köbler, U. Schöning, and J. Torán. On counting and approximation. *Acta Informatica*, 26:363–379, 1989.

[MS72] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proceedings 13th Symposium on Switching and Automata Theory*, pages 125–129. IEEE Computer Society Press, 1972.

[OH93] M. Ogiwara and L. Hemachandra. A complexity theory of feasible closure properties. *Journal of Computer and System Sciences*, 46:295–325, 1993.

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

[PPST83] W. J. Paul, N. Pippenger, E. Szemeredi, and W. T. Trotter. On determinism versus non-determinism and related problems. In *Proceedings 24th Foundations of Computer Science Conference*, pages 429 – 438, 1983.

[Sav70] W. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177 – 192, 1970.

[Sch89] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, 39:84–100, 1989.

[Sel94] A. L. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Sciences*, 48:357 – 381, 1994.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[TO92] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial time hierarchy. *SIAM Journal on Computing*, 21:315–328, 1992.

[Tod90a] S. Toda. The complexity of finding medians. In *Proceedings 31st Symposium on Foundations of Computer Science*, pages 778–787. IEEE Computer Society Press, 1990.

[Tod90b] S. Toda. *Computational complexity of counting complexity classes*. PhD thesis, Tokyo Institute of Technology, Department of Computer Science, 1990.

[Val76] L. G. Valiant. Relative complexity of checking and evaluation. *Information Processing Letters*, 5:20–23, 1976.

[Val79a] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189 – 201, 1979.

[Val79b] L. G. Valiant. The complexity of enumeration and reliabilty problems. *SIAM Journal of Computing*, 8(3):411–421, 1979.

[Ver93] N. K. Vereshchagin. Relativizable and non-relativizable theorems in the polynomial theory of algorithms. *Izvestija Rossijskoj Akademii Nauk*, 57:51–90, 1993. In Russian.

[Vol94] H. Vollmer. *Komplexitätsklassen von Funktionen*. PhD thesis, Universität Würzburg, Institut für Informatik, Germany, 1994.

[VW93] H. Vollmer and K. W. Wagner. The complexity of finding middle elements. *International Journal of Foundations of Computer Science*, 4:293–307, 1993.

[VW95] H. Vollmer and K. W. Wagner. Complexity classes of optimization functions. *Information and Computation*, 120:198–219, 1995.

[Wag86] K. W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

[Wra77] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.