The "Hardest" Natural Decidable Theory *

Sergei Vorobyov
Max-Planck Institut für Informatik
Im Stadtwald, Saarbrücken, D-66123, Germany
sv@mpi-sb.mpg.de

Abstract

We prove that any decision procedure for a modest fragment of L. Henkin's theory of pure propositional types [7, 12, 15, 11] requires time exceeding a tower of 2's of height ***exponential*** *in the length of input. Until now the highest known lower bounds for natural decidable theories were* ***at most linearly high*** *towers of 2's and since mid-seventies it was an open problem whether natural decidable theories requiring more than that exist [12, 2]. We give the affirmative answer. As an application of this today's strongest lower bound we improve known and settle new lower bounds for several problems in the simply typed lambda calculus.*

1. Introduction

In his survey paper [12] A. Meyer mentioned (p. 479), as a curious empirical observation, that all known natural decidable non-elementary problems require at most (upper bound)

$$F(1,n) = \exp_\infty(n) = 2^{2^{\cdot^{\cdot^{2}}}} \Big\}n$$

Turing machine steps on inputs of length n to decide[1].

Until now the highest known lower bounds for natural decidable theories were at most $\exp_\infty(cn)$ with *"only" linearly high* towers of 2's. Since mid-seventies it was an open problem whether natural decidable theories requiring more than $\exp_\infty(cn)$ exist [12, 2].

In this paper we give the *affirmative* answer by presenting a natural decidable theory (due to L. Henkin [7] and mentioned by A. Meyer [12]) and proving that *every* decision procedure for that theory should necessarily make more than

$$F(1,2^{cn}) = \exp_\infty(2^{cn}) = 2^{2^{\cdot^{\cdot^{2}}}} \Big\} \text{height } 2^{cn}$$

steps on infinitely many inputs of length n, where the height of the tower of 2's is *exponential* in the length of input, 2^{cn} for some $c > 0$ (i.e., is *not linearly bounded*).

K. Compton and C. Henson [2] developed a powerful and easy-to-use technique for proving lower bounds for the decision complexity of logical theories, based on encoding of large binary relations. By using their technique they obtained new concise proofs for virtually all known before lower bounds for logical theories in a uniform way. As an *open Problem* 10.13 (p. 75) Compton and Henson asked *whether there exist "natural" decidable theories with lower bounds of the form NTIME*$(\exp_\infty(f(n)))$, where $f(n)$ is *not linearly bounded*, i.e., $f(n) \leq cn$ does not hold for any constant c. Of course, by using time hierarchy theorems it is easy to construct non-natural decidable theories of arbitrary complexity.

In this paper, by using Compton-Henson's techniques, we show that the rudimentary set theory Ω over the universe $\cup_{i\in\omega}\mathcal{D}_i$, where $\mathcal{D}_0 = \{0,1\}$ and $\mathcal{D}_{i+1} = \mathit{powerset}(\mathcal{D}_i)$, a modest fragment of L. Henkin's theory of propositional types [7], mentioned by A. Meyer in [12] (Theorem pp. 478–479, no. 7) and used by R. Statman [15] to prove that the typed lambda calculus is not elementary recursive, in fact requires (lower bound) time $\exp_\infty(2^{cn})$ to decide, thus giving the affirmative answer to Compton-Henson's problem.

The earlier known lower bound for Ω was "only" $F(n, \varepsilon \cdot \log(n))$ due to M. Fischer and A. Meyer [12] (Theorem pp. 478–479, no. 7; the proof, however, was apparently never published).

*By using hierarchy theorems it is easy to construct "unnatural" decidable theories of arbitrary complexity.

[1] The m-story exponential function $F(n,m)$ is defined by: $F(n,0) = n$ and $F(n,m+1) = 2^{F(n,m)}$. The *tower of n twos* function is defined by $\exp_\infty(n) = F(1,n)$.

1043-6871/97 $10.00 © 1997 IEEE

1.1. Applications

We give several applications of this today's strongest lower bound $\exp_\infty(2^{cn})$ to improve known or settle new lower bounds for other *natural* problems related to the simply typed lambda calculus Λ by reduction.

First, we demonstrate that deciding β- and $\beta\eta$-equality in the simply typed lambda calculus Λ, known to be *not elementary recursive* (R. Statman [15]), in fact requires time

$$F(1,cn) = \exp_\infty(cn) = \left.2^{2^{\cdot^{\cdot^{2}}}}\right\}cn ,$$

as opposed to a far less impressive

$$F(n, c\log(\log(n))) = \left.2^{2^{\cdot^{\cdot^{2^n}}}}\right\} c\log(\log(n))$$

implicit in [15, 11]. Our improved lower bound $\exp_\infty(cn)$ matches (with a different constant) with the known $\exp_\infty(dn)$ upper bound for $\beta(\eta)$-equality in the simply typed lambda calculus Λ due to W. Tait [18, 6]. Moreover, the known $\exp_\infty(cn)$ lower bound for any normalization algorithm in Λ (H. Schwichtenberg [14]) immediately follows from our $\exp_\infty(cn)$ lower bound for $\beta(\eta)$-equality. This lower bound did not follow from Statman's result [15].

Second, we settle the new

$$F(1, cn/\log(n)) = \exp_\infty(cn/\log(n)) = \left.2^{2^{\cdot^{\cdot^{2}}}}\right\}cn/\log(n) ,$$

lower bound for a long-standing, today still *open, higher-order matching problem* in the simply typed lambda calculus due to G. Huet [9]. The problem consists in deciding, given a term t of type $\sigma_1 \to \ldots \to \sigma_n \to \tau$ and a term u of type τ (both in normal forms), whether there exist terms s_i of types σ_i $(1 \leq i \leq n)$ such that $ts_1 \ldots s_n =_{\beta\eta} u$. This gives an example asked for by Compton and Henson in *Problem* 10.11 from [2], p. 75: *Give nontrivial lower bounds for mathematically interesting theories whose decidability is still open.* There were no nontrivial known lower bounds for higher-order matching, except NP-hardness (D. Wolfram [21], p. 78). The best lower bound one could get for the higher-order matching problem would be only $F(n, c\log(\log(n/\log(n))))$ without our new strong lower bound for Ω.

Third, we reinforce the lower bound in Statman's analog of Rice's Theorem 6 ([17], p. 25) for the simply typed lambda calculus. This theorem asserts that any nontrivial set of closed terms of the same type closed under $\beta\eta$-equality is not elementary recursive. We improve it by replacing "not elementary recursive" with "requiring time exceeding $\exp_\infty(cn)$. Again, "only" $F(n, c\log(\log(n)))$ followed from previous results.

1.2. Practical Relevance of the Obtained Lower Bounds

It is by now commonly accepted that lower and upper complexity bounds are important in development and analysis of efficient algorithms, especially when large problem instances are involved. This becomes more and more crucial nowadays in connection with large-scale theorem proving, program verification, constraint logic and functional programming projects.

One may ask whether there exist any reasons in improving lower bounds for theories once they are already known to be not elementary recursive. The answer is clearly *yes*, because non-elementary does not necessarily mean bad.

Consider a non-elementary problem, which happens to have a $F(n, \log^6(n))$ lower bound (where by $\log^k(n)$ we denote the k-fold application of log, and not its k-th power). Such a problem may have a fascinatingly efficient and practical decision algorithm, since up to inputs of size[2] $\exp_\infty(5) = 2^{65536}$ the function $F(n, \log^6(n))$ is majorated by n.

Thus, the mere fact of being non-elementary is not at all informative, since any problem requiring time $F(n, log^k(n))$ is non-elementary. As we saw, even for small $k = 6$ this is practically immaterial. In this respect the previously known lower bound $F(n, c\log(\log(n)))$ for $\beta\eta$-equality in the simply typed lambda calculus was not very convincing: if $c = 1/16$ then $F(n, c\log(\log(n))) \leq 2^n$ for $n \leq \exp_\infty(5) = 2^{65536}$. The new $\exp_\infty(cn)$ lower bound for Λ is much more convincing: for $c = 1/16$, $\exp_\infty(cn) \geq 2^{65536}$ for $n > 80$. The same applies to the new lower bound $\exp_\infty(cn/\log(n))$ for the higher-order matching problem, which is only a little bit worse (i.e., better) than $\exp_\infty(cn)$.

The last common objection necessary to dispel is as follows. One might argue that despite a strong lower bounds like $\exp_\infty(2^{cn})$ or $\exp_\infty(cn)$, the sizes of inputs n on which any decision algorithm for a problem starts to exhibit its desperately bad behavior are very large and never occur in practice. However, M. Fischer and M. Rabin [5] observed that the lengths of inputs n, at which an algorithm AL starts to demonstrate its

[2] big enough, from the practical viewpoint: one light-year is approximately 2^{60} cm; the radius of the known universe is approximately 11×10^9 light years, or approximately $10^{28} \approx 2^{93}$ cm; one can hardly encounter formulas of this length in practice.

worst-case behavior is comparable with the size of the algorithm, $O(|AL|)$. Thus our lower bounds enter quite early in the game and cannot be neglected.

Outline of the Paper. After a background material on Compton-Henson's lower bounds techniques we settle the $\exp_\infty(2^{cn})$ lower bound for Ω, and then proceed to lambda calculus, higher-order matching, and Statman-Rice's theorem.

2. Type Theory Ω

Type theory Ω is a very rudimentary fragment of L. Henkin's theory of propositional types [7].

Definition 1 (Theory Ω, [15]) . *The language of* type theory Ω *is a language of set theory where every variable has a natural number type (written as a binary superscript) and there are two constants* $\mathbf{0}$, $\mathbf{1}$ *of type* 0. *The atomic formulas of* Ω *are* stratified, *i.e., have form* $\mathbf{0} \in x^1$, *or* $\mathbf{1} \in x^1$, *or* $x^n \in y^{n+1}$. *All other formulas are built as always, by using* $\neg$, $\wedge$, *and* $\forall$.

The interpretation of Ω *is as follows:* $\mathbf{0}$ *denotes* 0, $\mathbf{1}$ *denotes* 1, *and* x^n *ranges over* $\mathcal{D}_n$, *where* $\mathcal{D}_0 = \{0, 1\}$ *and* $\mathcal{D}_{n+1} = \textit{powerset}(\mathcal{D}_n)$. □

Convention 2 . For the complexity considerations below let us fix any reasonable encoding of formulas of Ω as binary strings. Let us agree that a variable of Ω is represented by its type and its identification number within a type, both written in *binary.* □

Remark 3 . A more wasteful way would be to write types and identification numbers of variables in *unary* notation. The papers [12, 15, 11] do not explicitly fix any choice of binary or unary notation for types in their complexity claims about Ω. Most probably [12] subsumes binary notation (see page 478). For the moment we stick to the binary notation, and return back to this question later on in Section 8. □

Definition 4 (Iterated Exponentials) . *Define the function* $\exp_\infty : \omega \to \omega$ *recursively as* $\exp_\infty(0) = 1$ *and* $\exp_\infty(k+1) = 2^{\exp_\infty(k)}$.

Define the m-story exponential function $F(n,m)$ *by:* $F(n,0) = n$ *and* $F(n,m+1) = 2^{F(n,m)}$. *Notice that* $\exp_\infty(n) = F(1,n)$. □

Definition 5 ((Non-)Elementary Problems) . *A problem is called* elementary recursive *iff it belongs to the complexity class* $NTIME(F(n,m))$ *for some (fixed)* $m \in \omega$, *and* non-elementary *otherwise.* □

Decidability. The validity problem for Ω is decidable, because every quantifier runs over a *finite* domain, but *is not elementary recursive.* Even stronger:

Theorem 6 (Fischer, Meyer [12], pp. 478–479) *Any Turing machine deciding* Ω *requires a number of steps exceeding*

$$F(|S|, \varepsilon \cdot \log(|S|)) \qquad (1)$$

for some constant $\varepsilon > 0$ *and infinitely many sentences* S *of* Ω. □

In this paper we considerably improve this lower bound to

Theorem 7 (Improved Lower Bound for Ω) . *Any Turing machine deciding* Ω *requires a number of steps exceeding*

$$\exp_\infty(2^{cn}) = 2^{2^{\cdot^{\cdot^{2}}}} \Big\} \ \textit{height}\ 2^{cn} \qquad (2)$$

for some constant $c > 0$ *and infinitely many sentences of* Ω *of length* n. □

Remark 8 . *This gives the first example of a natural decidable theory that requires time exceeding* $\exp_\infty(f(n))$, *where* $f(n)$ *is not linearly bounded.* □

Remark 9 . We do not know, however, whether the lower bound from Theorem 6 applies to Statman's or to Henkin's formulation of Ω. The paper by Fischer and Meyer with the proof of Theorem 6 was apparently never published. In any case, Theorem 7 gives a much stronger lower bound than Theorem 6, for a weak formulation of Ω given in Definition 1. □

3. Lower Bounds Techniques

We now briefly describe the lower bounds techniques we apply to prove our main Theorem 7.

In 1936 L. Kalmar proved that the first-order theory of a binary relation is undecidable, which greatly simplified undecidability proofs, as compared to those based on straightforward encodings of Turing machines, see, e.g., M. Rabin [13].

B. Trakhtenbrot [19] and later R. Vaught [20] proved even stronger

Theorem 10 . *Let* $\mathcal{L}$ *be the first-order language with the unique binary relation symbol. The set of valid sentences of* $\mathcal{L}$ *and the set of sentences of* $\mathcal{L}$ *refutable by some finite model are recursively inseparable.* □

Two sets are recursively inseparable iff there are no recursive sets containing one and disjoint with the other. Notice that recursive inseparability is stronger than simple undecidability: both of recursively inseparable sets are undecidable. Usually Theorem 10 is applied in conjunction with the method of interpretations, extensively discussed in [13, 4].

Recently, K. Compton and C. Henson [2] refined the above inseparability idea for proving hereditary lower complexity bounds for logical theories. Compton-Henson's method is based on interpretations, in a given theory, of *all finite binary relations up to a certain size, by means of short formulas.* The larger are binary relations interpretable in a theory, the higher is the lower bound for its decision complexity. Compton-Henson's method gives a considerable simplification over the pioneer lower bounds methods due to Meyer, Stockmeyer, Fischer, and Rabin, based on direct encodings of Turing machine computations.

More precisely, Compton and Henson proved the following theorem. In fact, they proved much stronger and more subtle theorems, but the variant below (we derive by simplification from their Theorem 6.2, p. 38) is enough for our purposes. We give the necessary definition of the special reducibility used right after the statement of the theorem. The *size of a binary relation* is the number of elements in its domain.

Theorem 11 (Compton-Henson) . *Let T be a theory, M be its model, and $t(n)$ be a time resource bound such that for some $0 < d < 1$ one has*

$$\lim_{n\to\infty} t(dn) \,/\, t(n) = 0.$$

Suppose there exists a sequence of formulas of T

$$\delta_n(x,p),\ \gamma_n(x,y,p) \quad \textit{for } n \in \omega, \tag{3}$$

reset log-lin-computable from n written in unary notation such that for every binary relation R of size $t(n)$ there exists an element $m \in M$ such that the model

$$\Big\langle \underbrace{\{x \mid M \models \delta_n(x,m)\}}_{\textit{carrier}};\ \underbrace{\{\langle x,y\rangle \mid M \models \gamma_n(x,y,m)\}}_{\textit{binary relation}} \Big\rangle$$

(consisting of elements x satisfying $\delta_n(x,m)$ with the binary relation on them defined by $\gamma_n(x,y,m)$) is isomorphic to R.

Then for some constant $c > 0$ the theory T requires nondeterministic time exceeding $t(cn)$ to decide, or, equivalently, $T \notin NTIME(t(cn))$. □

Compton and Henson define (pp. 10–11) and use a special kind of reducibility, which they call the *reset log-lin reducibility.* A machine performing such a reduction is a *log-space*, *linear time bounded* Turing machine with input tape, output tape, and auxiliary work tapes (log-space bounded). It operates as the usual Turing machine, but has the capability to reset the input tape head to the initial input cell on k moves during a computation, where k is *fixed for all inputs* (This differs from the standard log-lin reducibility, computable in logarithmic space and linearly bounded.)

We apply Compton-Henson's Theorem 11 for the time resource bound $t(n) = \exp_\infty(2^n)$ and the theory Ω in the next sections. This will prove our main Theorem 7. Obviously, all iterated exponentials satisfy the precondition of Theorem 11.

4. Extending Ω by Explicit Definitions

We extend the language of Ω by adding explicitly definable predicates for the usual set-theoretic notions. These are needed to speak succinctly about binary relations within Ω, as required by Compton-Henson's Theorem 11.

4.1. Subset.

For every $n \in \omega\backslash\{0\}$ explicitly define the binary predicate $Subset(x^n, y^n)$ meaning $x^n \subseteq y^n$ as follows:

$$Subset(x^n, y^n) \ \equiv_{df}$$

$$\equiv_{df}\ \forall z^{n-1}(z^{n-1} \in x^n \Rightarrow z^{n-1} \in y^n). \tag{4}$$

4.2. Singleton.

For every $n \in \omega$ explicitly define the binary predicate $Singleton(x^n, y^{n+1})$ meaning $y^{n+1} = \{x^n\}$ as follows:

$$Singleton(x^n, y^{n+1}) \ \equiv_{df}\ x^n \in y^{n+1} \wedge$$

$$\wedge \forall z^{n+1}(x^n \in z^{n+1} \Rightarrow y^{n+1} \subseteq z^{n+1}). \tag{5}$$

4.3. Unordered Pair.

For every $n \in \omega$ explicitly define the ternary predicate $U\text{-}Pair(x^n, y^n, z^{n+1})$ meaning $z^{n+1} = \{x^n, y^n\}$ as follows:

$$U\text{-}Pair(x^n, y^n, z^{n+1}) \ \equiv_{df}$$

$$x^n \in z^{n+1} \wedge y^n \in z^{n+1} \wedge$$

$$\wedge \forall w^{n+1} \quad (x^n \in w^{n+1} \wedge y^n \in w^{n+1} \Rightarrow \Rightarrow z^{n+1} \subseteq w^{n+1}). \tag{6}$$

Notice that a pair of two equal elements is a one-element set.

4.4. Ordered Pair

Usually, in set theory an ordered pair of x_1, x_2 is defined as $\{x_1, \{x_1, x_2\}\}$. However, this is *impossible in* Ω, where the components of a set should be of the *same type*. Alternatively, in Ω an ordered pair of the elements x_1^n, x_2^n of the *same type* can be defined as $\{\{x_1^n\}, \{x_1^n, x_2^n\}\}$. Obviously, it is an element of type $n+2$. We define formally

$$Pair(x^n, y^n, z^{n+2}) \equiv_{df}$$

$$\exists u^{n+1} v^{n+1} \Big[\; Singleton(x^n, u^{n+1}) \wedge U\text{-}Pair(x^n, y^n, v^{n+1}) \wedge U\text{-}Pair(u^{n+1}, v^{n+1}, z^{n+2}) \Big]. \tag{7}$$

An ordered pair of two equal elements is a one-element set, but it does not prevent us from decoding components of a pair correctly, see below.

4.5. Components of an Ordered Pair

For every $n \in \omega$ and an ordered pair z^{n+2} define its first and second components as follows:

$$Fst(x^n, z^{n+2}) \equiv_{df} \exists y^n \; Pair(x^n, y^n, z^{n+2}), \tag{8}$$
$$Snd(y^n, z^{n+2}) \equiv_{df} \exists x^n \; Pair(x^n, y^n, z^{n+2}). \tag{9}$$

5. Defining a Binary Relation

Say that an element of type $n+3$ *represents a binary relation* iff its every element (of type $n+2$) is an ordered pair of some elements of type n.

The following two predicates in (10) and (11) explicitly define the properties:

- "*x^n is an element in the domain of a binary relation represented by r^{n+3}*",
- "*two elements x^n and y^n are related by the binary relation represented by r^{n+3}*",

as required by Compton-Henson's Theorem 11:

$$\delta(x^n, r^{n+3}) \equiv_{df} \exists z^{n+2} \Big[z^{n+2} \in r^{n+3} \wedge (Fst(x^n, z^{n+2}) \vee Snd(x^n, z^{n+2})) \Big], \tag{10}$$

$$\gamma(x^n, y^n, r^{n+3}) \equiv_{df} \exists z^{n+2} \Big[z^{n+2} \in r^{n+3} \wedge Pair(x^n, y^n, z^{n+2}) \Big]. \tag{11}$$

6. Proof of the Main Claim

We are ready to prove our main claim that for some $c > 0$ the decision problem for the theory Ω requires time $\exp_\infty(2^{cn})$, by applying Compton-Henson's Theorem 11.

Notice that the formulas $\delta(x^n, r^{n+3})$ and $\gamma(x^n, y^n, r^{n+3})$ in (10) and (11) can be explicitly expressed in terms of the basic $\in$ predicate in a *fixed way*, uniform in n (by simply eliminating all defined predicates introduced in Section 4). These definitions use only a *fixed number* of variable occurrences of types n, $n+1$, $n+2$, and $n+3$. As n changes, only variable types (superscripts) change in these explicit definitions of formulas $\delta(x^n, r^{n+3})$ and $\gamma(x^n, y^n, r^{n+3})$. Since we agreed to write types of variables *in binary*, this gives only an $O(\log n)$ growth of the explicit definitions of $\delta(x^n, r^{n+3})$ and $\gamma(x^n, y^n, r^{n+3})$ as n grows. At the same time the sizes of representable binary relations grow as $\exp_\infty(n+1)$. This implies, by Compton-Henson's theorem, the lower bound (2) of our Theorem 7.

More accurately, consider the explicit definitions for the formulas

$$\delta_n \equiv_{df} \delta(x^{2^{n+2}}, r^{2^{n+2}+3}),$$
$$\gamma_n \equiv_{df} \gamma(x^{2^{n+2}}, y^{2^{n+2}}, r^{2^{n+2}+3}).$$

These definitions grow linearly as n grows, and are easily reset log-lin computable from n given in unary notation as follows.

Let n in unary be written on the input tape of the Turing machine. The machine starts writing the explicit definition of δ_n (resp. γ_n), and each time it needs to write a type of a variable, it writes 1 on the output tape, then resets its input tape, reads it and writes 0 to the output tape for every 1 read from input. Afterwards, it writes either 00, or 01, or 10, or 11, depending on whether it needs to write a type of a variable equal to 2^{n+2}, $2^{n+2}+1$, $2^{n+2}+2$, or $2^{n+2}+3$. In this manner it writes exactly 2^{n+2}, $2^{n+2}+1$, $2^{n+2}+2$, or $2^{n+2}+3$ *in binary*, as needed. By construction, this machine is a reset log-lin bounded Turing machine, since the explicit definition of δ_n (resp. γ_n) uses only a fixed number k of variable occurrences. So the machine resets k times independently of an input n.

To conclude the proof we notice that, as the parameter $r^{2^{n+2}+3}$ changes, the formulas δ_n and γ_n represent arbitrary binary relations of size up to $\exp_\infty(2^{n+2}+1)$. Recall that a variable of type k runs over the domain $\mathcal{D}_k$ of size $\exp_\infty(k+1)$. So, as the parameter $r^{2^{n+2}+3}$ changes, the models generated by δ_n and γ_n run over all possible binary relations of size up to $\exp_\infty(2^{n+2}+1)$. It follows, by Compton-Henson's Theorem 11, that the theory Ω does not belong to the complexity class $NTIME(\exp_\infty(2^{cn}))$ for some constant $c > 0$, or, equivalently, any decision procedure for Ω requires nondeterministic time $\exp_\infty(2^{cn})$ for some constant $c > 0$ and infinitely many inputs of size n.

This finishes the proof of the main claim of Theorem 7 and gives the first natural example of a decidable theory requiring time exceeding $\exp_\infty(f(n))$, with $f(n) = 2^{cn}$ being *not linearly bounded.*

7. Upper Bound for Ω

The upper complexity bound for Ω matches its lower bound (of course, with a larger constant):

Proposition 12 . The theory Ω can be decided within time $\exp_\infty(2^{dn})$ for some constant $d > 0$.

Proof. The maximal type of a variable in a sentence Φ of Ω of length n is $k = O(2^n)$. Thus, only the domains $\mathcal{D}_i$ with $i \leq k$ of cardinality $\leq \exp_\infty(k+1) = \exp_\infty(O(2^n))$ should be taken into consideration when deciding Φ. An arbitrary element of a domain $\mathcal{D}_i$ $(i \leq k)$ can be written in space $\exp_\infty(O(2^n))$. The number of quantifiers in Φ is $m = O(n)$. To decide Φ it suffices to cycle through all the m-tuples of elements of these domains. The space $\exp_\infty(O(2^n))$, hence time $2^{\exp_\infty(O(2^n))} = \exp_\infty(O(2^n))$ is enough for that purpose. □

8. Lower Bound for Ω_{unary}

When the variable types in the sentences of Ω are written in *unary*, in contrast to a more economic *binary notation* we adopted (cf., Definition 1, Convention 2, Remark 9), the straightforward modification of the argument from the previous Section yields the following lower bound (still stronger than in Theorem 6; recall that we do not know whether Theorem 6 is formulated for Ω or Ω_{unary}, cf., Definition 1, Convention 2, Remark 9):

Theorem 13 . *Any Turing machine deciding Ω_{unary} requires a number of steps exceeding*

$$\exp_\infty(\varepsilon \cdot |S|)$$

for some constant $\varepsilon > 0$ and infinitely many sentences S of Ω_{unary}.

Proof. As n grows, the sizes of binary relations described by linearly growing $\delta(x^n, r^{n+3})$ and $\gamma(x^n, y^n, r^{n+3})$ in (10) and (11) grow "*only*" as $\exp_\infty(n)$, and not as $\exp_\infty(2^n)$ as in the case of Ω. It is easy to see that $\delta(x^n, r^{n+3})$ and $\gamma(x^n, y^n, r^{n+3})$ in the *unary* case are still reset log-lin computable from n written in unary. Indeed, each time a machine needs to write a type of a variable, it resets its input tape (with n written in unary) copies it and adds, respectively, nothing, 1, 11, or 111, depending on whether it needs to write a type n, $n+1$, $n+2$, or $n+3$. Since the definitions of $\delta(x^n, r^{n+3})$ and $\gamma(x^n, y^n, r^{n+3})$ use only a fixed number k of variable occurrences, the machine resets the fixed number k of times, independently of an input n. □

9. Upper Bound for Ω_{unary}

Similarly to Proposition 12 we get

Proposition 14 . The theory Ω_{unary} can be decided within time $\exp_\infty(dn)$ for some constant $d > 0$. □

10. Simple Presumably Intractable Fragments of Ω

Already the simplest fragments of Ω are *presumably intractable*. Recall that it is still unknown whether the inclusions $P \subseteq NP \subseteq PSPACE$ are proper or not.

Proposition 15 . The class of existentially quantified prenex sentences of Ω with quantifier prefixes containing just one existentially quantified variable of type 1 and all other existentially quantified variables of type 0 is NP-complete.

Proof. Given a propositional formula F in 3-CNF with variables $x_1, \ldots, x_n$, consider the following sentence G of Ω:

$$\exists X^1 \exists x_1^0, \ldots, x_n^0 \Big(\mathbf{1} \in X^1 \land \mathbf{0} \notin X^1 \land F'\Big),$$

where F' is obtained from F by replacing all occurrences of x_i with $x_i^0 \in X^1$. The intuition here is that X^1 is forced to be a singleton containing the value $\mathbf{1}$ (true), and the truth of a propositional variable x_i is modeled by a membership $x_i^0 \in X^1$ in the truth set X^1. The sentence G is constructible from F in linear time and is true in Ω if and only if F is satisfiable. The membership in *NP* is straightforward: just guess a vector of $\mathbf{0}/\mathbf{1}$-values for $x_1, \ldots, x_n$ and check whether they satisfy F'. □

Proposition 16 . The class of quantified prenex sentences of Ω with quantifier prefix containing just one existentially quantified variable of type 1 and all other quantified variables of type 0 is *PSPACE*-complete.

Proof. Given a prenex quantified boolean formula $\Phi \equiv Q_1 x_1 \ldots Q_n x_n F$ with matrix in 3-CNF and $Q_i \in \{\exists, \forall\}$, consider the following sentence Ψ of Ω

$$\exists X^1 Q x_1^0 \ldots Q_n x_n^0 \Big(\mathbf{1} \in X^1 \wedge \mathbf{0} \notin X^1 \wedge F'\Big),$$

where F' is obtained from F by replacing all occurrences of x_i with $x_i^0 \in X^1$. The sentence Ψ is constructible from Φ in linear time and is true in Ω if and only if Φ is true. The membership in *PSPACE* is also straightforward, because linear space suffices to cycle through all n-tuples of $\mathbf{0}/\mathbf{1}$-values for $x_1, \ldots, x_n$, which is enough to check the validity of Ψ. □

11. Arbitrarily Hard Variants of Ω

Until now we were agreed to write down types of variables in formulas of Ω in the usual binary notation.

Now let f be an arbitrary function from $\{0,1\}^*$ to ω, $0 \in \{0,1\}^*$ denote the natural number 0, and s denote the usual successor function on ω. By using 0, s, and f we can denote any natural number in many different ways, in general. If, for example, we do not use f at all, we get the usual unary notation. If we do not use s, and f is defined by $f(0) = f() = 0$, $f(0\alpha) = f(\alpha)$, and $f(1\alpha) = 2^{|\alpha|} + f(\alpha)$, where $|\alpha|$ is the length of α, then we get the usual binary notation for natural numbers.

Now, if f is interpreted as a very rapidly growing function, for example, as $\lambda x.A(x,x,x)$ for Ackermann's function A, then we get a very compact notation for very large natural numbers. We can use this compact notation to write down types of variables in formulas of Ω, instead of the binary notation we accepted in Section 2. Denote the resulting theory Ω_f. By applying the same lower bounds techniques as in Sections 3 — 6, we get the following arbitrarily hard variants of Ω:

Theorem 17 (Lower Bound for Ω_f) . *Any Turing machine deciding Ω_f requires a number of steps exceeding*

$$\exp_\infty(f(cn))$$

for some constant $c > 0$ and infinitely many sentences S of Ω_f of length n.

Proof. Use exactly the same argument to represent a binary relation of size up to $\exp_\infty(f(n+2)+1)$ (instead of $\exp_\infty(2^{n+2}+1)$), as in Section 6, but use variable types written succinctly with f and s. □

One may argue, however, that with Theorem 17 we leave the world of "natural" decidable theories.

12. Simply Typed λ-Calculus: Improved Lower Bound Hits the Upper

Only a very superfluous knowledge of the simply typed lambda-calculus Λ is needed to understand this and the following sections. We use standard definitions and notation from [8, 1, 15, 17] and make a standard convention to omit types in writing lambda terms, assuming that they can be unambiguously restored. This does not influence complexity, see Remark 22 below.

In Section 12.1 we show that the best known before lower bound for β- and $\beta\eta$-equality in the simply typed lambda calculus, implicit in [15, 11], is *not very impressive*, although it implies that the problem is not elementary recursive:

Theorem 18 (Statman, Mairson) . *Any Turing machine deciding β- or $\beta\eta$-equality in the simply typed lambda calculus requires a number of steps exceeding*

$$F(n, c\log(\log(n))) = 2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}} \left.\vphantom{2^{2^{2^2}}}\right\} c\log(\log(n))$$

for some constant $c > 0$ and infinitely many inputs of length n. □

Note, however, that this is stronger than just saying that β- or $\beta\eta$-equality in the simply typed lambda calculus are not elementary recursive [15].

In Section 12.2, by using our Theorem 7 we considerably improve Theorem 18 to the following

Theorem 19 (Improved Lower Bound for Λ) . *Any Turing machine deciding β- or $\beta\eta$-equality in the simply typed lambda calculus requires a number of steps exceeding*

$$\exp_\infty(cn) = 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \left.\vphantom{2^{2^{2^2}}}\right\} cn$$

for some constant $c > 0$ and infinitely many inputs of length n. □

Remark 20 . This finally closes the deep gap between the known $\exp_\infty(dn)$ upper bound for deciding $\beta(\eta)$-equality in Λ (due to W. W. Tait, cf., [6], Theorem 4.4.2) and the weak lower bound from Theorem 18 above. Now both lower and upper bounds match exactly (with different constants). □

The following easy technical proposition is very useful in establishing lower bounds in terms of time requirements by reduction between problems, and will be used several times in the sequel. Notice that $L \notin NTIME(t(n))$ is equivalent to saying that any decision procedure for L requires nondeterministic time exceeding $t(n)$ for infinitely many inputs of length n.

Proposition 21 . Let a problem L_1 reduce to a problem L_2 by means of a transformation f computable in time $p(n)$ such that for some monotone function $g : \omega \to \omega$ one has $|f(x)| \leq g(|x|)$ for all x in the language of the problem L_1. Let $t(n)$ be a time bound. Then:

- if $L_2 \in NTIME(t(n))$ then $L_1 \in NTIME(t(g(n)) + p(n))$;
- vice versa, if $L_1 \notin NTIME(t(g(n)) + p(n))$ then $L_2 \notin NTIME(t(n))$.

Proof. To decide whether x of size n is in L_1 compute in time $p(n)$ the value $f(n)$ of size at most $g(|n|)$ and test if it belongs to L_2. So, the decision problem for L_1 is in $NTIME(t(g(n)) + p(n))$, provided that $L_2 \in NTIME(t(n))$. □

12.1. Statman-Mairson's Lower Bound

R. Statman [15] and later H. Mairson [11] gave *exponential* reductions * from the sentences of Ω into the simply typed lambda terms with the properties:

$$\Omega \models A \quad \Leftrightarrow \quad A^* =_\beta \mathbf{0},$$
$$\Omega \models \neg A \quad \Leftrightarrow \quad A^* =_\beta \mathbf{1},$$

where $\mathbf{0}$ and $\mathbf{1}$ are the usual Church numerals for zero and one.

It follows immediately that the problem of deciding whether two simply typed lambda terms are $\beta(\eta)$-equal is not elementary recursive [15, 11]. Note that the same would follow from the existence of *any* reduction of the sentences of Ω into the simply typed lambda terms computable in elementary space.

As we explained in the Introduction, once a theory is proved non-elementary, it is important to know, if possible, how large is k in the k-fold logarithm determining the speed of the exponentiation stack growth in its lower bound $F(n, c\log^k(n))$.

How large is this k for the β- and $\beta\eta$-equality in the simply typed lambda calculus? Neither Statman [15], nor Mairson [11] answers this question explicitly. Statman does not discuss it at all. Mairson makes some comments (pp. 389–391) concluding that the reduction from Ω to Λ is *linear.* However, it is *exponential* assuming that the variable types are written in binary (cf., Convention 2; recall that [12, 15, 11] *never specify* whether types are written in unary or in binary). This is because Mairson represents the quantification $\forall x^k$ by means of a k-iterated application of the *powerset* combinator (p. 389–391). This is especially clear from Statman's reduction ([15], p. 75–76), which represents the quantification $\forall x^k$ by using a_k, where a_1 is the Church numeral $\mathbf{2} \equiv \lambda f^{0\to 0}x^0.f(fx)$ and $a_{n+1} \equiv ([0 \to 0]a_n)a_1$.

Thus, the reduction from Ω with the lower bound from Fischer-Meyer's Theorem 6 by using the well-known techniques from Proposition 21 yields only a very poor lower bound

$$F(n, c\log(\log(n)))$$

of Theorem 18 for deciding β- and $\beta\eta$-equalities in the simply typed lambda calculus. This is the maximum that one can extract from Statman's and Mairson's proofs that rely on Fischer-Meyer's lower bound from Theorem 6.

Remark 22 . The only source of exponential explosion in Statman's translation * is the representation of the quantification $\forall x^k$ with type k written in binary by means of a_k's. This gives an exponential increase, because the type k represented *in binary* is rewritten in *unary* as a k-fold application of $\mathbf{2}$'s.

The retyping $a_{n+1} \equiv ([0 \to 0]a_n)a_1$ does not give any additional increase of length, even if the bounded variables are typed explicitly, despite the fact that the types in a_k are of length $O(2^k)$. Indeed, the length of a_k with explicit types is $\leq k \cdot O(2^k)$, i.e., $\leq O(2^k)$, the same as the length of a_k with implicit typing. □

12.2. Improved $\exp_\infty(cn)$ Lower Bound for the Simply Typed Lambda Calculus

Our improved $\exp_\infty(cn)$ lower bound for lambda calculus from Theorem 19 follows from our Main Theorem 7 and Proposition 21 by application of Statman's

[15] (p. 75–76) translation * of Ω into Λ, or Mairson's one [11] (pp. 389–391). Both reductions are *exponential*, see the previous section. This gives a less impressive $\exp_\infty(cn)$ lower bound, as compared to $\exp_\infty(2^{cn})$ lower bound for Ω. However, even this is much stronger than Statman-Mairson's lower bound from Theorem 18. Note that we cannot do better because of the $\exp_\infty(dn)$ upper bound on the $\beta(\eta)$-equality due to W. Tait ([6], Theorem 4.4.2).

Remark 23 . It is easy to see that the same lower bound $\exp_\infty(dn)$, as in Theorem 19, holds for the theory of $=_{\beta\eta}$ of $\beta\eta$-equality. In fact, $A^* =_{\beta\eta} \mathbf{0}$ implies $A^* =_\beta \mathbf{0}$ (and similarly for $\mathbf{1}$), because otherwise $A^* =_\beta \mathbf{1}$ (recall that Ω is complete); hence $A^* =_{\beta\eta} \mathbf{1}$ (since $=_{\beta\eta}$ extends $=_\beta$), and we get a contradiction with the Church-Rosser property known to hold for $\beta\eta$-equality in Λ. □

Remark 24 . We could equally use the lower bound $\exp_\infty(\varepsilon \cdot n)$ for Ω_{unary} from Theorem 13. In this case the reduction * from Ω_{unary} to Λ would be *linear*, and we would get the same lower bound $\exp_\infty(dn)$ for Λ, as in Theorem 19. □

13. A Variant of the Simply Typed Lambda Calculus with $\exp_\infty(2^{cn})$ Lower Bound

Thus both lower and upper bounds for the standard simply typed lambda calculus Λ are $\exp_\infty(cn)$ and $\exp_\infty(dn)$ respectively for some constants $d > c > 0$. As we explained in Section 12.1, the main reason why the lower bound for Λ is less impressive than the $\exp_\infty(2^{cn})$ lower bound for Ω is the *exponential* translation * of Ω into Λ.

In fact, Statman represents ([15], p. 75–76), the quantification $\forall x^k$ by using a_k, where a_1 is the usual Church numeral $\mathbf{2} \equiv \lambda f^{0\to 0}x^0.f(fx)$ and $a_{n+1} \equiv ([0 \to 0]a_n)a_1$. Consequently, $\forall x^k$ with the type k represented *in binary* is translated into the exponentially long sequence of applications of $\mathbf{2}$'s: $\underbrace{\mathbf{2}\ldots\mathbf{2}}_{k \text{ times}}$.

We can extend the standard simply typed lambda calculus Λ by the possibility to write iterators concisely, e.g., as $It_k(a)$, where k is written *in binary*, and a has appropriate type. Add also the extra reductions: $It_1(a)$ reduces to a, and $It_{n+1}(a)$ reduces to $([0 \to 0](It_n(a)))a$. With these modifications the reduction * from Ω to the extended Λ becomes *linear*, and we get a variant (conservative extension) of Λ with the $\exp_\infty(2^{cn})$ lower bound on equality and normalization.

14. Schwichtenberg's Lower Bound for Normalization

In [14] H. Schwichtenberg gave a very short and simple proof that any normalization algorithm for simply typed lambda calculus requires non-elementary recursive time. Although this result does not imply Statman's on non-elementary recursiveness of $\beta(\eta)$-equality in simply typed λ-calculus [15] (there may exist tricky algorithms deciding $\beta(\eta)$-equality without reducing terms to normal forms), it gives a better lower bound for normalization. In fact, Statman's proof implies that any normal form algorithm should be not elementary recursive, but as we explained in Section 12, only with a quite weak $F(n, c\log(\log(n)))$ lower bound.

Schwichtenberg showed in [14] that any normalization of the term

$$r_n \equiv \underbrace{\mathbf{2}\mathbf{2}\ldots\mathbf{2}}_{n},$$

where $\mathbf{2} \equiv \lambda f \lambda x.f(f(x))$, which normalizes to $F(1,n)$ (written as a Church numeral), takes more than $F(n, cn)$ reduction steps, thus yielding a stronger lower bound on any normalization algorithm[3]. The same lower bound easily follows from our improved $F(1, cn)$ lower bound for $\beta(\eta)$-equality in simply typed λ-calculus, cf., Section 12. In fact, if one could normalize faster than in time $F(1, cn)$, it would be possible to decide $\beta(\eta)$-equality faster than in $F(1, cn)$, in contradiction with our Theorem 19.

15. The $\exp_\infty(cn/log(n))$ Lower Bound for Higher-Order Matching

G. Huet [9] posed the following, ***today still open***, decidability problem for the simply typed lambda calculus, called the *higher-order matching problem*:

Given a term t of type $\sigma_1 \to \ldots \to \sigma_n \to \tau$ and a term u of type τ (both in normal forms), do there exist terms s_i of types σ_i (for $1 \le i \le n$) such that

$$ts_1 \ldots s_n =_{\beta\eta} u \ ?$$

This problem is referred to as the *range question* by R. Statman [17].

Huet's conjecture is that the problem is decidable, but the proof is probably hard. G. Dowek showed [3]

[3] It is easy to see that the length of the normal form of r_n is of length $O(F(1,n))$, so just spelling it out requires time $O(F(1,n))$; Schwichtenberg's proof is stronger: it also shows that r_n requires that many reduction steps to normalize.

that the third-order matching[4] is decidable. Statman reduced [17] the higher-order matching to the so-called definability problem: *given an element of a finitely generated type hierarchy, does there exist a closed simply typed λ-term denoting this element?* The positive answer was long known as Plotkin-Statman's conjecture. Recently Ralph Loader settled it in the negative [10], thus leaving the higher-order problem *open.*

We can contribute to the settlement of the higher-order matching problem the following strong lower bound:

Theorem 25 ***(Lower Bound for Higher-Order Matching).*** *Any algorithm for the higher-order matching in the simply typed lambda calculus should necessarily make a number of steps exceeding*

$$\exp_\infty(cn/\log(n)) = 2^{2^{\cdot^{\cdot^{2}}}} \Big\} \; cn/\log(n)$$

for some constant $c > 0$ and infinitely many inputs of size n. □

Remark 26 . It follows immediately that the problem is not elementary recursive, see also Remark 27 below. One can ask him/herself whether there is any difference, from the practical viewpoint, between an undecidable problem and a problem requiring as much as $\exp_\infty(cn/\log(n))$ time to decide. □

We prove Theorem 25 by reduction from Ω and applying Theorem 7 as follows. Consider a sentence A of Ω and its Statman's translation A^*, as described in the previous section.

Starting with the initial equation $A^* = \mathbf{0}$ we stepwise "eliminate" redices in A^* by introducing new variables and adding new equations to the initial one (recall that the terms in the higher-order matching should be normalized) as follows:

- replace a β-redex $(\lambda xM)N$ in A^* with $(f(\lambda xM))N$, where f is a fresh variable of appropriate type;
- replace an η-redex λxMx (x not free in M) in A^* with $\lambda xf(Mx)$, where f is a fresh variable of appropriate type;
- in both cases add an equation $fC = C$, where C is a new constant of appropriate type, to the initial equation.

The result of this transformation is a system of simultaneous equations with new functional variables and new constants, with all terms being in normal forms. It is not difficult to prove that the resulting system has a solution iff $A^* =_{\beta\eta} \mathbf{0}$. However, the transformation increases the input as $n\log(n)$, since one has to expect a linear number of redices and consequently linearly many new variables and constants, each one needs $log(n)$ bits to be represented. The system of equations is then reduced to *just one equation* by tupling: $\bigwedge_{i=1}^{n} M_i = N_i \Leftrightarrow \lambda f.fM_1\ldots M_k = \lambda f.fN_1\ldots N_k$. Note that the resulting equation *does not contain variables on the right,* so we translated the initial matching problem to the matching problem. Finally, an equation *with constants* may be reduced to an equation *without constants,* in the pure simply typed lambda calculus, by application of Statman's trick [16] (p. 330–331). The last two transformations are *linearly bounded.* They do not introduce new variables into right-hand sides of equations, nor do they change the $\beta\eta$-normality of terms. So, the result is a matching problem with normalized terms.

Since the total transformation from Ω to the pure higher-order matching gives growth of order $O(n\log(n))$, by Proposition 21, we get for the higher-order matching problem the nondeterministic time lower bound of Theorem 25.

Thus, even if the higher-order problem is decidable, any corresponding decision procedure should be *desperately hard.*

Remark 27 . As we explained in Section 12.1, the best lower bound for the higher-order matching problem one could have extracted from Statman's proof [15] based on Fischer-Meyer's lower bound from Theorem 6, would have been much weaker, namely, "*only*"

$$F(n, c\log(\log(n/\log(n)))).$$

Although even this weak lower bound implies that the higher-order matching problem is not elementary recursive, it is not as convincing as our strong lower bound from Theorem 25. □

16. Improved Lower Bound for Statman's Analog of Rice's Theorem

As one further application of the strong lower bound for Ω from Theorem 7, we improve the lower bound in Statman's analog of Rice's theorem.

[4] when the orders of types σ_i in the statement of the problem are at most three; $ord(\alpha) = 1$ for a basic type and $ord(\sigma \to \tau) = \max\{1 + ord(\sigma), ord(\tau)\}$; the type of Church numerals $(0 \to 0) \to (0 \to 0)$ is of order 3; in Dowek's formulation, where terms are not required to be normalized, the higher-order matching is non-elementary for *all* orders.

R. Statman proved ([17], p. 25, Theorem 6) that in the simply typed lambda calculus any nontrivial subset of closed terms closed under $\beta\eta$-conversions is not elementary recursive. This corresponds, modulo the replacement of "not recursive" by "not elementary recursive", to Rice's theorem in recursion theory. Recall that the theorem asserts that any nontrivial subset of partial recursive functions is not recursive. In the context of untyped lambda calculus the analog of Rice's theorem is due to D. Scott: any nontrivial set of untyped lambda terms closed under equality is not recursive.

We can improve the lower bound in Statman's theorem as follows (we state it in a stronger separability form):

Theorem 28 . *Let S_1, S_2 be two nonempty disjoint sets of closed simply typed lambda terms of the same type. Any set S closed under $\beta\eta$-conversion and separating S_1 and S_2 (i.e., $S_1 \subseteq S$ and $S_2 \cap S = \emptyset$) requires time exceeding*

$$F(1,cn) = \exp_\infty(cn) = 2^{2^{\cdot^{\cdot^{2}}}}\Big\}cn,$$

for some constant $c > 0$ and infinitely many inputs of length n to be recognized.

In particular, any nonempty set S of closed terms of type σ, different from the set of all terms of type σ, and closed under $\beta\eta$-conversion requires that much time to be recognized.

Proof. By a slight modification of Statman's proof. If a set S closed under $\beta\eta$-conversion and separating nonempty sets S_1, S_2 of terms of type σ exists, then there exist two terms $M \in S$ and $N \notin S$. Let M, N have long $\beta\eta$-normal forms $\lambda x_1 \ldots x_k.x_i(M_1 \ldots M_m)$ and $\lambda x_1 \ldots x_k.x_j(N_1 \ldots N_n)$ respectively, where $x_i(M_1 \ldots M_m)$ and $x_j(N_1 \ldots N_n)$ are of the base type 0. Consider the term L of type $((0 \to 0) \to (0 \to 0)) \to \sigma$ defined by $L \equiv_{df} \lambda y \lambda x_1 \ldots x_k.y(\lambda z^0.x_j(N_1 \ldots N_n))(x_i(M_1 \ldots M_m))$, where y is of type $(0 \to 0) \to (0 \to 0)$. Note that $L\mathbf{0} =_{\beta\eta} M$ and $L\mathbf{1} =_{\beta\eta} N$. Now let * be the translation of sentences of Ω into simply typed lambda terms used in Section 12.1. We thus have for an arbitrary sentence A of Ω:

$$\Omega \models A \Leftrightarrow L\,A^* \in S.$$

It remains to apply Proposition 21. The last claim of the theorem is a particular case of the first one. □

As we explain above in Section 12.1, the best lower bound in Theorem 28 one could have obtained without our Theorem 7 would be "only" $F(n, c\log(\log(n)))$.

17. Conclusions

We settled the strongest currently known lower bound of the form

$$\exp_\infty(2^{cn}) = 2^{2^{\cdot^{\cdot^{2}}}}\Big\}\text{ height } 2^{cn}$$

for a natural decidable theory of pure propositional types, a rudimentary fragment of L. Henkin's theory of propositional types [7]. Until now it was an *open* problem as to whether natural decidable theories requiring more than

$$\exp_\infty(cn) = 2^{2^{\cdot^{\cdot^{2}}}}\Big\}\text{ height } cn$$

for any constant $c > 0$ exist [12, 2].

We used this unprecedentedly high lower bound to considerably improve the known

$$F(n, c\log(\log(n))) = 2^{2^{\cdot^{\cdot^{2^n}}}}\Big\}\text{ height } c\log(\log(n))$$

lower bound for the $\beta(\eta)$-equality in the simply typed lambda calculus [15] to a much more impressive

$$\exp_\infty(cn) = 2^{2^{\cdot^{\cdot^{2}}}}\Big\}\text{ height } cn$$

matching the known $\exp_\infty(dn)$ upper bound due to W. Tait [6].

We also used our new strong lower lower bound to settle the new

$$\exp_\infty(cn/\log(n)) = 2^{2^{\cdot^{\cdot^{2}}}}\Big\}\text{ height } cn/\log(n)$$

lower bound for a long-standing, today still open, higher-order matching problem for the simply-typed lambda calculus due to G. Huet [9, 21].

It is now "natural" to ask whether there exist natural decidable theories that require time exceeding

$$\exp_\infty(f(n)),$$

where $f(n)$ is not *exponentially bounded*, i.e.,

$$f(n) \leq c \cdot 2^n$$

does not hold for any constant $c > 0$.

For example, do there exist natural theories requiring time

$$\exp_\infty(\exp_\infty(cn))$$

to decide?

Whether these theories can be used to improve known lower bounds for decidable theories (as we did it for Λ), or to settle new lower bounds for theories which are not known to be decidable (as we did it for the higher-order matching)? Only the positive answer to this question can serve a criterion as to whether a theory is natural or not.

A. Mairson's Proof

The sketch of the proof in [11] Section 4 that the theory Ω is not elementary recursive appears to be not completely correct for two reasons.

First, the promised "simulation of an arbitrary Turing machine for non-elementary time" (pp. 388 and 391) does not work, e.g., for a machine running in non-elementary time $\exp_\infty(\exp_\infty(n)))$. The correct way would be to show that for every fixed $m \in \omega$ any Turing machine running in time $F(n, m)$ can be represented by a formula of Ω of length $O(n)$ or $O(poly(n))$. Then the standard diagonal argument using hierarchy theorems shows that Ω cannot belong to $NTIME(F(n, m))$ for any $m \in \omega$, i.e., is non-elementary.

Second, the reference, concerning the representation of a Turing machine, as being "straightforward, more or less on the level of the detailed coding in Cook's theorem" (p. 392) is not completely correct. In proving Cook's theorem one uses polynomially many variables to represent cells of a polynomially long tape. Analogous usage of $O(F(n, m))$ variables to represent an $F(n, m)$ run cannot lead to any useful conclusion. All considerations about the lengths of formulas arising from the encoding are absent from [11]. This does not allow even to settle a weak lower bound of Fischer-Meyer's Theorem 6.

The proof from [11] can be fixed, but it turns out to be unnecessary in view of our stronger Theorem 7.

Acknowledgments. The author thanks Harald Ganzinger, David Basin, Ralph Loader, David Wolfram, and anonymous referees for fruitful discussions and useful comments.

References

[1] H. Barendregt. *The Lambda Calculus. Its Syntax and Semantics.* North-Holland, 1984.

[2] K. J. Compton and C. W. Henson. A uniform method for proving lower bounds on the computational complexity of logical theories. *Annals Pure Appl. Logic*, 48:1–79, 1990.

[3] G. Dowek. Third order matching is decidable. *Annals Pure Appl. Logic*, 69:135–155, 1994.

[4] Y. L. Ershov, I. A. Lavrov, A. D. Taimanov, and M. A. Taitslin. Elementary theories. *Russian Math. Surveys*, 20:35–105, 1965.

[5] M. J. Fischer and M. O. Rabin. Super-exponential complexity of Presburger arithmetic. In *SIAM-AMS Proceedings*, volume 7, pages 27–41, 1974.

[6] S. Fortune, D. Leivant, and M. O'Donnell. The expressiveness of simple and second-order type structures. *J. ACM*, 30(1):151–185, 1983.

[7] L. Henkin. A theory of propositional types. *Fund. Mathematicæ*, 52:323–344, 1963.

[8] J. Hindley and J. Seldin. *Introduction to Combinators and Lambda Calculus.* Cambridge Univ. Press, 1986.

[9] G. Huet. *Résolution d'Équations dans les Langages d'Ordre 1, 2, ..., ω.* Thèse de Doctorat d'État, Université de Paris VII, 1976.

[10] R. Loader. The undecidability of λ-definability. "Types" electronic forum, 1993.

[11] H. Mairson. A simple proof of a theorem of Statman. *Theor. Comput. Sci.*, 103:387–394, 1992.

[12] A. R. Meyer. The inherent computational complexity of theories of ordered sets. In *International Congress of Mathematicians*, pages 477–482, Vancouver, 1974.

[13] M. O. Rabin. A simple method for undecidability proofs and some applications. In Y. Bar-Hillel, editor, *Proc. Intern Congress on Logic, Methodology and Philosophy of Science*, Studies in Logic and the Foundations of Mathematics, pages 58–68. North-Holland, 1964.

[14] H. Schwichtenberg. Complexity of normalization in the pure typed λ-calculus. In A. S. Troelstra and D. van Dalen, editors, *L. E. J. Brouwer Centenary Symposium*, pages 453–458. North-Holland, 1982.

[15] R. Statman. The typed λ-calculus is not elementary recursive. *Theor. Comput. Sci.*, 9:73–81, 1979.

[16] R. Statman. On the existence of closed terms in the typed λ-calculus II: transformations of unification problems. *Theor. Comput. Sci.*, 15:329–338, 1981.

[17] R. Statman. Completeness, invariance, and λ-definability. *J. Symb. Logic*, 47(1):17–26, 1982.

[18] W. W. Tait. Infinitely long terms of transfinite type. In J. N. Crossley and M. Dummett, editors, *Formal systems and recursive functions*, pages 176–185. North-Holland, 1967.

[19] B. A. Trakhtenbrot. On the impossibility of an algorithm for the decision problem in finite domains. *Soviet Math. Doklady*, 70:569–572, 1950. In Russian.

[20] R. L. Vaught. Sentences true in all constructive models. *J. Symb. Logic*, 25(1):39–53, 1960.

[21] D. A. Wolfram. *The Clausal Theory of Types.* Cambridge Univ. Press, 1993.