Cook Reductions Blur Structural Differences Between Functional Complexity Classes

Aggelos Kiayias* Aris Pagourtzis† Stathis Zachos*†

Abstract

We investigate various notions of reducibilities among polynomial-time complexity classes of functions. Our basic claim is that Cook-like reductions for functions obscure significant differences of complexity classes while Karp-like functional reductions do not. To demonstrate this dissimilarity we compare $\sharp\mathsf{P}$ (counting accepting computational paths) and TotP (counting all computational paths). We consider different modes of computation with respect to the "shape" of the non-deterministic computation tree, and investigate these classes in this context. We see that TotP possesses a remarkably distinct behavior compared to traditional complexity classes. We prove that TotP is as powerful as $\sharp\mathsf{P}$ when used as an oracle by a polynomial-time computation — even if the oracle is used once. In other words, $\sharp\mathsf{P}$ and TotP are indistinguishable under Cook reductions although actually very different classes (unless of course $\mathsf{P} = \mathsf{NP}$).

1 Introduction

One of the most important topic of classical computational complexity theory deals with language classes and Karp or Cook reductions among languages. In this paper we define and compare some functional complexity classes by using computational paths of non-deterministic Turing machines and investigate some reductions that are defined in a similar manner to the Karp and Cook reductions for language classes.

We note here that several researchers [Kre88, KST89] have investigated such issues considering functions computed by non-deterministic transducers, i.e. non-deterministic machines which output some string on each possible computation. We, on the other hand, restrict our attention to non-deterministic acceptors (or deciders) and our functions are defined by counting all computational paths, or only accepting computational paths, or etc..

More specifically, Valiant [Val79] has introduced the functional class $\sharp\mathsf{P}$ (counting only accepting paths) and we [KPSZ98] have introduced TotP (counting all computational paths). By using Karp-like and Cook-like reductions we compare the power of these functional complexity classes.

Our new class is Cook-interreducible with $\sharp\mathsf{P}$: $\mathsf{P}^{\mathsf{TotP}} = \mathsf{P}^{\sharp\mathsf{P}}$, even $\mathsf{P}^{\mathsf{TotP}[1]} = \mathsf{P}^{\sharp\mathsf{P}[1]}$. On the other hand, $\sharp\mathsf{P}$ cannot be Karp-reduced to TotP.

*CIS Department, Brooklyn College, New York, USA, akiayias@sci.brooklyn.cuny.edu

†Computer Science Division, Department of ECE, National Technical University of Athens, Greece, pagurzis@cs.ntua.gr

Computations of a non-deterministic Turing machine (NTM) can be represented by trees. Early on it was noted that in most cases we can restrict computation trees to binary trees without loss of generality w.r.t. the computational power. Recently the trend has been to study the influence of even more detailed structural properties of (binary) computation trees (e.g. whether complete, balanced, etc.). An important question in this context is whether the computational power of a complexity class decreases if only computation trees belonging to a special family are allowed.

In [HVW96] Hertrampf, Vollmer and Wagner defined the class Path which is based on the value of the i-th path of a NTM. Path is supposedly a class much higher than P. In that paper it is stated that if the definition of the class Path was restricted only to balanced computation trees the corresponding complexity class would be lower than P. Balanced trees were discussed in [JMT94]. We may say that Path has an "irregular" behavior, since most known complexity classes remain invariant under similar "reasonable" definitional restrictions. For example NP, $\sharp\mathsf{P}$, PP possess the same computational power even if we restrict shapes of computation trees to complete binary trees.

In this paper we note that such "irregular" behavior may be related to the fact that both accepting and rejecting paths are important for determining the outcome of the computation. TotP is a natural class with such behavior.

We also introduce the concept of class-shape, in order to provide a formal framework for investigating complexity classes whose possible computation trees are restricted. If a family of trees S is sufficient to describe all computations of all members of a class $\mathcal{C}$, then we say that S is a *sufficient shape* for $\mathcal{C}$. We focus on the family of complete binary trees ($\mathtt{CBT}$) that turns out to be a sufficient shape for many known complexity classes but not for TotP (unless $\mathsf{P} = \mathsf{NP}$). The $\mathtt{CBT}$ version of our TotP class is trivial and thus certainly computable in polynomial time.

2 Karp and Cook Reductions for Function Classes

The model of computation we use is the Non-Deterministic Turing Machine (NTM) with the following normalizations: on any input x, the computation tree of a NTM is binary and all paths of the tree have the same length.

A NTM running in polynomial time will be called poly-NTM. For any poly-NTM M, on input x, all computation paths have the same length $p_M(|x|)$ for some polynomial p_M. We can assign to each computation path a string from $\{0,1\}^{p_M(|x|)}$ that describes the non-deterministic choices at each step. Nodes with a single next configuration (i.e. actually deterministic steps) will by convention be considered as 0-choices.

The set of all choicestrings (or paths) of a NTM M on input x will be denoted by $Paths_M(x)$. For a comprehensive introduction to NTMs and other notions discussed in this paper the reader is referred to [Pap94] or [BDG88]. All functions that are considered throughout the paper, unless otherwise stated, have Σ^* as their domain and $\mathbb{N}$ (nonnegative integers) as their range: $(\Sigma^* \rightarrow \mathbb{N})$.

Definition 2.1 *The following functions are defined via a NTM M:*

1. $\mathrm{acc}_M(x)$ *is the number of accepting paths of M on input x.*
2. $\mathrm{tot}_M(x)$ *is the total number of paths of M on input x minus one*[1]*.*

[1]The "minus one" is introduced so that 0 is a possible function value.

We consider the following functional complexity classes:

$$\sharp\mathsf{P} = \{f \mid \exists \text{ poly-NTM } M : f = \mathrm{acc}_M\} \text{ defined in [Val79]}$$
$$\mathsf{TotP} = \{f \mid \exists \text{ poly-NTM } M : f = \mathrm{tot}_M\} \text{ defined in [KPSZ98]}$$

We will also make use of the class FP that contains functions of $(\Sigma^* \to \Sigma^*)$ computable in polynomial time by deterministic Turing machines with output tape (*transducers*).

An *oracle* NTM M^g is a NTM equipped with an extra semi-infinite tape; at any time, M can write a string y on this tape (called oracle or query tape) and enter a special query state. The string in the oracle tape is then replaced by the representation of $g(y)$, g is called the oracle function. Then M can read $g(y)$ from the tape and continue computing, possibly posing several queries to the oracle function. We use P^g to denote the class of languages that can be decided by a poly-NTM with the help of the oracle function g. For a function class $\mathcal{G}$, $\mathsf{P}^{\mathcal{G}} = \bigcup_{g\in\mathcal{G}} \mathsf{P}^g$. The function classes FP^g and $\mathsf{FP}^{\mathcal{G}}$ are defined analogously, considering transducers equipped with a functional oracle.

Reductions for function classes are defined in a similar manner to the Cook and Karp reductions for language classes:

Definition 2.2

(i) *Karp reduction:* $f \leq^p_{\mathrm{m}} g$ *iff* $\exists\, h \in \mathsf{FP}$ *such that* $\forall x\; f(x) = g(h(x))$.

(ii) *Cook[1] reduction:* $f \leq^p_{1-\mathrm{T}} g$ *iff* $\exists\, h_1, h_2 \in \mathsf{FP}$ *such that* $\forall x\; f(x) = h_1(x, g(h_2(x)))$.

(iii) $\mathcal{F}$ *Karp-(Cook[1]-) reduces to* $\mathcal{G}$*:* $\forall f \in \mathcal{F}, \exists\, g \in \mathcal{G}$ *such that* $f \leq^p_{\mathrm{m}} (\leq^p_{1-\mathrm{T}}) g$.

Remark 2.3 Many people have used the notion of Karp reduction between functions: e.g. [Wag86, Tor88, Vol94]. Some researchers (e.g. [TW92]) call it *parsimonious*, a name that Simon [Sim75] first used for Karp reductions between counting functions. The notion of Cook reduction between functions has been widely used (e.g. in [Val79, Kre88, KST89, TW92]). Krentel in [Kre88] used Cook[1] reduction for functions and introduced a new name for it: *metric*. This latter reduction is called "polynomial-time 1-Turing reduction" in [TW92]; we prefer "Cook[1]" which is a shorter way to say both "polynomial-time" and "1-Turing".

The following proposition can be easily shown:

Proposition 2.4

(a) $f \leq^p_{1-\mathrm{T}} g$ *iff* $f \in \mathsf{FP}^{g[1]}$ *where* $[1]$ *stands for at most one question to* g.

(b) $\leq^p_{\mathrm{m}}$ *and* $\leq^p_{1-\mathrm{T}}$ *are reflexive and transitive.*

(c) $f \leq^p_{\mathrm{m}} g \Rightarrow f \leq^p_{1-\mathrm{T}} g$.

(d) *if* g *is a* $\leq^p_{\mathrm{m}}$*-complete (or* $\leq^p_{1-\mathrm{T}}$*-complete) function for the class* $\mathcal{G}$ *then* $\mathsf{P}^{g[1]} = \mathsf{P}^{\mathcal{G}[1]}$ *and* $\mathsf{FP}^{g[1]} = \mathsf{FP}^{\mathcal{G}[1]}$.

(e) *if* f *is* $\leq^p_{\mathrm{m}}$*-complete (or* $\leq^p_{1-\mathrm{T}}$*-complete) function for the class* $\mathcal{F}$ *and* $f \leq^p_{1-\mathrm{T}} g$ *then* $\mathcal{F} \subseteq \mathsf{FP}^{g[1]}$.

(f) *the function classes* $\mathcal{F}$ *and* $\mathcal{G}$ *are Cook[1]-interreducible (or indistinguishable under Cook[1] reductions)*
iff $\mathcal{F} \subseteq \mathsf{FP}^{\mathcal{G}[1]}$ *and* $\mathcal{G} \subseteq \mathsf{FP}^{\mathcal{F}[1]}$
iff $\mathsf{FP}^{\mathcal{F}[1]} = \mathsf{FP}^{\mathcal{G}[1]}$
iff $\mathsf{P}^{\mathcal{F}[1]} = \mathsf{P}^{\mathcal{G}[1]}$.

3 Computation Trees and Shapes

For a NTM M let $R_M = \{\langle x, y\rangle \mid y \in Paths_M(x)\}$.

Definition 3.1 *A binary relation R is called* (polynomial time) tree-constructible *if there exists a poly-NTM M such that $R = R_M$.*

Notice that a tree-constructible relation can be decided in polynomial time because one can deterministically check whether a given computational path represents a legal computation for a given input x. If $\mathtt{S}$ is a family of tree-constructible relations, and $\mathcal{F}$ is a class of functions defined via NTMs then the $\mathtt{S}$-restriction of $\mathcal{F}$ is defined as follows:

$\mathtt{S}\diamond\mathcal{F} = \{f \mid f \in \mathcal{F}$, and there exists M s.t. f is defined via M and $R_M \in \mathtt{S}\}$.

Definition 3.2 *A family of tree-constructible relations $\mathtt{S}$ is a* sufficient shape *for a class of functions $\mathcal{F}$ if and only if $\mathtt{S}\diamond\mathcal{F} = \mathcal{F}$.*

Some interesting families of tree-constructible relations:

- $\mathtt{Bin}$, the family corresponding to binary trees.
- $\mathtt{CBT}$, the family corresponding to complete binary trees, i.e. $R \in \mathtt{CBT}$ iff there exists a polynomial p such that $R(x, y) \iff |y| = p(|x|)$.
- $\mathtt{Bal}$, defined in [JMT94], an extension of $\mathtt{CBT}$ where $R \in \mathtt{Bal}$ iff there exist a polynomial p such that for all inputs x, $\exists z \in \Sigma^*, |z| = p(|x|) : \forall y(R(x, y) \iff |y| = p(|x|) \wedge y \leq z)$.

Notice that $\mathtt{CBT}\diamond\mathcal{F} \subseteq \mathtt{Bal}\diamond\mathcal{F} \subseteq \mathtt{Bin}\diamond\mathcal{F} \subseteq \mathcal{F}$ where $\mathcal{F}$ is a functional complexity class. It should be obvious that for all traditional complexity classes, $\mathtt{Bin}$ is a sufficient shape, i.e. we can restrict our model of computation to NTMs whose computational trees are binary. Moreover for the vast majority of classes (including $\sharp\mathsf{P}$), $\mathtt{Bal}$ and even the more restricted $\mathtt{CBT}$ family can be proven to be sufficient shapes. In other words, we can restrict our computational model to NTMs with complete binary computation trees. However, for probabilistic complexity classes like BPP, RP, ZPP it seems that unrestricted-tree versions properly contain $\mathtt{CBT}$ versions. We will see in the following section that the class TotP is also an exception.

4 Similarities and Differences Between $\sharp\mathsf{P}$ and TotP

If we can count accepting paths then we can surely count all paths (minus one): just transform every (but the first) path to an accepting path and count. Thus:

Proposition 4.1 $\mathsf{TotP} \subseteq \sharp\mathsf{P}$.

Furthermore, the classes TotP and $\sharp\mathsf{P}$, despite their significant structural differences (see below), are Cook[1]-interreducible.

Proposition 4.2 $\mathsf{P}^{\sharp\mathsf{P}[1]} = \mathsf{P}^{\mathsf{TotP}[1]}$.

Proof. The "$\supseteq$"-direction follows immediately from 4.1.

The "$\subseteq$"-direction: Assume $L \in \mathsf{P}^{\sharp\mathsf{P}[1]}$; then there is a $f \in \sharp\mathsf{P}$ s.t. L is decided by a DTM D with a single query to f. Since $\mathtt{CBT}$ is a sufficient shape for $\sharp\mathsf{P}$, we can assume w.l.o.g. that f is defined via a CBT-NTM M. The query of D asking for the number of accepting paths of M can be replaced by a query asking for the total number of paths of an appropriate NTM N. N operates exactly as M on input x, but when an accepting configuration is reached, N forks once duplicating the accepting configuration. Therefore we have that $\mathrm{acc}_M(x) = \mathrm{tot}_N(x) - \mathrm{tot}_M(x) = \mathrm{tot}_N(x) - 2^{p_M(|x|)} + 1$.

So the $\sharp\mathsf{P}$-query can be simulated by one TotP-query (and a polynomial computation). Consequently $L \in \mathsf{P}^{\mathsf{TotP}[1]}$. $\square$

In other words TotP and $\sharp\mathsf{P}$ are indistinguishable under Cook[1] reductions. On the other hand, TotP has a different behavior than $\sharp\mathsf{P}$ when it comes to shapes of computation trees.

Proposition 4.3 $\mathtt{Bal}\diamond\mathsf{TotP} \neq \mathsf{TotP}$ *unless* $\mathsf{P} = \mathsf{NP}$.

Proof. We first show that $\mathtt{Bal}\diamond\mathsf{TotP} \subseteq \mathsf{FP}$. Let $f \in \mathtt{Bal}\diamond\mathsf{TotP}$ i.e. there exists a balanced-tree NTM M s.t. $f(x) = \mathrm{tot}_M(x)$. A deterministic Turing machine can follow the (lexicographically) greatest path of M and output the corresponding binary number, which is exactly the number of paths of the computational tree minus one. Thus $f \in \mathsf{FP}$.

Now, suppose that $\mathsf{TotP} = \mathtt{Bal}\diamond\mathsf{TotP}$. Then (using 4.2) $\mathsf{NP} \subseteq \mathsf{P}^{\sharp\mathsf{P}} = \mathsf{P}^{\mathsf{TotP}} = \mathsf{P}^{\mathtt{Bal}\diamond\mathsf{TotP}} \subseteq \mathsf{P}^{\mathsf{FP}} = \mathsf{P}$ $\square$

Thus, TotP is also different than $\mathtt{CBT}\diamond\mathsf{TotP}$ which is a completely trivial class (the number of paths of a $\mathtt{CBT}$ is $2^{p(|x|)}$), subsumed in FP. Note also that $\mathtt{Bal}\diamond\mathsf{TotP} = \mathsf{FP}$. Consequently, $\mathtt{CBT}\diamond\mathsf{TotP} \subset \mathtt{Bal}\diamond\mathsf{TotP} \subseteq \mathsf{TotP}$ (the second inclusion being proper under the assumption $\mathsf{P} \neq \mathsf{NP}$). In contrast, $\mathtt{CBT}\diamond\sharp\mathsf{P} = \mathtt{Bal}\diamond\sharp\mathsf{P} = \sharp\mathsf{P}$.

We have seen that $\sharp\mathsf{P}$ can be Cook[1]-reduced to TotP, although these two classes have significant structural differences, namely with respect to what shapes are sufficient for them. In other words Cook[1] reductions blur important differences. On the other hand Karp reductions do not. Specifically, it does not seem possible to Karp-reduce $\sharp\mathsf{P}$ to TotP (i.e. there exist functions in $\sharp\mathsf{P}$ that cannot be $\leq_{\mathrm{m}}^{p}$-reduced to any function in TotP).

Proposition 4.4 $\sharp\mathsf{P}$ *does not Karp-reduce to* TotP *unless* $\mathsf{P} = \mathsf{NP}$.

Proof. Suppose that $\sharp\mathsf{P}$ Karp-reduces to TotP. This means that for every poly-NTM M we can find another one N so that: $\mathrm{acc}_M(x) = \mathrm{tot}_N(h(x))$, for some $h \in \mathsf{FP}$. Recall that $\mathrm{tot}_N(y) = |Paths_N(y)| - 1$. But then we would have $\mathsf{NP} \subseteq \mathsf{P}$, because given any poly-NTM M, we could check deterministically in polynomial time whether the corresponding poly-NTM N has more than one paths (i.e. $\mathrm{acc}_M(x) > 0$), in the following manner:

input x;
compute $h(x)$ (in polynomial time); simulate N on $h(x)$;
if $N(h(x))$ branches **then** accept
else (i.e. if N terminates without branching) reject. $\square$

5 Summary and Conclusion

We discussed functional complexity classes, i.e. FP, TotP and $\sharp\mathsf{P}$, and sufficient shapes for computation trees, i.e. binary ($\mathtt{Bin}$), complete binary ($\mathtt{CBT}$) and balanced ($\mathtt{Bal}$); furthermore, we compared classes using Karp — and/or Cook[1] — reductions. The main results are:

(a) $\mathsf{P}^{\mathsf{TotP}[1]} = \mathsf{P}^{\sharp\mathsf{P}[1]}$, i.e. TotP and $\sharp\mathsf{P}$ are Cook[1]-equivalent, and
(b) $\mathtt{CBT}\diamond\mathsf{TotP} \subset \mathsf{FP} = \mathtt{Bal}\diamond\mathsf{TotP} \subseteq \mathsf{TotP} \subseteq \sharp\mathsf{P}$, where the last two inclusions are proper unless $\mathsf{P} = \mathsf{NP}$.

We conclude that Karp reductions are much more suitable than Cook reductions, for classification of functions and investigation of functional complexity classes.

References

[BDG88] J. L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity I*. Springer-Verlag, 1988.

[HVW96] U. Hertrampf, H. Vollmer, and K. W. Wagner. On balanced versus unbalanced computation trees. *Mathematical Systems Theory*, 29(4):411–421, July/August 1996.

[JMT94] Birgit Jenner, Pierre McKenzie, and Denis Thérien. Logspace and logtime leaf languages. In *Proceedings of the Ninth Annual Structure in Complexity Theory Conference*, pages 242–254, Amsterdam, The Netherlands, 28 June–1 July 1994. IEEE Computer Society Press.

[KPSZ98] Aggelos Kiayias, Aris Pagourtzis, Kiron Sharma, and Stathis Zachos. The complexity of determining the order of solutions. In *Proceedings of the First Southern Symposium on Computing*, Hattiesburg, Mississippi, 4–5 December 1998.

[Kre88] Mark W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, June 1988.

[KST89] Johannes Köbler, Uwe Schöning, and Jacobo Torán. On counting and approximation. *Acta Informatica*, 26(4):363–379, 1989.

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

[Sim75] J. Simon. *On Some Central Problems of Computational Complexity*. PhD thesis, Cornell University, Ithaca, NY, 1975.

[Tor88] J. Toran. *Structural Properties of the Counting Hierarchies*. PhD thesis, Facultat d'Informatica de Barcelona, 1988.

[TW92] Seinosuke Toda and Osamu Watanabe. Polynomial-time 1-turing reductions from $\sharp$PH to $\sharp$P. *Theoretical Computer Science*, 100(1):205–221, 1992.

[Val79] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, April 1979.

[Vol94] Heribert Vollmer. On different reducibility notions for function classes. In *11th Annual Symposium on Theoretical Aspects of Computer Science*, volume 775 of *Lecture Notes in Computer Science*, pages 449–460, Caen, France, 24–26 February 1994. Springer.

[Wag86] Klaus W. Wagner. Some observations on the connection between counting and recursion. *Theoretical Computer Science*, 47(2):131–147, 1986.