$\mathcal{P}$ is not equal to $\mathcal{NP}$*

Sten-Åke Tärnlund†

November 20, 2021

Abstract

Turing machine computations are Robinson resolution proofs from the single Axiom 1, a universal Turing machine. Then, in Hilbert's proof theory: SAT is not in $\mathcal{P}$, thus: $\mathcal{P}$ is not equal to $\mathcal{NP}$.

1 Introduction

Theorem 2 $\mathcal{P} \neq \mathcal{NP}$ follows from Theorem 1 $SAT \notin \mathcal{P}$ that has a proof from Haken's theorem[6] [7] and Lemma 1,[1] which is proved from Corollaries 1–3, and some calculus. The proofs are in Hilbert's proof theory [11].

In Turing's first-order theory [18], all Turing machine computations are proofs in a Hilbert-type logic system, see Hilbert and Ackermann [10]. But, all Turing machine computations are Robinson resolution proofs [15] from Axiom 1, a universal Turing machine, see Corollary 1.[2]

2 A single axiom of computing

A universal Turing machine written informally:[2] (i) [Boot up] If U (symbols $\emptyset$, $\emptyset$, state 1, and the tape-head at $\emptyset$ initialize a two-way tape) for index i (a list of quintuples) and argument a (written $d(z,\, z)$) computes u then Turing machine i computes u for a (written $d(z,\, i)$).[3] (ii) [Termination] At state 0, U computes u and halts. (iii) [A state-symbol cycle] (iii) If U for the tape x, y, p, $d(r.\, z, r.\, z)$, u (the tape-head moves left to y) computes u then U for the tape $x\,.\,y$, s, q, $d(z, q\,.\,s\,.\,p\,.\,r\,.\,0\,.\,j)$, u (quintuple $q\,.\,s\,.\,p\,.\,r\,.\,0$ is found) computes u. (iv) [A state-symbol cycle] Similarly to (iii), but the tape-head moves right to y. (v) [Quintuple search] If U for the tape x, y, q, $d(z,\, j)$, u computes u then U for the tape x, y, q, $d(z,\, v\,.\,j)$, u computes u.

The domains are: symbols $r\,s\,y\,v \in S$, states $p\,q \in Q$, right-tapes $a\,z \in R$, left-tapes $u\,x \in L$, and indices $i\,j \in I$. Then, $U \subseteq L \times S \times Q \times R \times L$, and $T \subseteq I \times R \times L$ are relations. The terms, left and right tapes, are maps $h : list \times element \to list$, $l : element \times list \to list$, $d\text{-}list : list \times list \to list$.[3]

In first-order logic (binary Horn clauses) (i)–(v) are more precisely:

---

* Edition 12, minor editing of edition 11 Tärnlund [17].

[1] For the notation $\mathcal{P}$, $\mathcal{NP}$ and $\mathcal{P}$ vs $\mathcal{NP}$ see Cook [4] and Karp [12].

[2] The universal Turing machine (4a)–(4e) is equivalent to Axiom 1. It runs as a Prolog program, just as Axiom 1 would, see Tärnlund [16], Colmerauer [3], and Warren [19].

[3] Maps of d-lists $(z,\, i) \mapsto a$, and concatenation $(a,\, i) \mapsto z$, see Clark and Tärnlund [2].

Axiom 1

$$U(\emptyset, \emptyset, 1, d(z, z), u) \supset T(i, d(z, i), u). \tag{1}$$
$$U(u, y, 0, z, u). \tag{2}$$
$$U(x, y, p, d(l(r\,, z), l(r\,, z)), u) \supset \tag{3}$$
$$U(h(x, y), s, q, d(z, l(q, l(s, l(p, l(r, l(0, j)))))), u).$$
$$U(h(x, r), y, p, d(z, z), u) \supset \tag{4}$$
$$U(x, s, q, d(l(y, z), l(q, l(s, l(p, l(r, l(1, j)))))), u).$$
$$U(x, y, q, d(z, j), u) \supset U(x, y, q, d(z, l(v, j)), u). \tag{5}$$

All variables in a sentence are universally quantified implicitly, in front of the entire sentence. Only $\emptyset$, 0 *and* 1 *are constants.*

Writing B for Axiom 1, i.e., a universal Turing machine. (6)

Writing Turing machine i for B_i i.e., i is a name of the index that is a list of quintuples assigned to B. B_i is deterministic if and only if there is a map onto i $(q, s) \mapsto (q, s, p, r, d)$ all $i \in I$. (7)

Lists and d-lists are interchangeable for coding the T predicate to run B.[4]

$$a = d(z, i) \supset T(i, a, u) \equiv T(i, d(z, i), u) \text{ all } a\, z \in R\ i \in I\ u \in L. \tag{8}$$

3 Complexity of Computing

Definition 1 *Let* $T(i, a, u)$ *in* t *be* $T(i, a, u)$ *has a formal proof from* B_i *using (3)–(4) at most* t *times, by (8), i.e., Turing machine* i *computes* u *for* a *in computing time* t *all* $i \in I$ $a \in R$ $u \in L$ $t \in Z^+$, *cf. Hartmanis and Stearns [9].*

If $T(i, a, u)$ in t then there is a proof of $T(i, a, u)$ from B in t all $i \in I$ $a \in R$ $u \in L$ $t \in Z^+$, by (8) and SLD-resolution, see Kowalski [14] and completeness see Clark [1]. The index is an algorithm for proofs.[2] More formally,

Corollary 1 *If* $T(i, a, u)$ *in* t *then* $\vdash_R B \rightarrow T(i, a, u)$ *in* t *all* $i \in I$ $a \in R$ $u \in L$ $t \in Z^+$, *by (8) and SLD-resolution using G*4 *notation, cf. Kleene [13].*

Proof.

Assume $T(i, a, u)$ in t $i \in I$ $a \in R$ $u \in L$ $t \in Z^+$. Thus, (9)

$\vdash_R B_i, \neg T(i, a, u) \rightarrow$ in t, (7), (8), Definition 1, and SLD-resolution. (10)

So, if $T(i, a, u)$ in t then $\vdash_R B \rightarrow T(i, a, u)$ in t all $i \in I$ $a \in R$ $u \in L$ $t \in Z^+$, (6) and (7). $\square$ (11)

Let D be the set of all indices of deterministic Turing machines (7) computing whether $\not\models \neg G$, i.e., satisfiability, $D \subset I$ all $G \in PROP$ (the set of all formulas in propositional logic.) (12)

A satisfiability relation for any deterministic Turing machine more formally,

[4] Program transformation, see Hansson and Tärnlund [8].

Definition 2 $T(i, G, 0) \equiv \not\models \neg G$ *and* $T(i, G, 1) \equiv \models \neg G$ *all* $i \in D$ $G \in PROP$.

If satisfiability is in P then there is a deterministic Turing machine computing whether G is satisfiable, in polynomial computing time in the size of G (number of symbols $|G|$ of G) all $G \in PROP$, see Cook [4]. More formally,

Definition 3 *If* $SAT \in \mathcal{P}$ *then* $\exists u\, T(i, G, u)$ *in* $c \cdot |G|^n$ *some* $i \in D$ $c\ n \in Z^+$ *all* $G \in PROP$.

Writing d for the name of some index $i \in D$ satisfying Definition 3. (13)

If $SAT \in \mathcal{P}$ then Turing machine d proves $\neg F$ unsatisfiable polynomially all $F \in TAUT$, by SLD resolution. More precisely,

Corollary 2 *If* $SAT \in \mathcal{P}$ *then* $\vdash_{R} B \rightarrow T(d, \neg F, 1)$ *in* $c \cdot |F|^n$ *some* $n \in Z^+$ *all* $F \in TAUT$ *(the set of all tautological propositions) – using SLD-resolution.*

Proof.

Suppose $SAT \in \mathcal{P}$. Then, Definitions 2–3, and (13), (14)

$T(d, \neg F, 1)$ in $c \cdot |F|^n$ $c\ n \in Z^+$ $F \in TAUT$. Hence, (15)

if $SAT \in \mathcal{P}$ then $\vdash_{R} B \rightarrow T(d, \neg F, 1)$ in $c \cdot |F|^n$ (16)

some $c\ n \in Z^+$ all $F \in TAUT$, Corollary 1. $\square$

If $SAT \in \mathcal{P}$ then Turing machine d does not prove G unsatisfiable polynomially all satisfiable $G \in PROP$, using SLD-resolution. More precisely,

Corollary 3 *If* $SAT \in \mathcal{P}$ *then* $\neg T(d, G, 1)$ *in* $c \cdot |G|^n$ *some* $c\ n \in Z^+$ *all satisfiable* $G \in PROP$, *by SLD-resolution.*

Proof.

Suppose $SAT \in \mathcal{P}$. So, $\not\vdash_{R} B, \neg T(d, G, 1) \rightarrow$ in $c \cdot |G|^n$ $c\ n \in Z^+$ (17)

$\not\models \neg G$ $G \in PROP$, Definitions 2–3, (13), Corollary 1. (18)

Assume $T(d, G, 1)$ in $c \cdot |G|^n$. Then, $\vdash_{R} B, \neg T(d, G, 1) \rightarrow$ in $c \cdot |G|^n$, (19)

Corollary 1. Thus, if $SAT \in \mathcal{P}$ then $\neg T(d, G, 1)$ in $c \cdot |G|^n$ some (20)

$c\ n \in Z^+$ all satisfiable $G \in PROP$, contradiction (17)–(19). $\square$

The size (number of symbols) of an existing proof (computation) is another complexity measure, cf. Gödel [6].

Definition 4 *Let* $| \vdash_{R_p} F | \in O(|F|^n)$ *be there exists a propositional Robinson resolution proof of* F *of polynomial size* $O(|F|^n)$ *all* $n \in Z^+$ $F \in TAUT$ *in DNF (disjunctive normal form).*

4 Lemma 1

If $SAT \in \mathcal{P}$ then any sufficiently large tautology F in DNF has a propositional Robinson resolution proof of polynomial size – in the size of F. More formally,

Lemma 1 *If* $SAT \in \mathcal{P}$ *then* $| \vdash_{R_p} F | \in O(|F|^n)$ *some* $n \in Z^+$ *all sufficiently large* $F \in TAUT$ *in DNF.*

Proof.

$$\text{Suppose } SAT \in \mathcal{P}. \tag{21}$$

$$\text{Further assume } \neg F, \text{ then there is} \tag{22}$$

an SLD-proof of F polynomially, by (8), and Corollaries 2–3,

$$\vdash_R B, \neg F, \neg T(d, \neg F, 1) \to \quad \text{in } c \cdot |F|^n \; c \; n \in Z^+ \; F \in TAUT \text{ in } DNF. \tag{23}$$

Writing $U_{j\,k}$ for the U predicates of Axiom 1, j enumerates the computing time, using (3)–(4), k enumerates the search for a quintuple of d, using (5), $j\,k \in N$. Moreover, writing T for $T(d, \neg F, 1)$. Using these short names, there is an SLD-resolution proof tree, $j+1, k \leq c \cdot |F|^n$, by (23),

$$B, \neg F, U_{j\,k_j} \overset{\times}{\to} U_{j\,k_j} \qquad B, \neg F, U_{(j+1)\,0} \overset{\times}{\to} U_{(j+1)\,0} \tag{24}$$

$$\vdots$$

$$B, \neg F, \; U_{0\,k} \overset{\times}{\to} U_{0\,k} \qquad B, \neg F, U_{1\,1} \supset U_{1\,0} \to U_{1\,0} \tag{25}$$

$$\backslash \qquad\qquad |$$

$$B, \neg F, U_{0\,0} \overset{\times}{\to} U_{0\,0} \qquad B, \neg F, U_{1\,0} \supset U_{0\,k} \to U_{0\,k} \tag{26}$$

$$\backslash \qquad\qquad \vdots$$

$$B, \neg F, T \overset{\times}{\to} T \qquad B, \neg F, U_{0\,1} \supset U_{0\,0} \to U_{0\,0} \tag{27}$$

$$\backslash \qquad\qquad |$$

$$B, \neg F, U_{0\,0} \supset T \to T \tag{28}$$

$$|$$

$$B, \neg F, \neg T \; \to \tag{29}$$

Further, (29)–(24), yields a blue SLD-metaproof $\Delta(F)$ of F:

$$U_{(j+1)\,0}, U_{(j+1)\,0} \supset U_{j\,k_j}, U_{j\,k_j}, \ldots, U_{j\,0}, U_{j\,0} \supset U_{j-1\,k_{(j-1)}}, \ldots, \tag{30}$$
$$U_{1\,0}, U_{1\,0} \supset U_{0\,k}, U_{0\,k}, \ldots, U_{0\,0}, U_{0\,0} \supset T, T, \neg F, \neg T, \square, F.$$

A contradiction, $\square$, of (22) giving F, and $j+1, k \leq c \cdot |F|^n$.

In $\Delta(F)$, there is no first formula $G \notin PROP$.[5] So, cf. Definition 4,

$$\vdash_{R_p} F. \tag{31}$$

It remains to calculate the size of $\Delta(F)$. For all F such that $|d| < |F|$, the size of all $U_{j\,k}\, T \in PROP$ in $\Delta(F)$ has an upper bound $c \cdot |F|^n$. The indices have an upper bound $j+1, k \leq c \cdot |F|^n$.

The size of deducing $\neg T$ (29) has a similar upper bound, by (17)–(20).

Hence, the size $|\Delta(F)|$ of $\Delta(F)$ is:

$$|\Delta(F)| \in O(|F|^{3n}) \text{ sufficiently large } F \text{ for } |d| \leq |F|. \tag{32}$$

Therefore, Definition 4, (32) and discharging the assumption (21) give,

$$\text{if } SAT \in \mathcal{P} \text{ then } |\vdash_{R_p} F| \in O(|F|^n) \text{ some } n \in Z^+ \text{ all} \tag{33}$$
$$\text{sufficiently large } F \in TAUT \text{ in DNF.} \qquad \square$$

[5] Robinson unifications are implicit. Writing $\Gamma, L \overset{\times}{\to} L, \Theta$ for unified prime predicates $L \in \{T, U_{j\,k}\} \subset PROP$. Γ, and Θ are any list of formulas, using Kleene $G4$ notation.

5 $SAT \notin \mathcal{P}$ and $\mathcal{P} \neq \mathcal{NP}$

Haken's theorem,[6] and Lemma 1 give, by reductio ad absurdum,

Theorem 1 *$SAT \notin \mathcal{P}$.*

Proof.

$$\text{Assume } SAT \in \mathcal{P}. \text{ Then, using Lemma 1,} \tag{34}$$

$$|\vdash_{R_p} F| \in O(|F|^n) \text{ some } n \in Z^+ \text{ all sufficiently large } F \in TAUT \text{ in} \tag{35}$$

DNF, *i.e.*, $|d| \leq |F|$, (32). But, Haken's Theorem and (35) are a contradiction using $PF_n \in TAUT$ for sufficiently large n.

$$\text{Consequently, } SAT \notin \mathcal{P}. \text{ (There is no Turing machine } d, \text{ (13)).} \quad \square \tag{36}$$

Therefore, by Theorem 1 and $SAT \in \mathcal{NP}$, see Cook [4],

Theorem 2 *$\mathcal{P} \neq \mathcal{NP}$.*

Corollary 4 *$TAUT \notin \mathcal{P}$.*

Acknowledgment

Keith Clark, Hanna-Nina Ekelund, Niklas Ekelund, Pierre Flener, Andreas Hamfelt, Svante Janson, Torsten Palm, Alan Robinson, Bo Steinholtz, Anton Tärnlund, and participants of The Stockholm-Uppsala Logic Seminar 02/03/2010, and The AI-course 10/17/2014 thank you all.

References

[1] Keith L. Clark. *Predicate Logic as a Computational Formalism.* Research Report DOC 79/59, Department of Computing, Imperial College, London, UK, 1979.

[2] Keith L. Clark and Sten-Åke Tärnlund. A first order theory of data and programs. In Bruce Gilchrist, editor, *Information Processing 77*, volume 7, pages 939–944, Amsterdam, The Netherlands, 1977. North-Holland.

[3] A. Colmerauer, H. Kanoui, R. Pasero, and P. Roussel. Un Système de Communication Homme-machine en Français. Technical report, Groupe Intelligence Artificielle, Université d'Aix-Marseille, Luminy, 1973.

[4] Stephen Cook. The complexity of theorem-proving procedures. In *Third Annual ACM Symposium on Theory of Computing*, pages 151–158, New York, NY, USA, 1971. ACM Press.

[5] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, 1979.

[6] Writing PF_n (in DNF) for n pigeons in $n+1$ holes and an empty hole all $n \in Z^+$.
Lemma. (Cook and Reckhow [5]) For all n, PF_n is a tautology.
Theorem. (Haken) There exists a constant c, $c > 1$, so that, for sufficiently large n, every resolution proof of PF_n, contains at least c^n different clauses.

[6] Kurt Gödel. Über die Länge von Beweisen. *Ergebnisse eines mathematischen Kolloquims, Heft 7*, 7:23–24, 1936.

[7] Armin Haken. The intractability of resolution (complexity). *Theoretical Computer Science*, 39:297–308, 1985. PhD thesis University of Illinois at Urbana-Champaign 1984.

[8] Åke Hansson and Sten-Åke Tärnlund. Program Transformation by Data Stucture Mapping. In Keith L. Clark and Sten-Åke Tärnlund, editors, *Logic Programming*, pages 117–122. Academic Press, London, UK, 1982.

[9] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[10] David Hilbert and Wilhelm Ackermann. *Grundzüge der theoretischen Logik*. Springer-Verlag, 1928. Reprinted 1972; In English by Lewis Hammond et al., Principles of Mathematical Logic, Chelsea, New York, 1950.

[11] David Hilbert and Paul Bernays. *Grundlagen der Mathematik*, volume 1. Springer-Verlag, Berlin, 1934. Volume 2, 1939.

[12] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, USA, 1972.

[13] Stephen C. Kleene. *Mathematical Logic*. John Wiley and Sons, New York, USA, 1967. First corrected printing, March, 1968.

[14] Robert A. Kowalski. Predicate Logic as a Programming Language. In J.L. Rosenfeldt, editor, *Information Processing 74*, pages 569–574, Amsterdam, The Netherlands, 1974. North-Holland.

[15] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1):23–41, 1965.

[16] Sten-Åke Tärnlund. *Logic Information Processing*. TRITA-IBADB-1034, The Royal Institute of Technology, Stockholm, Sweden, 1975. Horn clause computability, BIT vol 17, no 2, 215–226, 1977.

[17] Sten-Åke Tärnlund. $\mathcal{P}$ is not equal to $\mathcal{NP}$. *DiVA e-prints*, December 2019. 11th edition.

[18] Alan M. Turing. On Computable Numbers with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2/46:230–265, 1936.

[19] David H. D. Warren. *Applied logic - its use and implementation as a programming tool*. Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK, 1977. Reprinted in Technical Report 290, 1983, AI Center, SRI International, Menlo Park, CA, USA.