

# Consistency and Optimality

Yijia Chen<sup>1</sup>, Jörg Flum<sup>2</sup>, and Moritz Müller<sup>3</sup>

<sup>1</sup> Department of Computer Science and Engineering, Shanghai Jiao Tong University,  
Dongchuan Road, No. 800, 200240 Shanghai, China

`yijia.chen@cs.sjtu.edu.cn`

<sup>2</sup> Abteilung für mathematische Logik, Albert-Ludwigs-Universität Freiburg,  
Eckerstraße 1, 79104 Freiburg, Germany

`joerg.flum@math.uni-freiburg.de`

<sup>3</sup> Centre de Recerca Matemàtica, Campus de Bellaterra, Edifici C 08193, Bellaterra,  
Spain

`mmueller@crm.cat`

**Abstract.** Assume that the problem  $Q_0$  is not solvable in polynomial time. For theories  $T$  containing a sufficiently rich part of true arithmetic we characterize  $T \cup \{\text{Con}_T\}$  as the minimal extension of  $T$  proving for some algorithm that it decides  $Q_0$  as fast as any algorithm  $\mathbb{B}$  with the property that  $T$  proves that  $\mathbb{B}$  decides  $Q_0$ . Here,  $\text{Con}_T$  claims the consistency of  $T$ . Moreover, we characterize problems with an optimal algorithm in terms of arithmetical theories.

## 1 Introduction

By Gödel's Second Incompleteness Theorem a consistent, computably enumerable and sufficiently strong theory  $T$  cannot prove its own consistency  $\text{Con}_T$ . In other words,  $T \cup \{\text{Con}_T\}$  is a proper extension of  $T$ .

In Bounded Arithmetic one studies the complexity of proofs in terms of the computational complexity of the concepts involved in the proofs (see e.g. [1, Introduction]). Stronger theories allow reasoning with more complicated concepts. For example, a computational problem may be solvable by an algorithm whose proof of correctness needs tools not available in the given theory; moreover, stronger theories may know of faster algorithms solving the problem. When discussing these issues with the authors, Sy-David Friedman asked whether  $T \cup \{\text{Con}_T\}$  can be characterized in this context as a minimal extension of  $T$ . We could prove the following result (all terms will be defined in the paper).

**Theorem 1.** *Let  $Q_0$  be a decidable problem not in PTIME. Then there is a finite true arithmetical theory  $T_0$  and a computable function  $F$  assigning to every computably enumerable theory  $T$  with  $T \supseteq T_0$  an algorithm  $F(T)$  such that (a) and (b) hold:*

(a)  $T_0$  proves that  $F(T)$  is as fast as any algorithm  $T$ -provably deciding  $Q_0$ .

(b) For every theory  $T^*$  with  $T^* \supseteq T$  the following are equivalent:

(i)  $T^*$  proves  $\text{Con}_T$ .

- (ii) *The algorithm  $F(T)$   $T^*$ -provably decides  $Q_0$ .*
- (iii) *There is an algorithm such that  $T^*$  proves that it decides  $Q_0$  and that it is as fast as any algorithm  $T$ -provably deciding  $Q_0$ .*

Hence, by merely knowing the extension  $T$  of  $T_0$  we are able to compute the algorithm  $F(T)$ , which is, provably in  $T_0$ , as fast as any algorithm  $T$ -provably deciding  $Q_0$ ; however, in order to prove that  $F(T)$  decides  $Q_0$  we need the full strength of  $T \cup \{\text{Con}_T\}$ . In this sense,  $T \cup \{\text{Con}_T\}$  is a minimal extension of  $T$ .

The content of the different sections is the following. In Section 3, by a standard diagonalization technique we derive a result showing for every computably enumerable set  $D$  of algorithms the existence of an algorithm that on every input behaves as some algorithm in  $D$  and that is as fast as every algorithm in  $D$  (see Lemma 2). In Theorem 7 of Section 4 we characterize problems with an optimal algorithm in terms of arithmetical theories. Finally Section 5 contains a proof of Theorem 1.

Many papers in computational complexity, older and recent ones, address the question whether hard problems have *optimal* or *almost optimal* algorithms. Although Levin [5] observed that there exists an optimal algorithm that finds a satisfying assignment for every satisfiable propositional formula, it is not known whether the class of satisfiable propositional formulas or the class of tautologies have an almost optimal algorithm.

Krajíček and Pudlák [4] showed for the latter class that an almost optimal algorithm exists if and only if “there exists a finitely axiomatized fragment  $T$  of the true arithmetic such that, for every finitely axiomatized consistent theory  $S$ , there exists a deterministic Turing machine  $\mathbb{M}$  and a polynomial  $p$  such that for any given  $n$ , in time  $\leq p(n)$  the machine  $\mathbb{M}$  constructs a proof in  $T$  of  $\text{Con}_S(\underline{n})$ .” Here  $\text{Con}_S(\underline{n})$  claims that no contradiction can be derived from  $S$  by proofs of lengths at most  $n$ .

Hartmanis [2] and Hutter [3] considered ‘provable’ algorithms, where ‘provable’ refers to a computably enumerable, more or less specified true theory  $T$ . Hartmanis compares the class of problems decidable within a given time bound with the class of problems  $T$ -provably decidable within this time bound and he studies time hierarchy theorems in this context. Hutter constructs an algorithm “which is the fastest and the shortest” deciding a given problem. As Hutter says, Peter van Emde Boas pointed out to him that it is not provable that his algorithm decides the given problem and that his proof is a “meta-proof which cannot be formalized within the considered proof system” and he adds that “a formal proof of its correctness would prove the consistency of the proof system, which is impossible by Gödel’s Second Incompleteness Theorem.”

## 2 Some preliminaries

First we fix some notations and introduce some basic concepts. We consider problems as subsets of  $\Sigma^*$ , the set of strings over the alphabet  $\Sigma = \{0, 1\}$ . For

an algorithm  $\mathbb{A}$  and a string  $x \in \Sigma^*$  we let  $t_{\mathbb{A}}(x)$  denote the running time of  $\mathbb{A}$  on  $x$ . In case  $\mathbb{A}$  does not halt on  $x$ , we set  $t_{\mathbb{A}}(x) := \infty$ . If  $t_{\mathbb{A}}(x)$  is finite, we denote by  $\mathbb{A}(x)$  the output of  $\mathbb{A}$  on  $x$ .

If  $\mathbb{A}$  and  $\mathbb{B}$  are algorithms, then  $\mathbb{A}$  is as fast as  $\mathbb{B}$  if there is a polynomial  $p$  such that for every  $x \in \Sigma^*$

$$t_{\mathbb{A}}(x) \leq p(t_{\mathbb{B}}(x) + |x|). \quad (1)$$

Note that here we do not require that  $\mathbb{A}$  and  $\mathbb{B}$  decide the same  $Q \subseteq \Sigma^*$ .

An algorithm deciding  $Q$  is *optimal* if it is as fast as every other algorithm deciding  $Q$ , that is, if it has no superpolynomial speedup infinitely often. An algorithm  $\mathbb{A}$  deciding  $Q$  is *almost optimal* if (1) holds for every other algorithm deciding  $Q$  and every  $x \in Q$  (hence nothing is required of the relationship between  $t_{\mathbb{A}}(x)$  and  $t_{\mathbb{B}}(x)$  for  $x \notin Q$ ).

We do not distinguish algorithms from their codes by strings and we do not distinguish strings from their codes by natural numbers. However, we do not fix a computation model (Turing machines, random access machines,...) for algorithms. We state the results in such a way that they hold for every standard computation model.

### 3 Diagonalizing over algorithms

In computability theory diagonalization techniques are used in various contexts. We will make use of the following result.

**Lemma 2 (Diagonalization Lemma).** *Let  $D$  be a computably enumerable and nonempty set of algorithms. Then there is an algorithm  $\mathbb{A}$  such that (a) and (b) hold.*

(a) *The algorithm  $\mathbb{A}$  halts precisely on those inputs on which at least one algorithm in  $D$  halts, and in that case it outputs the same as some algorithm in  $D$ ; more formally, for all  $x \in \Sigma^*$*

$$- t_{\mathbb{A}}(x) < \infty \iff t_{\mathbb{D}}(x) < \infty \text{ for some } \mathbb{D} \in D;$$

$$- \text{if } t_{\mathbb{A}}(x) < \infty, \text{ then there is } \mathbb{D} \in D \text{ with } \mathbb{A}(x) = \mathbb{D}(x).$$

(b) *There is a  $d \in \mathbb{N}$ <sup>4</sup> such that for all  $\mathbb{D} \in D$  there is a  $c_{\mathbb{D}}$  such that for all  $x \in \Sigma^*$*

$$t_{\mathbb{A}}(x) \leq c_{\mathbb{D}} \cdot (t_{\mathbb{D}}(x) + |x|)^d.$$

*Moreover, there is a computable function that maps any algorithm  $\mathbb{E}$  enumerating the set  $D$  of algorithms to an algorithm  $\mathbb{A}$  satisfying (a) and (b).*

*In particular, if all algorithms in  $D$  decide  $Q \subseteq \Sigma^*$ , then  $\mathbb{A}$  is an algorithm deciding  $Q$  as fast as every  $\mathbb{D} \in D$ .*

---

<sup>4</sup> As the proof shows the constant  $d \in \mathbb{N}$  does not even depend on  $D$  but it depends on the concrete machine model one uses.

*Proof.* Let the algorithm  $\mathbb{E}$  enumerate the set  $D$  of algorithms, that is,  $\mathbb{E}$ , once having been started, eventually prints out exactly the algorithms in  $D$ . For each  $i \in \mathbb{N}$  we denote by  $\mathbb{E}_i$  the last algorithm printed out by  $\mathbb{E}$  in  $i$  steps; in particular,  $\mathbb{E}_i$  is undefined if  $\mathbb{E}$  hasn't printed any algorithm in  $i$  steps.

Algorithm  $\mathbb{A}$  is defined as follows.

```

 $\mathbb{A}(x)$  //  $x \in \Sigma^*$ 

1.  $\ell \leftarrow 0$ 
2. for  $i = 0$  to  $\ell$ 
3.     if  $\mathbb{E}_i$  is defined then simulate the  $(\ell - i)$ th step
4.     of  $\mathbb{E}_i$  on  $x$ 
5.     if the simulation halts then halt and output
6.     accordingly
7.  $\ell \leftarrow \ell + 1$ 
8. goto 2.

```

Of course (the code of)  $\mathbb{A}$  can be computed from (the code of)  $\mathbb{E}$ . It is easy to see that  $\mathbb{A}$  satisfies (a). Furthermore, there are constants  $c_0, d_0 \in \mathbb{N}$  such that for all  $x \in \Sigma^*$  and every  $\ell \in \mathbb{N}$ , lines 2–6 take time at most

$$c_0 \cdot (\ell + |x|)^{d_0}. \quad (2)$$

To verify (b), let  $\mathbb{D} \in D$  and  $i_{\mathbb{D}}$  be the minimum  $i \in \mathbb{N}$  with  $\mathbb{E}_i = \mathbb{D}$ . Fix an input  $x \in \Sigma^*$ . For

$$\ell = i_{\mathbb{D}} + t_{\mathbb{E}_{i_{\mathbb{D}}}}(x) \text{ and } i = i_{\mathbb{D}}$$

the simulation in line 3 halts if it didn't halt before. Therefore

$$\begin{aligned} t_{\mathbb{A}}(x) &\leq O\left(\sum_{\ell=0}^{i_{\mathbb{D}}+t_{\mathbb{D}}(x)} (\ell + |x|)^{d_0}\right) \quad (\text{by (2)}) \\ &\leq O((i_{\mathbb{D}} + t_{\mathbb{D}}(x) + |x|)^{d_0+1}) \leq c_{\mathbb{D}} \cdot (t_{\mathbb{D}}(x) + |x|)^{d_0+1} \end{aligned}$$

for an appropriate constant  $c_{\mathbb{D}} \in \mathbb{N}$  only depending on  $\mathbb{D}$ . □

The preceding proof uses the idea underlying standard proofs of a result due to Levin [5]. Even more, Levin's result is also a consequence of Lemma 2:

**Example 3 (Levin [5]).** Let  $F : \Sigma^* \rightarrow \Sigma^*$  be computable. An *inverter* of  $F$  is an algorithm  $\mathbb{I}$  that given  $y$  in the image of  $F$  halts with some output  $\mathbb{I}(y)$  such that  $F(\mathbb{I}(y)) = y$ . On inputs not in the image of  $F$ , the algorithm  $\mathbb{I}$  may do whatever it wants.

Let  $\mathbb{F}$  be an algorithm computing  $F$ . For an arbitrary algorithm  $\mathbb{B}$  define  $\mathbb{B}^*$  as follows. On input  $y$  the algorithm  $\mathbb{B}^*$  simulates  $\mathbb{B}$  on  $y$ ; if the simulation halts, then by simulating  $\mathbb{F}$  it computes  $F(\mathbb{B}(y))$ ; if  $F(\mathbb{B}(y)) = y$ , then it outputs

$\mathbb{B}(y)$ , otherwise it does not stop. Thus if  $\mathbb{B}^*$  halts on  $y \in \Sigma^*$ , then it outputs a preimage of  $y$  and

$$t_{\mathbb{B}^*}(y) \leq O(t_{\mathbb{B}}(y) + t_{\mathbb{F}}(\mathbb{B}(y)) + |y|). \quad (3)$$

Furthermore, if  $\mathbb{B}$  is an inverter of  $F$ , then so is  $\mathbb{B}^*$ .

Let  $D := \{\mathbb{B}^* \mid \mathbb{B} \text{ is an algorithm}\}$ . Denote by  $\mathbb{I}_{\text{opt}}$  an algorithm having for this  $D$  the properties of the algorithm  $\mathbb{A}$  in Lemma 2. By the previous remarks it is easy to see that  $\mathbb{I}_{\text{opt}}$  is an inverter of  $F$ . Moreover, by Lemma 2 (b) and (3), we see that for any other inverter  $\mathbb{B}$  of  $F$  there exists a constant  $c_{\mathbb{B}}$  such that for all  $y$  in the image of  $F$

$$t_{\mathbb{I}_{\text{opt}}}(y) \leq c_{\mathbb{B}} \cdot (t_{\mathbb{B}}(y) + t_{\mathbb{F}}(\mathbb{B}(y)) + |y|)^d.$$

In this sense  $\mathbb{I}_{\text{opt}}$  is an optimal inverter of  $F$ .

## 4 Algorithms and arithmetical theories

To talk about algorithms and strings we use *arithmetical formulas*, that is, first-order formulas in the language  $L_{\text{PA}} := \{+, \cdot, 0, 1, <\}$  of Peano Arithmetic. Arithmetical sentences are *true (false)* if they hold (do not hold) in the standard  $L_{\text{PA}}$ -model. For a natural number  $n$  let  $\dot{n}$  denote the natural  $L_{\text{PA}}$ -term without variables denoting  $n$  (in the standard model).

Recall that an arithmetical formula is  $\Delta_0$  if all quantifiers are bounded and it is  $\Sigma_1$  if it has the form  $\exists x_1 \dots \exists x_m \psi$  where  $\psi$  is  $\Delta_0$ .

We shall use a  $\Delta_0$ -formula

$$\text{Run}(u, x, y, z)$$

that defines (in the standard model) the set of tuples  $(u, x, y, z)$  such that  $u$  is an algorithm that on input  $x$  outputs  $y$  by the (code of a complete finite) run  $z$ ; recall that we do not distinguish algorithms from their codes by strings and strings from their codes by natural numbers.

*For the rest of this paper we fix a  $Q_0 \subseteq \Sigma^*$  and an algorithm  $\mathbb{A}_0$  deciding  $Q_0$ .*

The formula

$$\begin{aligned} \text{Dec}_{Q_0}(u) := & \forall x \exists y \exists z \text{Run}(u, x, y, z) \wedge \\ & \forall x \forall y \forall y' \forall z \forall z' ((\text{Run}(\mathbb{A}_0, x, y, z) \wedge \text{Run}(u, x, y', z')) \rightarrow y = y') \end{aligned}$$

defines the set of algorithms deciding  $Q_0$ .

Let  $L_{\text{all}}$  with  $L_{\text{PA}} \subset L_{\text{all}}$  be a language containing countably many function and relation symbols of every arity  $\geq 1$  and countably many constants. A *theory* is a set  $T$  of first-order  $L_{\text{all}}$ -sentences.

**Definition 4.** Let  $T$  be a theory.

- (a) An algorithm  $\mathbb{A}$  *T-provably decides*  $Q_0$  if  $T$  proves  $\text{Dec}_{Q_0}(\mathbb{A})$ .  
(b)  $T$  is *sound for  $Q_0$ -decision* means that for every algorithm  $\mathbb{A}$

if  $\mathbb{A}$   $T$ -provably decides  $Q_0$ , then  $\mathbb{A}$  decides  $Q_0$ .

- (c)  $T$  is *complete for  $Q_0$ -decision* means that for every algorithm  $\mathbb{A}$

if  $\mathbb{A}$  decides  $Q_0$ , then  $\mathbb{A}$   $T$ -provably decides  $Q_0$ .

For a computably enumerable sound theory  $T$  that proves  $\text{Dec}_{Q_0}(\mathbb{A}_0)$  the set

$$D(T) := \{\mathbb{D} \mid \mathbb{D} \text{ } T\text{-provably decides } Q_0\} \quad (4)$$

is a computably enumerable set of algorithms deciding  $Q_0$ . Thus, by Lemma 2 for  $D = D(T)$  we get an algorithm  $\mathbb{A}$  deciding  $Q_0$  as fast as every algorithm in  $D(T)$ . If in addition  $T$  is complete for  $Q_0$ -decision, then  $D(T)$  would be the set of all algorithms deciding  $Q_0$  and thus  $\mathbb{A}$  would be an optimal algorithm for  $Q_0$ . So, the problem  $Q_0$  would have an optimal algorithm if we can find a computably enumerable theory that is both sound and complete for  $Q_0$ -decision. Unfortunately, there is no such theory as shown by the following proposition. We relax these properties in Definition 6 and show in Theorem 7 that the new ones are appropriate to characterize problems with optimal algorithms.

**Proposition 5.** *There is no computably enumerable theory that is sound and complete for  $Q_0$ -decision.*

*Proof.* We assume that there is a computably enumerable theory  $T$  that is sound and complete for  $Q_0$ -decision and derive a contradiction by showing that then the halting problem for Turing machines would be decidable.

For every Turing machine  $\mathbb{M}$  we consider two algorithms. On every input  $x \in \Sigma^*$  the first algorithm  $\mathbb{B}_0(\mathbb{M})$  first checks whether  $x$  codes a run of  $\mathbb{M}$  accepting the empty input tape and then it simulates  $\mathbb{A}_0$  on  $x$  (recall  $\mathbb{A}_0$  is the fixed algorithm deciding  $Q_0$ ). If  $x$  codes an accepting run, then  $\mathbb{B}_0(\mathbb{M})$  reverses the answer  $\mathbb{A}_0(x)$  of  $\mathbb{A}_0$  on  $x$ , otherwise it outputs exactly  $\mathbb{A}_0(x)$ . Clearly  $\mathbb{B}_0(\mathbb{M})$  decides  $Q_0$  if and only if  $\mathbb{M}$  does not halt on the empty input tape.

The second algorithm  $\mathbb{B}_1(\mathbb{M})$ , on every input  $x \in \Sigma^*$  first checks exhaustively whether  $\mathbb{M}$  halts on the empty input tape; if eventually it finds an accepting run, then it simulates  $\mathbb{A}_0$  on  $x$  and outputs accordingly. It is easy to verify that  $\mathbb{B}_1(\mathbb{M})$  decides  $Q_0$  if and only if  $\mathbb{M}$  halts on the empty input tape.

As  $T$  is sound for  $Q_0$ -decision, it proves at most one of  $\text{Dec}_{Q_0}(\mathbb{B}_0(\mathbb{M}))$  and  $\text{Dec}_{Q_0}(\mathbb{B}_1(\mathbb{M}))$ , and as it is complete for  $Q_0$ -decision it proves at least one of these sentences. Hence, given  $\mathbb{M}$ , by enumerating the  $T$ -provable sentences we can decide whether  $\mathbb{M}$  halts on the empty input tape.  $\square$

**Definition 6.** A theory  $T$  is *almost complete for  $Q_0$ -decision* if for every algorithm  $\mathbb{A}$  deciding  $Q_0$  there is an algorithm  $T$ -provably deciding  $Q_0$  that is as fast as  $\mathbb{A}$ .

**Theorem 7.** *The following are equivalent for  $Q_0 \subseteq \Sigma^*$ :*

- (i)  $Q_0$  has an optimal algorithm;
- (ii) There is a computably enumerable and arithmetical theory  $T$  that is sound and almost complete for  $Q_0$ -decision.

*Proof.* (i)  $\Rightarrow$  (ii): We set  $T := \{\text{Dec}_{Q_0}(\dot{\mathbb{A}})\}$  where  $\mathbb{A}$  is an optimal algorithm for  $Q_0$ . Then  $T$  is a computably enumerable true arithmetical theory. Truth implies soundness and almost completeness follows from the optimality of  $\mathbb{A}$ .

(ii)  $\Rightarrow$  (i): Let  $T$  be as in (ii). Then the set  $D(T)$  defined by (4) is a computably enumerable set of algorithms deciding  $Q_0$  (by soundness). By Lemma 2 for  $D = D(T)$  we get an algorithm  $\mathbb{A}$  deciding  $Q_0$  as fast as every algorithm in  $D(T)$  and hence by almost completeness as fast as any algorithm deciding  $Q_0$ . Thus,  $\mathbb{A}$  is an optimal algorithm for  $Q_0$ .  $\square$

A result related to the implication (ii)  $\Rightarrow$  (i) is shown by Sadowski in [7]. He shows assuming that there does not exist an almost optimal algorithm for the set TAUT of all propositional tautologies, that for every theory  $T$  there exists a subset of TAUT in PTIME which is not  $T$ -provably in PTIME (cf. [7, Definition 7.5]).

## 5 Proof of Theorem 1

Recall that  $Q_0 \subseteq \Sigma^*$  and that  $\mathbb{A}_0$  is an algorithm deciding  $Q_0$ . A theory  $T$  is  $\Sigma_1$ -complete if every true arithmetical  $\Sigma_1$ -sentence is provable in  $T$ . The following result is a consequence of Lemma 2.

**Lemma 8.** *Assume  $Q_0 \notin \text{PTIME}$ . Let  $T$  be a computably enumerable  $\Sigma_1$ -complete theory such that  $T$  proves  $\text{Dec}_{Q_0}(\dot{\mathbb{A}}_0)$ . Then there is an algorithm  $\mathbb{A}$  such that:*

- (a) *The algorithm  $\mathbb{A}$  is total (i.e.,  $t_{\mathbb{A}}(x) < \infty$  for all  $x \in \Sigma^*$ ) and as fast as every algorithm  $T$ -provably deciding  $Q_0$ ;*
- (b)  *$T$  is consistent if and only if  $\mathbb{A}$  decides  $Q_0$ .*

*Moreover, there is a computable function  $\text{diag}$  that maps any algorithm  $\mathbb{E}$  enumerating some  $\Sigma_1$ -complete theory  $T$  proving  $\text{Dec}_{Q_0}(\dot{\mathbb{A}}_0)$  to an algorithm  $\mathbb{A}$  with (a) and (b).*

*Proof.* For an algorithm  $\mathbb{B}$  let  $\mathbb{B}\|\mathbb{A}_0$  be the algorithm that on input  $x \in \Sigma^*$  runs  $\mathbb{B}$  and  $\mathbb{A}_0$  on  $x$  in parallel and returns the first answer obtained. Then

$$t_{\mathbb{B}\|\mathbb{A}_0} \leq O\left(\min\{t_{\mathbb{B}}, t_{\mathbb{A}_0}\}\right). \quad (5)$$

*Claim 1.* If  $T$  is consistent and proves  $\text{Dec}_{Q_0}(\dot{\mathbb{B}})$ , then  $\mathbb{B}\|\mathbb{A}_0$  decides  $Q_0$ .

*Proof of Claim 1:* By contradiction, assume that  $T$  is consistent, proves  $\text{Dec}_{Q_0}(\dot{\mathbb{B}})$  and  $\mathbb{B}\|\mathbb{A}_0$  does not decide  $Q$ . Then  $\mathbb{B}\|\mathbb{A}_0$  and  $\mathbb{A}_0$  differ on some input  $x \in \Sigma^*$ .

Thus  $t_{\mathbb{B}}(x) \leq t_{\mathbb{A}_0}(x)$  and in particular  $\mathbb{B}$  halts on  $x$ . Therefore, the following  $\Sigma_1$ -sentence  $\varphi$  is true

$$\varphi := \exists x \exists y \exists y' \exists z \exists z' (\text{Run}(\mathbb{A}_0, x, y, z) \wedge \text{Run}(\mathbb{B}, x, y', z') \wedge \neg y = y').$$

By  $\Sigma_1$ -completeness  $T$  proves  $\varphi$ . However,  $\varphi$  logically implies  $\neg \text{Dec}_{Q_0}(\mathbb{B})$  and thus  $T$  is inconsistent, a contradiction.  $\dashv$

The set

$$D_1(T) := \left\{ \mathbb{B} \parallel \mathbb{A}_0 \mid T \text{ proves } \text{Dec}_{Q_0}(\mathbb{B}) \right\}$$

is nonempty as  $\mathbb{A}_0 \parallel \mathbb{A}_0 \in D_1(T)$  by assumption. Let  $\mathbb{A}$  be the algorithm obtained for  $D = D_1(T)$  by Lemma 2. Then the statement (a) of our lemma holds by (5) and Lemma 2 (b).

For consistent  $T$ , by Claim 1 the set  $D_1(T)$  only contains algorithms deciding  $Q_0$ , thus  $\mathbb{A}$  decides  $Q_0$  by Lemma 2.

If  $T$  is inconsistent, let  $\mathbb{B}_{\text{bad}}$  be an algorithm that accepts every input in the first step. Then  $\mathbb{B}_{\text{bad}} \parallel \mathbb{A}_0 \in D_1(T)$  by inconsistency of  $T$ . Thus, by Lemma 2 (b), the algorithm runs in polynomial time and thus does not decide  $Q_0$ .

As from an algorithm enumerating  $T$  we effectively get an algorithm enumerating  $D_1(T)$ , by Lemma 2 it should be clear that a computable function diag as claimed exists.  $\square$

**Remark 9.** As the preceding proof shows we only need the assumption  $Q_0 \notin \text{PTIME}$  in the proof of the implication from right to left in (b).

*Proof of Theorem 1:* Let  $Q_0$  be a decidable problem not in PTIME. Among others, the finite true arithmetical theory  $T_0$  claimed to exist in Theorem 1 will contain a formalization of Lemma 8.

We choose a  $\Sigma_1$ -formula  $\text{Prov}(x, y)$  defining (in the standard model) the set of pairs  $(m, n)$  such that algorithm  $m$  enumerates a theory<sup>5</sup> that proves the sentence  $n$ .

We let

$$\text{Con}(x) := \neg \text{Prov}(x, \ulcorner \neg 0 = 0 \urcorner)$$

(here  $\ulcorner \varphi \urcorner$  denotes the Gödel number of  $\varphi$ ). If  $\mathbb{E}$  enumerates a theory  $T$ , we write  $\text{Con}_T$  for  $\text{Con}(\mathbb{E})$ .<sup>6</sup>

Let  $f : \mathbb{N} \rightarrow \mathbb{N}$  be the function given by

$$f(m) := \ulcorner \text{Dec}_{Q_0}(\dot{m}) \urcorner.$$

Both, this function  $f$  and the function diag, from Lemma 8 are computable and hence,  $\Sigma_1$ -definable in the standard model. For better readability we shall use  $f$  and diag like function symbols in arithmetical formulas.

<sup>5</sup> We may assume that every enumeration algorithm enumerates a theory by deleting those printed strings that are not sentences.

<sup>6</sup> The notation is ambiguous, as the definition depends on the choice of  $\mathbb{E}$ , however not the arguments to follow.



Further, let the arithmetical formula  $\text{As-fast-as}(x, y)$  define the pairs  $(n, m)$  such that algorithm  $n$  is as fast as algorithm  $m$  and let  $\text{Ptime}(x)$  define the set of polynomial time algorithms. Finally, we set

$$\text{Afap}(x, y) := \forall z (\text{Prov}(x, f(z)) \rightarrow \text{As-fast-as}(y, z)).$$

Then for an algorithm  $\mathbb{E}$  enumerating a theory  $T$  the statement “the algorithm  $F(T)$  is as fast as any algorithm  $T$ -provably deciding  $Q_0$ ,” that is, the statement (a) in Theorem 1 is formalized by the sentence

$$\text{Afap}(\dot{\mathbb{E}}, F(\dot{T})). \quad (6)$$

Recall that Robinson introduced a finite,  $\Sigma_1$ -complete arithmetical theory  $R$ . Let  $\text{e-Rob}(x)$  be a  $\Sigma_1$ -formula expressing that the algorithm  $x$  enumerates a theory extending  $R \cup \{\text{Dec}_{Q_0}(\dot{A}_0)\}$ .

We now define the theory  $T_0$ . It extends  $R$  by the following sentences (s1)–(s5):

- (s1)  $\forall x (\text{e-Rob}(x) \rightarrow \text{Afap}(x, \text{diag}(x)))$ ,  
(a formalization of Lemma 8 (a))
- (s2)  $\forall x ((\text{Con}(x) \wedge \text{e-Rob}(x)) \rightarrow \text{Dec}_{Q_0}(\text{diag}(x)))$ ,  
(a formalization of part of Lemma 8 (b))
- (s3)  $\forall x (\text{Ptime}(x) \rightarrow \neg \text{Dec}_{Q_0}(x))$ ,  
( $Q_0$  is not in PTIME)
- (s4)  $\forall x (\neg \text{Con}(x) \rightarrow \forall y (\text{Sent}(y) \rightarrow \text{Prov}(x, y)))$   
(every inconsistent theory proves every sentence; here  $\text{Sent}(y)$  is a  $\Delta_0$ -formula defining the first-order  $L_{\text{all}}$ -sentences)
- (s5)  $\forall x \forall y ((\text{As-fast-as}(x, y) \wedge \text{Ptime}(y)) \rightarrow \text{Ptime}(x))$   
(if algorithm  $x$  is as fast as the polynomial algorithm  $y$ , then it is polynomial too).

Let  $T$  be a computably enumerable extension of  $T_0$  and let  $\mathbb{E}$  be an algorithm enumerating  $T$ . We claim that for the algorithm

$$F(T) := \text{diag}(\mathbb{E})$$

(see Lemma 8) the statements (a) and (b) of Theorem 1 hold.

The arithmetical sentence  $F(\dot{T}) = \text{diag}(\dot{\mathbb{E}})$  is  $\Sigma_1$  and true, so  $T_0$  proves it by  $\Sigma_1$ -completeness (as  $T_0 \supseteq R$ ). By the same reason,  $T_0$  proves  $\text{e-Rob}(\dot{\mathbb{E}})$ . As  $T_0$  contains (s1),  $T_0$  proves  $\text{Afap}(\dot{\mathbb{E}}, F(\dot{T}))$ ; that is,  $T_0$  proves that  $F(T)$  is as fast as any algorithm  $T$ -provably deciding  $Q_0$ . Thus (a) in Theorem 1 holds.

We turn to (b). Let  $T^*$  be a theory with  $T^* \supseteq T$ .

(i)  $\Rightarrow$  (ii): So, we assume that  $T^*$  proves  $\text{Con}_T$ . We already know that  $T_0$ , and hence  $T^*$ , proves  $\text{e-Rob}(\dot{\mathbb{E}})$ . As  $T^*$  contains (s2), for  $x = \dot{\mathbb{E}}$  we see that  $T^*$  proves  $\text{Dec}_{Q_0}(\text{diag}(\dot{\mathbb{E}}))$  and thus  $\text{Dec}_{Q_0}(F(\dot{T}))$ ; that is,  $F(T)$   $T^*$ -provably decides  $Q_0$ .

(ii)  $\Rightarrow$  (iii): Immediate by part (a) of the theorem.

(iii)  $\Rightarrow$  (i): Let  $\mathbb{A}$  be an algorithm such that  $T^*$  proves  $\text{Dec}_{Q_0}(\dot{\mathbb{A}})$  and  $\text{Afap}(\dot{\mathbb{E}}, \dot{\mathbb{A}})$ ; the latter means that  $T^*$  proves

$$\forall z(\text{Prov}(\dot{\mathbb{E}}, f(z)) \rightarrow \text{As-fast-as}(\dot{\mathbb{A}}, z)). \quad (7)$$

Let  $\mathbb{B}$  be an algorithm such that  $T^*$  proves

$$\text{Ptime}(\dot{\mathbb{B}}). \quad (8)$$

Then  $T^*$  proves the following implications:

$$\begin{aligned} \neg\text{Con}_T &\rightarrow \text{Prov}(\dot{\mathbb{E}}, f(\dot{\mathbb{B}})) && \text{(by (s4) and as } \text{Sent}(f(\dot{\mathbb{B}})) \text{ is } \Sigma_1) \\ \neg\text{Con}_T &\rightarrow \text{As-fast-as}(\dot{\mathbb{A}}, \dot{\mathbb{B}}) && \text{(by (7))} \\ \neg\text{Con}_T &\rightarrow \text{Ptime}(\dot{\mathbb{A}}) && \text{(by (8) and (s5))} \\ \neg\text{Con}_T &\rightarrow \neg\text{Dec}_{Q_0}(\dot{\mathbb{A}}) && \text{(by (s3)).} \end{aligned}$$

As  $T^*$  proves  $\text{Dec}_{Q_0}(\dot{\mathbb{A}})$ , we see that  $T^*$  proves  $\text{Con}_T$ .  $\square$

We close with an application to Zermelo-Fraenkel set theory ZFC. Here we add the usual ZFC-definitions of the symbols of  $L_{\text{PA}}$  as new axioms.

**Corollary 10.** *Assume ZFC is consistent. Then there exist a problem  $Q$  and an algorithm  $\mathbb{A}$  satisfying (a) and (b).*

- (a) *There is no algorithm deciding  $Q$  and being as fast as every other algorithm deciding  $Q$ .*
- (b) *The algorithm  $\mathbb{A}$  decides  $Q$  and is as fast as any algorithm that ZFC-provably decides  $Q$ .*

*Proof.* Messner [6] proved that there is a problem  $Q$ , even decidable in exponential time, that does not have an almost optimal algorithm. In particular, then  $Q$  satisfies (a) and  $Q \notin \text{PTIME}$ . We choose  $\mathbb{A}$  according to Lemma 8 for  $Q_0 := Q$  and  $T := \text{ZFC}$ ; then (b) holds.  $\square$

## Acknowledgments

The authors thank the John Templeton Foundation for its support under Grant #13152, *The Myriad Aspects of Infinity*. Yijia Chen is affiliated with BASICS and MOE-MS Key Laboratory for Intelligent Computing and Intelligent Systems which is supported by National Nature Science Foundation of China (61033002).

## References

1. S. A. Cook and P. Nguyen. *Logical Foundations of Proof Complexity*. Cambridge University Press, 2010.

2. J. Hartmanis. Relations between diagonalization, proof systems, and complexity gaps. *Theoretical Computer Science*, 8:239–253, 1979.
3. M. Hutter. The fastest and shortest algorithm for all well-defined problems. *International Journal of Foundations of Computer Science*, 13:431–443, 2002.
4. J. Krajčiček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54:1063–1079, 1989.
5. L. Levin. Universal search problems (in Russian). *Problemy Peredachi Informatsii*, 9:115–116, 1973.
6. J. Messner. On optimal algorithms and optimal proof systems. In *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science (STACS'99)*, Lecture Notes in Computer Science 1563, 361–372, 1999.
7. Z. Sadowski. On an optimal propositional proof system and the structure of easy subsets. *Theoretical Computer Science*, 288:181–193, 2002.