On the Connection between Interval Size Functions and Path Counting*

Evangelos Bampas[1], Andreas-Nikolas Göbel[1], Aris Pagourtzis[1], and Aris Tentes[2]

[1] School of Elec. & Comp. Eng., National Technical University of Athens
Polytechnioupoli Zografou, 157 80 Athens, Greece
ebamp@cs.ntua.gr, agob@corelab.ntua.gr, pagour@cs.ntua.gr
[2] New York University, USA
tentes@cs.nyu.edu

Abstract. We investigate the complexity of hard counting problems that belong to the class #P but have easy decision version; several well-known problems such as #Perfect Matchings, #DNFSat share this property. We focus on classes of such problems which emerged through two disparate approaches: one taken by Hemaspaandra *et al.* [1] who defined classes of functions that count the size of intervals of ordered strings, and one followed by Kiayias *et al.* [2] who defined the class TotP, consisting of functions that count the total number of paths of NP computations. We provide inclusion and separation relations between TotP and interval size counting classes, by means of new classes that we define in this work. Our results imply that many known #P-complete problems with easy decision are contained in the classes defined in [1]—but are unlikely to be complete for these classes under certain types of reductions. We also define a new class of interval size functions which strictly contains FP and is strictly contained in TotP under reasonable complexity-theoretic assumptions. We show that this new class contains some hard counting problems.

1 Introduction

Valiant's pioneering work on counting problems associated with NP computations [3] revealed the existence of functions that are quite hard to compute exactly (#P-complete), despite the fact that deciding whether the function value is nonzero is easy (in P). This category contains the problem of evaluating the permanent of a 0-1 matrix (Permanent), which is equivalent to counting perfect matchings in bipartite graphs (#PM), the problem of counting satisfying assignments to monotone Boolean formulae in 2-CNF form (#Mon2Sat), and many more [4]. A common feature of all these problems is that their #P-completeness property is based on the Cook (poly-time Turing) reduction which blurs structural differences between complexity classes; for example, Permanent is also

* Research supported in part by a Basic Research Support Grant (ΠΕΒΕ 2007) of the National Technical University of Athens.

J. Chen and S.B. Cooper (Eds.): TAMC 2009, LNCS 5532, pp. 108–117, 2009.

complete in the Cook sense for the whole counting version of the Polynomial Hierarchy [5,6], but also for subclasses of #P [7]. Hence, #P is not considered to be the most appropriate class to describe the complexity of these problems.

During the last twenty years there has been constant interest for identifying subclasses of #P that contain hard counting problems with easy decision version [1,2,8,9,10,11,12] and may therefore be more adequate to describe their complexity. In this paper we investigate the relation among subclasses of #P defined and studied through two independent lines of research: (a) classes $\mathrm{IF}_{\mathrm{p}}^{\prec}$ and $\mathrm{IF}_{\mathrm{t}}^{\prec}$ [1] that consist of functions that count the size of intervals of strings under poly-time decidable partial or total (resp.) orders equipped with efficient adjacency checks, and (b) the class TotP [2] that consists of functions that count the total number of paths of NPTMs, and the class #PE [11] that contains all functions of #P for which telling whether the function value is nonzero is easy (in P). Since it is clear from properties of $\mathrm{IF}_{\mathrm{p}}^{\prec}$ shown in [1] that $\mathrm{IF}_{\mathrm{p}}^{\prec} = \#\mathrm{PE}$ we turn our focus to the relation between $\mathrm{IF}_{\mathrm{t}}^{\prec}$ and TotP, which are subclasses of $\mathrm{IF}_{\mathrm{p}}^{\prec}$. To this end we define new interval size function classes by replacing efficient adjacency checks with other suitable feasibility constraints. Our results can be summarized as follows (see also Figure 1):

- TotP is equal to $\mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}}$, that is, to the class of interval size functions defined on total p-orders with efficiently computable lexicographically nearest function.
- $\mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}}$, hence also TotP, is contained in $\mathrm{IF}_{\mathrm{t}}^{\prec}$. The inclusion is strict unless $\mathrm{P} = \mathrm{UP} \cap \mathrm{coUP}$. This, among others, implies that several problems that lie in TotP are unlikely to be $\mathrm{IF}_{\mathrm{t}}^{\prec}$-complete via reductions under which TotP is closed downwards (for example, under Karp reductions); in particular, the class of problems that reduce to #MonSat by such reductions is strictly contained in $\mathrm{IF}_{\mathrm{t}}^{\prec}$ unless $\mathrm{P} = \mathrm{UP} \cap \mathrm{coUP}$. This partially answers an open question posed in [1].
- One of our new classes, namely $\mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$, lies between FP and $\mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}} = \mathrm{TotP}$. We show that $\mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$ contains hard counting problems: we define $\#\mathrm{SAT}_{+2^n}$, which is #P-complete under Cook reductions, and prove that it lies in $\mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$. We also show that any #P function can be obtained by subtracting a function in FP from a function in $\mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$. Therefore $\mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$ is Cook-interreducible with TotP, $\mathrm{IF}_{\mathrm{t}}^{\prec}$, $\mathrm{IF}_{\mathrm{p}}^{\prec} = \#\mathrm{PE}$, and #P but not Karp-interreducible with any of these classes under reasonable assumptions.

2 Definitions–Preliminaries

In the following we assume a fixed alphabet Σ, conventionally $\Sigma = \{0, 1\}$. The symbol $\Sigma^\star$ denotes the set of all finite strings over the alphabet Σ. The length of a string $x \in \Sigma^\star$ is denoted by $|x|$. If S is a set, $\|S\|$ denotes the cardinality of S.

A binary relation over $\Sigma^\star$ is a *partial order* if it is reflexive, antisymmetric, and transitive. A partial order A is a *total order* if for any $x, y \in \Sigma^\star$, it holds that $(x, y) \in A$ or $(y, x) \in A$. An order A is called a *p-order* if there exists a bounding polynomial p such that for all $(x, y) \in A$ it holds that $|x| \leq p(|y|)$.

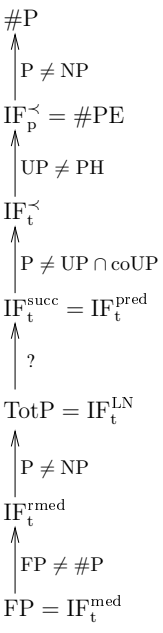

Fig. 1. Inclusions among interval size function classes. Next to each arrow appear the assumptions under which the inclusions are proper; it is open whether $\mathrm{TotP} = \mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}}$ implies an unlikely collapse.

Definition 1 (Notation for orders, cf. [1]). *For any order A we will use the following notation:*

1. $x \leq_A y$ *is equivalent to* $(x, y) \in A$,
2. $x <_A y$ *is equivalent to* $(x \leq_A y \wedge x \not\equiv y)$,
3. $x \prec_A y$ *is equivalent to* $(x <_A y \wedge \neg\exists z \in \Sigma^\star (x <_A z <_A y))$ *(we say that x is the* predecessor *of y, or y is the* successor *of x),*
4. $A_\prec \stackrel{\text{def}}{=} \{(x, y) : x \prec_A y\}$, *and*
5. $(x, y)_A \stackrel{\text{def}}{=} \{z \in \Sigma^\star : x <_A z <_A y\}$ *($(x, y)_A$ will be called an* interval*, even if A is a partial order). We will also use $[x, y]_A$, $[x, y)_A$, and $(x, y]_A$ for the* closed*,* right-open*, and* left-open intervals *respectively.*

We will use lex *to denote the standard lexicographic order of the strings in $\Sigma^\star$.*

Remark 1. For any p-order A with bounding polynomial p and any $y \in \Sigma^\star$, $\|\{x : x \leq_A y\}\| \leq 2^{p(|y|)+1} - 1$. As a corollary, every p-order has a minimal element.

Definition 2 (Notation for total orders). *For any total order A we will use the following notation:*

1. $\mathrm{succ}_A : \Sigma^\star \to \Sigma^\star$ *is the* successor function *for A,*
2. $\mathrm{pred}_A : \Sigma^\star \to \Sigma^\star$ *is the* predecessor function *for A (if A contains a bottom element,* pred_A *is undefined for that element),*

3. $\mathrm{med}_A : \Sigma^\star \times \Sigma^\star \to \Sigma^\star$ *is the* median function *for* A*, defined recursively as follows:*
   - *if* $y <_A x$ *then* $\mathrm{med}_A(x, y)$ *is undefined,*
   - *otherwise if* $x = y$ *or* $x \prec_A y$ *then* $\mathrm{med}_A(x, y) = y$*,*
   - *otherwise* $\mathrm{med}_A(x, y) = \mathrm{med}_A(\mathrm{succ}_A(x), \mathrm{pred}_A(y))$.
4. $\mathrm{LN}_A : \Sigma^\star \times \Sigma^\star \times \Sigma^\star \to \Sigma^\star$ *is the* lexicographically nearest *function for* A*:* $\mathrm{LN}_A(x, y, z)$ *is the string* $w \in [x, y]_A$ *such that* w *is as close to* z *as possible in the lexicographic order (breaking ties arbitrarily).*
5. $\mathrm{rmed}^c_A : \Sigma^\star \times \Sigma^\star \to \Sigma^\star$*,* $c \in \left(0, \frac{1}{2}\right]$*, is some* relaxed median *function for* A*, that satisfies the following properties:*
   - *if* $y <_A x$ *then* $\mathrm{rmed}^c_A(x, y)$ *is undefined,*
   - *otherwise if* $x = y$ *or* $x \prec_A y$ *then* $\mathrm{rmed}^c_A(x, y) = y$*,*
   - *otherwise* $\mathrm{rmed}^c_A(x, y)$ *is a string* $z \in (x, y)_A$ *such that* $\|[x, z)_A\| \geq \lceil c \cdot \|[x, y]_A\| \rceil$ *and* $\|[z, y]_A\| \geq \lfloor c \cdot \|[x, y]_A\| \rfloor$.

Remark 2. For a total order A, we will say that $\mathrm{rmed}_A \in \mathrm{FP}$ if there is *some* $c \in \left(0, \frac{1}{2}\right]$ such that *some* relaxed median function $\mathrm{rmed}^c_A \in \mathrm{FP}$. Observe that med_A is a function that satisfies the properties of $\mathrm{rmed}^{\frac{1}{2}}_A$, therefore if $\mathrm{med}_A \in \mathrm{FP}$ then also $\mathrm{rmed}_A \in \mathrm{FP}$.

We say that an order A is P*-decidable* if $A \in \mathrm{P}$, and we say that it has *efficient adjacency checks* if $A_\prec \in \mathrm{P}$. We also say that a function $f \in \mathrm{FP}$ is FP-computable.

We say that a function $f : \Sigma^\star \to \mathbb{N}$ is an *interval size function defined on an order* A if there exist *boundary functions* $b, t : \Sigma^\star \to \Sigma^\star$ such that for all $x \in \Sigma^\star$, $f(x) = \|(b(x), t(x))_A\|$. In the following, we will primarily be concerned with interval size functions defined on P-decidable p-orders via polynomial-time computable boundary functions.

Definition 3 (Hemaspaandra et al. [1])
$\mathrm{IF}^\prec_\mathrm{p}$ *(*$\mathrm{IF}^\prec_\mathrm{t}$*) is the class of interval size functions defined on* P*-decidable partial (total) p-orders with efficient adjacency checks via polynomial-time computable boundary functions.*

Remark 3. Note that in [1], $\mathrm{IF}^\prec_\mathrm{p}$ and $\mathrm{IF}^\prec_\mathrm{t}$ were called IF_p and IF_t, respectively. We will use superscript in order to distinguish these classes from other classes that we will define below.

Furthermore, we will be interested in interval size functions defined on P-decidable p-orders with various other feasibility constraints, apart from $A_\prec \in \mathrm{P}$. We define the following classes:

Definition 4. $\mathrm{IF}^\mathrm{succ}_\mathrm{t}$ *(resp.* $\mathrm{IF}^\mathrm{pred}_\mathrm{t}$*,* $\mathrm{IF}^\mathrm{LN}_\mathrm{t}$*,* $\mathrm{IF}^\mathrm{rmed}_\mathrm{t}$*,* $\mathrm{IF}^\mathrm{med}_\mathrm{t}$*) is the class of interval size functions each of which is defined on some* P*-decidable total p-order* A *via polynomial-time computable boundary functions, where in addition* $\mathrm{succ}_A \in \mathrm{FP}$ *(resp.* pred_A*,* LN_A*,* rmed_A*,* $\mathrm{med}_A \in \mathrm{FP}$*).*

The computational model we are going to use is the Non-deterministic Polynomial-time Turing Machine (NPTM). For an NPTM M we denote with $M(x)$ the computation of M on input x. We say that M is in normal form if we can represent the computation $M(x)$ with a full, complete binary tree of depth exactly $p(|x|)$, where p is the polynomial that bounds the running time of M.

Valiant in [4] defines as #P the class of all total functions f for which there exists an NPTM M such that for all x, $f(x)$ is the number of accepting paths of $M(x)$.

In [11] the class #PE is defined as the class of #P functions with their underlying language in P, where for a function f, its underlying language is defined to be the language $L_f = \{x \mid f(x) > 0\}$. In [2] the class TotP is defined as the class that contains the functions f for which there exists an NPTM M such that for all x, $f(x)$ is the number of the computation paths of the computation of $M(x)$ minus one. The functions of TotP are usually denoted with $\mathrm{tot}_M(x)$, where M is the associated NPTM, and x the input. In [12] TotP is proven to be exactly the closure under Karp (parsimonious) reduction of the set of self-reducible functions of #PE. The results can be summarized by the following chain of inclusions:

$$\mathrm{FP} \subseteq \mathrm{TotP} \subseteq \#\mathrm{PE} \subseteq \#\mathrm{P} \ ,$$

where all the inclusions are proper unless P = NP.

In [1] it is (implicitly) proven that $\#\mathrm{PE} = \mathrm{IF}_{\mathrm{p}}^{\prec}$, and furthermore that:

$$\mathrm{FP} \subseteq \mathrm{IF}_{\mathrm{t}}^{\prec} \subseteq \mathrm{IF}_{\mathrm{p}}^{\prec} \subseteq \#\mathrm{P} \ .$$

Again the inclusions are proper unless unlikely complexity class collapses occur.

Definition 5. *Polynomial-time reductions between functions:*

- *Cook (poly-time Turing):* $f \leq_T^p g : f \in \mathrm{FP}^g$.
- *Karp (parsimonious):* $f \leq_m^p g : \exists\, h \in \mathrm{FP}, \forall x\ f(x) = g(h(x))$.

Proposition 1. *Every interval size function class $\mathcal{F}$ that contains functions defined via polynomial-time boundary functions is downward closed under Karp reductions.*

Proof. Consider $f \in \mathcal{F}$ via an arbitrary order A and boundary functions $b, t \in \mathrm{FP}$. That is, for every x, $f(x) = \|(b(x), t(x))_A\|$. Assume also that $g \leq_m^p f$, that is $\exists h \in \mathrm{FP}$ such that $\forall x, g(x) = f(h(x))$. This implies that $g(x) = f(h(x)) = \|(b(h(x)), t(h(x)))_A\|$, therefore $g \in \mathcal{F}$ via the same order A and boundary functions $b' = b \circ h \in \mathrm{FP}$ and $t' = t \circ h \in \mathrm{FP}$. □

3 The *status quo* between TotP and $\mathrm{IF}_{\mathrm{t}}^{\prec}$

As we have seen in the previous section, both TotP and $\mathrm{IF}_{\mathrm{t}}^{\prec}$ are contained in $\#\mathrm{PE} = \mathrm{IF}_{\mathrm{p}}^{\prec}$. In this section we will investigate the relationship between these two classes. Namely we will show that $\mathrm{TotP} \subseteq \mathrm{IF}_{\mathrm{t}}^{\prec}$, and that the inclusion is proper unless $\mathrm{P} = \mathrm{UP} \cap \mathrm{coUP}$.

Theorem 1. $\mathrm{TotP} \subseteq \mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}} \subseteq \mathrm{IF}_{\mathrm{t}}^{\prec}$.

Proof (sketch). Intuitively, given a path encoding of a TotP computation tree, it is easy to find the next one. The idea is to map computation path encodings to appropriately ordered strings. The detailed proof will appear in the full version. □

We now proceed to show that $\mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}}$, and therefore also TotP, is strictly contained in $\mathrm{IF}_{\mathrm{t}}^{\prec}$, under the assumption that $\mathrm{P} \neq \mathrm{UP} \cap \mathrm{coUP}$. We need a new definition and a couple of lemmata.

Definition 6. *For any constant $k \geq 0$, we define the operator $\mathcal{C}_{>k}$. If $\mathcal{F}$ is any function class, then $\mathcal{C}_{>k} \cdot \mathcal{F}$ defines the following class of languages:*

$$\mathcal{C}_{>k} \cdot \mathcal{F} = \{L \mid \exists f \in \mathcal{F} \ \forall x \, (x \in L \iff f(x) > k)\} \ .$$

Remark 4. Observe that $\mathcal{C}_{>0}\cdot$ coincides with the $\exists\cdot$ operator used by Hemaspaandra et al. in [1], which in turn coincides with the Sig$\cdot$ operator defined by Hempel and Wechsung in [13].

Lemma 1. $\mathrm{UP} \cap \mathrm{coUP} \subseteq \mathcal{C}_{>1} \cdot \mathrm{IF}_{\mathrm{t}}^{\prec}$.

Proof. Let $L \in \mathrm{UP} \cap \mathrm{coUP}$, so there is an NPTM M that decides L with the property that, for any input x, M has exactly one decisive path (either accepting or rejecting) and all the other paths output "?". We assume that M is normalized so that its computation for any input x is a full complete binary tree in which all computation paths have length exactly $p(|x|)$, where p is the polynomial that bounds the running time of M.

We construct an order A that coincides with the lexicographic order of $\Sigma^\star$, except that for every $x \in \Sigma^\star$ the interval between $x0^{p(|x|)+2}$ and $x1^{p(|x|)+2}$ (inclusive) is ordered in the following way:

- First comes $x0^{p(|x|)+2}$,
- if $x \notin L$ next comes $x01z$, where z encodes the unique rejecting path of M on input x, while if $x \in L$ next come $x01z$ and $x10z$, where z encodes the unique accepting path of M on input x,
- next comes $x110^{p(|x|)}$,
- and last come the rest of the strings of the form xw, where $|w| = p(|x|) + 2$, in the lexicographic order.

It is easy to see that A is a p-order with efficient adjacency checks. We define the boundary functions $b, t \in \mathrm{FP}$: for any $x \in \Sigma^\star$, $b(x) = x0^{p(|x|)+2}$ and $t(x) = x110^{p(|x|)}$. It holds that, for any $x \in \Sigma^\star$, $\|(b(x), t(x))_A\| > 1$ if and only if $x \in L$. Therefore, $L \in \mathcal{C}_{>1} \cdot \mathrm{IF}_{\mathrm{t}}^{\prec}$. □

Lemma 2. $\mathcal{C}_{>1} \cdot \mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}} = \mathrm{P}$.

Proof. For any $f \in \mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}}$, we can decide in polynomial time whether for a given x, $f(x) > 1$ or not. Just compute $\mathrm{succ}_A(b(x))$ and $\mathrm{succ}_A(\mathrm{succ}_A(b(x)))$,

where A is the underlying total p-order with $\mathrm{succ}_A \in \mathrm{FP}$ and $b, t \in \mathrm{FP}$ the boundary functions for f. If any of the computed strings is equal to $t(x)$ then reject, else accept. Therefore, $\mathcal{C}_{>1} \cdot \mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}} \subseteq \mathrm{P}$. For the other direction, note that $\mathrm{P} = \mathcal{C}_{>1} \cdot \mathrm{FP} \subseteq \mathcal{C}_{>1} \cdot \mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}}$. □

Theorem 2. *If* $\mathrm{IF}_{\mathrm{t}}^{\prec} = \mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}}$*, then* $\mathrm{P} = \mathrm{UP} \cap \mathrm{coUP}$.

Proof. Assuming that $\mathrm{IF}_{\mathrm{t}}^{\prec} = \mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}}$, from Lemma 1 and Lemma 2 we get that $\mathrm{UP} \cap \mathrm{coUP} \subseteq \mathcal{C}_{>1} \cdot \mathrm{IF}_{\mathrm{t}}^{\prec} = \mathcal{C}_{>1} \cdot \mathrm{IF}_{\mathrm{t}}^{\mathrm{succ}} = \mathrm{P}$. □

4 TotP as an Interval Size Function Class

In this section we prove that TotP coincides with the class of interval size functions defined on orders with polynomial-time computable *lexicographically nearest* functions. To this end, we will employ two variations of the LN function and show a useful property of them.

Definition 7. *For a p-order A we define the following partial functions:*

1. $\mathrm{LN}_A^+(u, v, x)$ *is the lexicographically smallest* $y \in [u, v]_A$ *such that* $x \leq_{\mathrm{lex}} y$.
2. $\mathrm{LN}_A^-(u, v, x)$ *is the lexicographically largest* $y \in [u, v]_A$ *such that* $x \geq_{\mathrm{lex}} y$.

Lemma 3. *For a total p-order A, if* $\mathrm{LN}_A \in \mathrm{FP}$ *then also* $\mathrm{LN}_A^+ \in \mathrm{FP}$ *and* $\mathrm{LN}_A^- \in \mathrm{FP}$.

Proof. We will prove the claim for LN_A^+ only; the proof for LN_A^- is symmetric. Let p be the bounding polynomial of A. We will compute $\mathrm{LN}_A^+(u, v, x)$. Let $y = \mathrm{LN}_A(u, v, x)$. If $x \leq_{\mathrm{lex}} y$, then $\mathrm{LN}_A^+(u, v, x) = y$. For the rest of the proof, assume that $y <_{\mathrm{lex}} x$ and let $\delta = \|[y, x)_{\mathrm{lex}}\|$.

We compute a sequence of strings $x = x_0 <_{\mathrm{lex}} x_1 <_{\mathrm{lex}} \ldots <_{\mathrm{lex}} x_k$ where for all i, $\|[y, x_i)_{\mathrm{lex}}\| = 2^i \cdot \delta$, and k is the smallest index such that $\mathrm{LN}_A(u, v, x_k) \neq y$. It is clear that for all i, $[u, v]_A \cap (y, x_i)_{\mathrm{lex}} = \emptyset$, therefore $\mathrm{LN}_A^+(u, v, x) = \mathrm{LN}_A^+(u, v, x_k) = \mathrm{LN}_A(u, v, x_k)$. If, during this process, we reach some x_j such that $|x_j| > p(|v|)$, then for all $w \geq_{\mathrm{lex}} x_j$ we have $|w| \geq |x_j| > p(|v|)$, which implies that $w >_A v$. So we can safely conclude that $[u, v]_A$ contains no string lexicographically larger than x and halt the computation leaving $\mathrm{LN}_A^+(u, v, x)$ undefined. Note that the size of $[y, x_i)_{\mathrm{lex}}$ is doubled after each iteration, therefore the length of x_i will exceed $p(|v|)$ after at most $\mathcal{O}(p(|v|))$ iterations. □

Theorem 3. $\mathrm{TotP} = \mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}}$.

Proof. We first prove that $\mathrm{TotP} \subseteq \mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}}$. The intuition behind it is that given a TotP computation $M(x)$ and a string z, we can efficiently find a computation path, the encoding of which is lexicographically closest to z.

Let f be a TotP function, i.e. there exists an NPTM M such that on all $x \in \Sigma^\star$, $f(x) = \mathrm{tot}_M(x)$. We assume that all paths of $M(x)$ are of length exactly $p(|x|)$.

We define a total order A on $\Sigma^\star$ as follows: A coincides with the lexicographic order except that, for every $x \in \Sigma^\star$, the interval between $x00^{p(|x|)+1}$ and $x10^{p(|x|)+1}$ (inclusive) is ordered in the following way: first comes $x00^{p(|x|)}0$, next come the elements of $\{x0y0 \mid |y| = p(|x|) \wedge y \text{ encodes a path of } M(x)\}$, in lexicographic order, next comes $x10^{p(|x|)}0$, and last come the elements of $\{x0y0 : |y| = p(|x|) \wedge y \text{ does not encode a path of } M(x)\} \cup \{x0y1 : |y| = p(|x|)\}$, in lexicographic order.

We will show that $\mathrm{LN}_A(u, v, z)$ can be computed in polynomial time. If $z \in [u, v]_A$ then $\mathrm{LN}_A(u, v, z) = z$. If $z \notin [u, v]_A$ we distinguish among three cases:

Case 1. Let $u = x0y_u0$ and let $v = x0y_v0$, where both y_u, y_v encode paths in $M(x)$, and let $z = x0y0$, for some y, $|y| = p(|x|)$, where y does not encode a path in $M(x)$. Let also $y_u <_{\mathrm{lex}} y <_{\mathrm{lex}} y_v$ (if not, the output is u or v). We simulate $M(x)$ following the non-deterministic choices according to the bits of y, until we encounter a choice that is not available. Assume without loss of generality that this choice is '1'. Then we follow the available choice, '0', and we continue the simulation by choosing '1' whenever this is available. This way we obtain the "rightmost" computation path of $M(x)$ which is lexicographically smaller than y, call it y'. Then by following a standard procedure we obtain the "leftmost" path of $M(x)$ which is lexicographically larger than y, call it y''. Return the lexicographically closest to y between y' and y''.

Case 2. Let $u = x0y_ua_u$ and $v = x0y_va_v$, where y_u (y_v) encodes a path in $M(x)$ and $a_u = 1$ ($a_v = 1$), or y_u (y_v) is of length $p(|x|)$ and $a_u \in \Sigma$ ($a_v \in \Sigma$). And, furthermore let $z = z0y0$, where y encodes a computation path of $M(x)$, and $y_u <_{\mathrm{lex}} y <_{\mathrm{lex}} y_v$ (if not, the output is u or v). Return $x0y1$.

Case 3. The remaining cases are either trivial or can be dealt with by combining techniques used for the above two cases. Details are left for the full version.

We now give a sketch of the proof for the inclusion $\mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}} \subseteq \mathrm{TotP}$. Let f be an $\mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}}$ function, via a total p-order $A \in \mathrm{P}$ with bounding polynomial p and boundary functions $b, t \in \mathrm{FP}$. By definition, $\mathrm{LN}_A \in \mathrm{FP}$, therefore by Lemma 3 we have that $\mathrm{LN}_A^+ \in \mathrm{FP}$ and $\mathrm{LN}_A^- \in \mathrm{FP}$.

We outline the operation of an NPTM N that, on input x, performs a computation with exactly $\|(b(x), t(x))_A\| + 1$ computation paths. It first computes $b(x)$ and $t(x)$ and halts if $b(x) \prec_A t(x)$, otherwise it branches into two paths: one of them is a dummy path that halts immediately, and the other one runs a recursive procedure that accepts as input two strings u, v which satisfy the conditions $u <_{\mathrm{lex}} v$ and $u, v \in [b(x), t(x)]_A$. This procedure first computes $z = \mathrm{med}_{\mathrm{lex}}(u, v)$, $z^+ = \mathrm{LN}_A^+(b(x), t(x), z)$, and $z^- = \mathrm{LN}_A^-(b(x), t(x), z)$. It then branches into either one or two paths that halt immediately, depending on whether $z^- = z^+$ or not. Furthermore, it branches into two recursive calls of this procedure with inputs (u, z^-) and (z^+, v), respectively. The effect of this procedure, when initially called with inputs $u = \mathrm{LN}_A(b(x), t(x), \varepsilon)$ and $v = \mathrm{LN}_A(b(x), t(x), 0^{p(|t(x)|)+1})$ (that is, the lexicographically smallest and largest string in $[b(x), t(x)]_A$, respectively) is to output exactly $\|(b(x), t(x))_A\|$ computation paths. We omit the details of how to avoid branching into a recursive call that would have to count

the strings of an empty interval, but it should be clear that it is possible to check if the upcoming procedure call will have to count zero strings or more *before* the machine actually branches into it. □

The above result, combined with the fact that TotP contains all problems in #PE which possess a natural self-reducibility property [12], implies that a number of known problems are contained in $\mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}}$. Actually, Theorem 3 from [12] can be restated as follows:

Corollary 1. *The problems* #DNFSat*,* #MonSat*,* NonNegative Permanent*,* #Perfect Matchings*,* Ranking *are* $\mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}}$*-complete under Cook-1 reductions.*

Remark 5. Note that in [12] it was shown that #Mon2Sat is in TotP but the proof can be easily adapted to show that #MonSat is in TotP as well. In fact, by slightly extending a property shown in [1], namely that it is easy to find the least satisfying assignment that is lexicographically greater than a given assignment, it is possible to show directly that #MonSat is in $\mathrm{IF}_{\mathrm{t}}^{\mathrm{LN}}$.

5 Inside TotP

In this section we give a characterization of FP as an interval size function class, and show that $\mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$ is a class that contains FP and is contained in TotP.

Theorem 4. $\mathrm{FP} = \mathrm{IF}_{\mathrm{t}}^{\mathrm{med}} \subseteq \mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}} \subseteq \mathrm{TotP}$*. The first inclusion is proper unless* $\#\mathrm{P} = \mathrm{FP}$ *and the second inclusion is proper unless* $\mathrm{P} = \mathrm{NP}$*.*

Proof. The detailed proof is left for the full version. We only sketch some key ideas. For showing that $\mathrm{FP} = \mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$ implies $\#\mathrm{P} = \mathrm{FP}$, we consider any function $f \in \#\mathrm{P}$ and derive a function $g(x) = f(x) + 2^{p(|x|)}$, where p is a polynomial bounding the computation length of the NPTM that corresponds to f. We next show that $g \in \mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$; it then suffices to notice that if $g \in \mathrm{FP}$, then so does f. For the proof of the assumptions under wich the second inclusion is proper, we introduce the exponential gap operator, $\mathcal{C}_{\mathrm{eg}}\cdot$, defined as follows: if $\mathcal{F}$ is a function class, then $\mathcal{C}_{\mathrm{eg}} \cdot \mathcal{F}$ contains exactly the languages L for which there exist some $f \in \mathcal{F}$, $q \in \mathrm{poly}$, and $q' \in \omega(1)$ such that for all x: if $x \notin L$ then $f(x) \leq 2^{q(|x|)}$, while if $x \in L$ then $f(x) \geq 2^{q(|x|)\cdot q'(|x|)}$. We then prove that $\mathrm{NP} \subseteq \mathcal{C}_{\mathrm{eg}} \cdot \mathrm{TotP}$ and $\mathcal{C}_{\mathrm{eg}} \cdot \mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}} \subseteq \mathrm{P}$. □

Let us now define a problem that lies in $\mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$:

$\#\mathrm{SAT}_{+2^n}$: given a Boolean formula φ with n variables, count the number of satisfying assignments of the formula $\varphi \vee x_{n+1}$, where x_{n+1} is a fresh variable not appearing in φ.

Proposition 2. $\#\mathrm{SAT}_{+2^n}$ *is* $\mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}$*-complete under Cook-1 reductions.*

Proof. Membership can be shown by similar techniques to those used for proving the first part of Theorem 4 (omitted due to lack of space). For completeness it suffices to observe that #SAT can be immediately reduced to $\#\mathrm{SAT}_{+2^n}$ by subtracting 2^n. □

From the argument used in the above proof, the following is immediate (for function classes $\mathcal{F}$, $\mathcal{G}$, let $\mathcal{F}\text{-}\mathcal{G} = \{f - g \mid f \in \mathcal{F}, g \in \mathcal{G}\}$):

Corollary 2. $\#\mathrm{P} \subseteq \mathrm{IF}_{\mathrm{t}}^{\mathrm{rmed}}\text{-}\mathrm{FP}$.

Acknowledgements. We would like to thank Taso Viglas and Stathis Zachos for stimulating discussions, and the anonymous referees for their useful comments and suggestions.

References

1. Hemaspaandra, L.A., Homan, C.M., Kosub, S., Wagner, K.W.: The complexity of computing the size of an interval. SIAM J. Comput. 36(5), 1264–1300 (2007)
2. Kiayias, A., Pagourtzis, A., Sharma, K., Zachos, S.: The complexity of determining the order of solutions. In: Proceedings of the First Southern Symposium on Computing, Hattiesburg, Mississippi, December 4-5 (1998); Extended and revised version: Acceptor-definable complexity classes. LNCS 2563, pp. 453–463. Springer, Heidelberg (2003)
3. Valiant, L.G.: The complexity of computing the permanent. Theor. Comput. Sci. 8, 189–201 (1979)
4. Valiant, L.G.: The complexity of enumeration and reliability problems. SIAM J. Comput. 8(3), 410–421 (1979)
5. Toda, S.: PP is as hard as the polynomial-time hierarchy. SIAM J. Comput. 20(5), 865–877 (1991)
6. Toda, S., Watanabe, O.: Polynomial time 1-Turing reductions from #PH to #P. Theor. Comput. Sci. 100(1), 205–221 (1992)
7. Kiayias, A., Pagourtzis, A., Zachos, S.: Cook reductions blur structural differences between functional complexity classes. In: Panhellenic Logic Symposium, pp. 132–137 (1999)
8. Dyer, M.E., Goldberg, L.A., Greenhill, C.S., Jerrum, M.: The relative complexity of approximate counting problems. Algorithmica 38(3), 471–500 (2003)
9. Àlvarez, C., Jenner, B.: A very hard log space counting class. In: Structure in Complexity Theory Conference, pp. 154–168 (1990)
10. Saluja, S., Subrahmanyam, K.V., Thakur, M.N.: Descriptive complexity of #P functions. J. Comput. Syst. Sci. 50(3), 493–505 (1995)
11. Pagourtzis, A.: On the complexity of hard counting problems with easy decision version. In: Proceedings of 3rd Panhellenic Logic Symposium, Anogia, Crete, July 17-21 (2001)
12. Pagourtzis, A., Zachos, S.: The complexity of counting functions with easy decision version. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 741–752. Springer, Heidelberg (2006)
13. Hempel, H., Wechsung, G.: The operators min and max on the polynomial hierarchy. Int. J. Found. Comput. Sci. 11(2), 315–342 (2000)