Verifying that P is not equal to NP using a theorem prover

Sten-Åke Tärnlund

Division of Information Systems
Research Report 2013:2
(Second edition of report 2012:1)
Verifying that $\mathcal{P}$ is not equal to $\mathcal{NP}$ using a theorem prover*

Sten-Ake Tärnlund†‡

September 10, 2013

Abstract

The problem of computing whether any formula in propositional logic is satisfiable is not in $\mathcal{P}$. Therefore, $\mathcal{P}$ is not equal to $\mathcal{NP}$. In the first-order theory $\mathcal{B}$, axiomatizing Turing’s theory of computing, three versions of the proofs are presented. First, an informal (conceptual) proof about formal proofs. Second, a more formal proof in Hilbert’s proof theory. Third, a formal proof in Hilbert’s proof theory, using a theorem prover.

1 Introduction

Let $\text{SAT}$ be the set of satisfiable formulas of propositional logic. The satisfiability problem for whether $p \in \text{SAT}$, all formulas $p$ of propositional logic. Let $\mathcal{P}$ be the set of problems with a solution of a deterministic Turing machine in polynomial computing time. Let $\mathcal{NP}$ be the set of problems with a solution of a nondeterministic Turing machine in polynomial computing time. $\text{SAT} \in \mathcal{P}$ for the satisfiability problem is in $\mathcal{P}$.

Turing’s [23] theory of computing is axiomatized in the first-order Theory $\mathcal{B}$ with an Axiom $B$ that characterizes a universal Turing machine. Thus, Axiom $B$ defines computing, cf. Sections 2–4.

Theorem 1: $\text{SAT} \not\in \mathcal{P}$, and Theorem 2: $\mathcal{P} \neq \mathcal{NP}$, are proved (three times), in a conservative extension of Theory $\mathcal{B}$. The third version is by the automated theorem prover Vampire, cf. Riazanov and Voronkov [14], in Hilbert’s [7, 8] proof theory.

Consequently, Vampire thereby verifies that there are proofs of $\text{SAT} \not\in \mathcal{P}$, and $\mathcal{P} \neq \mathcal{NP}$ in Theory $\mathcal{B}$, cf. Tärnlund [19, 20].

1.1 Overview

The first order formalization of Axiom $B$ connects computing with foundations of mathematics e.g., proof theory. As a consequence, Lemma 1 follows in Theory $\mathcal{B}$. Informally, it states: if $\text{SAT}$ is in $\mathcal{P}$ then there exists a formal deduction

---

*Second edition. No error has been found, but the presentation is simplified. Vampire’s proofs are not changed, but there is a change of names. Corollary 6, of the first edition, is now Definition 12, and the old Definition 13 is deleted.

†Copyright © 2012 by Sten-Ake Tärnlund, Stockholm, Sweden.

‡gmail name stenake

1Vampire’s proofs are computed at the useful TPTP-site of Sutcliffe 2009 [17].
of $F$ of polynomial size (in the size of $F$) in Robinson [15] resolution systems for all sufficiently large tautologies $F$ on disjunctive normal form (DNF).\footnote{Formally, $SAT \in \mathcal{P} \supset \vdash_n F \in O(|F|^{2n+1})$ some $n \in \mathbb{Z}^+$ all sufficiently large $F \in \text{TAUT}$ on DNF.} The proof of this new result uses textbook methods, cf. Kleene [9].

Haken’s [5] theorem gives, directly, Corollary 6, which states: it is not the case that there exists a formal deduction of $F$ of polynomial size (in the size of $F$) in Robinson resolution systems for all sufficiently large tautologies $F$ on disjunctive normal form.\footnote{Formally, $\neg (\vdash_n F \in O(|F|^{2n+1})$ all sufficiently large $F \in \text{TAUT}$ some $n \in \mathbb{Z}^+$ on DNF).}

Thus, Lemma 1 and Corollary 6 give a proof of Theorem 1: $SAT \not\in \mathcal{P}$. Then, Theorem 2: $\mathcal{P} \neq \mathcal{N\mathcal{P}}$ follows directly, because $SAT \in \mathcal{N\mathcal{P}}$.

1.1.1 Outline of the proofs

The three versions of the proofs of Lemma 1, Theorem 1: $SAT \not\in \mathcal{P}$, and Theorem 2: $\mathcal{P} \neq \mathcal{N\mathcal{P}}$ are successively more formalized.

First, an informal proof about formal proofs, in a simply consistent conservative extension of Theory $B$, cf. Section 1.3 and Section 1.3.1. The proof idea is similar in Tärnlund [19, 20], but the presentation corresponds more to the proofs of Tärnlund [21].

Second, the proofs are carried out in Hilbert’s proof theory, in the extended Theory $B$, see Sections 2–4.

Third, formal proofs, in Hilbert’s proof theory, using Vampire. The proofs are developed in the extended Theory $B$ that is re-written in the syntax of Vampire, with a precision and formalization needed by Vampire for its automated proofs, cf. Section 5.

1.1.2 Summary of Vampire’s proofs

Vampire proves, in Hilbert’s proof theory, the following results in a simply consistent conservative extension of Theory $B$.

(i) Vampire proves Corollary 1 and Corollaries 3–5, cf. Sections 5.1–5.4. Vampire uses formalized information of Sections 2–4, e.g., Definition 11.

(ii) Vampire proves Lemma 1 using Corollary 1 and Corollaries 3–5, cf. Section 5.5.

(iii) Vampire proves Theorem 1: $SAT \not\in \mathcal{P}$ from Lemma 1, and Corollary 6 that follows directly from Haken’s theorem, cf. Section 5.6.

(iv) Vampire proves Theorem 2: $\mathcal{P} \neq \mathcal{N\mathcal{P}}$ from Theorem 1: $SAT \not\in \mathcal{P}$ and Definition 13; $\mathcal{P} = \mathcal{N\mathcal{P}} \supset SAT \in \mathcal{P}$, the if-part of the Cook-Levin [3, 11] theorem, cf. Section 5.7.

(v) Hence, Vampire verifies, in Hilbert’s proof theory, that Theorem 1 and Theorem 2 have proofs in an extension of Theory $B$, cf. Tärnlund [19, 20].

1.2 Some elements of Hilbert’s proof theory

Briefly, in Hilbert’s [7, 8] proof theory, theories, axioms, inference rules, and proofs are formal mathematical objects, cf. Kleene [9].

The following notions of Hilbert’s proof theory are used:

\footnote{SAT $\in \mathcal{P} \equiv \mathcal{P} = \mathcal{N\mathcal{P}}$, cf. Sipser 2005 [16].}
(i) formal axiom, (ii) formal deduction, (iii) the existence of a formal deduction, and (iv) simple consistency. They are presented in more detail in Sections 1.2.1–1.2.4.

1.2.1 Formal axiom

The first-order theory $B$ has an Axiom $B$,\textsuperscript{5} which is introduced in Axiom 1.

Axiom $B$ connects computing with foundations of mathematics e.g., proof theory. As a consequence, Lemma 1 follows in Theory $B$.

Conceptually, $B$ is also a logic program, cf. Kowalski [10]. Any formal proving from axiom $B$ is computing, any computation can be done in a formal deduction from $B$ (Turing’s thesis), and the size of a computation is the size of the proof (the number of symbols in the proof).

In addition to axiom $B$ several significant definitions are used. For example, the notion of computing time, that is, the number of moves of the head of a Turing machine in a computation, cf. Hartmanis and Stearns [6], is introduced for formal deductions in Definition 2. The set $U$ of all deterministic Turing machines that compute whether a proposition is satisfiable is introduced in Definition 7. The formula $SAT \in \mathcal{P}$ is introduced in Definition 9. The formal deduction $\Delta(z)$ in computing time $z$ is introduced in Definition 11. The size of a formal deduction, in Robinson resolution systems, is introduced in Definition 12. The formula $\mathcal{P} = \mathcal{NP}$ is introduced in Definition 13.

However, the definitions are non-creative, and are conservative extensions of theory $B$.

For convenience, number theory, cf. Kleene [9], and a theory of data and programs, cf. Clark and Tärnlund [1], are used to prove properties of computations.

1.2.2 Formal deductions

The notion of a formal deduction, in the proof of Lemma 1, and Theorem 1, is fundamental.

Formal deductions are defined by the inference rules of the inference systems. In Lemma 1, and its proof, Robinson resolution systems are themselves mathematical objects, and identified by the formal deduction.

Kleene’s G4 system is used in the proofs of all three versions of Lemma 1, cf. Section 1.3, Section 4.3 and Section 5.5.\textsuperscript{6}

However, Vampire, uses Robinson resolution systems for its automated proof of the third version of the proofs, e.g., about Kleene’s G4 systems, cf. Section 5.

For instance, a formal propositional deduction of Robinson resolution systems can be introduced as follows.\textsuperscript{20}

Writing

\begin{equation}
F_1, F_2, \ldots, F_n \text{ of formulas. } F_i \text{ and } F_j \text{ are an assumption, an axiom, or a deduced formula. } F_k \text{ follows from } F_i \text{ and } F_j \text{ by the inference rule (43). } F_n \text{ is the deduced formula, } 1 \leq i, j < k \leq n.
\end{equation}

\textsuperscript{5}Axiom $B$ is a slight simplification of the formulation originally given by Tärnlund [18].

\textsuperscript{6}Kleene’s G4 system is a choice of convenience, other systems can also be used, e.g., Robinson resolution systems, cf. Proposition 1 in footnote 9.
To check that a formal deduction is, indeed, a formal deduction, all that is needed is to check that the applications of the inference rules are correct. This is a laborious task that can be left to computers. Of course, being a theorem prover, Vampire not only checks formal deductions, but constructs them.

It should be noted that interpretations have no part in a formal deduction. Simply, there is no inference rule for them.

If Hilbert’s proof theory only allowed formal deductions, it would not reach as far as easily. Fortunately, informal proofs about formal deductions are allowed in Hilbert’s proof theory, if they can be converted into formal deductions.

In Hilbert’s proof theory, metamathematics is like mathematics with a difference, that is, the proofs are either formal deductions or informal deductions that can be converted into formal deductions.

1.2.3 Existence of a formal deduction

The following two notions are used in Lemma 1. In Kleene G4 systems, the existence of a formal deduction is defined as follows.

\[ \vdash B \rightarrow F \text{ for there exists a formal deduction of } F \]  

from axiom \( B \) in Kleene G4 systems.

A polynomial upper bound on the size \( |\vdash F| \) of a formal deduction of \( F \) that exists in Robinson resolution systems is written as follows, using the big-O notation.

\[ |\vdash F| \leq O(|F|^n) \text{ for the size } |\vdash F| \text{ of a formal deduction of } F \text{ that exists in Robinson resolution systems has an upper bound } c \cdot |F|^n \]  

all \( n \in N \) some \( c \in N \) all sufficiently large tautologies \( F \).

1.2.4 Simple consistency

Theory \( B \) is simply consistent\(^7\) in the subset \( U \) of deterministic Turing machines that compute whether a proposition is satisfiable, see Corollary 2.

Further, the applications of number theory \( N \), for example to compute the size, and the computing time of a formal deduction, are simply consistent in this subset of theory \( B \).

1.3 Lemma 1 and a proof: first version

Axiom \( B \), cf. Axiom 1, is the essence of the proof of Lemma 1, conceptually, it is also a logic program. The proof, using a deterministic Turing machine, of Lemma 1 corresponds to the proof in Tärnlund [21], and is similar to the proof, using a nondeterministic Turing machine, of \( TAUT \notin NP \) in Tärnlund [22].

The nonempty set \( U \) in Theory \( B \) is introduced as follows, cf. Definition 7.

Let \( U \) be the set of deterministic Turing machines that compute whether \( G \) is satisfiable, with output \( \emptyset \cdot 0 \), or not, with output \( \emptyset \cdot 1 \), for all propositions \( G \).

---

\(^7\)A theory is simply consistent if for no formula \( B \) are both \( B \) and \( \neg B \) provable in it.
Assume that, a deterministic Turing machine with the name $i$, $i \in U$, that computes the output in computing time $c \cdot |G|^n$ exists some $c n \in Z^+$ all propositions $G$.

If $SAT \in \mathcal{P}$ then if $i$ for input $\neg F$ computes $\emptyset . 1$ then $F$ for all tautologies $F$. More formally in Theory $B$, by Axiom $B$, (4)–(5), cf. Definition 7, and Corollary 1.

If $SAT \in \mathcal{P}$ then $T(i, \neg F, \emptyset, \emptyset, 1) \supset F$ all tautologies $F$ on DNF. (6)

If $SAT \in \mathcal{P}$ then there is a proof of $F$ from axiom $B$ in computing time $c \cdot |F|^n$ some $c n \in Z^+$ all tautologies $F$ on DNF in Kleene G4 systems. More formally, in Theory $B$, by Axiom $B$, (5)–(6), and Definition 2, cf. Corollary 3.

If $SAT \in \mathcal{P}$ then there is a proof from axiom $B$ in computing time $c \cdot |F|^n$ some $c n \in Z^+$ all tautologies $F$ on DNF in Kleene G4 systems. (7)

If $SAT \in \mathcal{P}$ then the size of a proof of $F$ that exists in Robinson resolution systems has a polynomial upper bound (in the size of $F$) for all sufficiently large tautologies $F$ on DNF. This is written more formally as follows, using (3).

**Lemma 1** If $SAT \in \mathcal{P}$ then $|\vdash_r F| \in O(|F|^{2 \cdot n + 1})$ some $n \in Z^+$ all sufficiently large tautologies $F$. (8)

Proof.

Assume that,

$SAT \in \mathcal{P}$. (9)

Then, there is a proof of $F$ from axiom $B$ in computing time $c \cdot |F|^n$ some $c n \in Z^+$ all tautologies $F$ on DNF in Kleene G4 systems, by (7). Therefore,

$\vdash B \rightarrow F$ in $c \cdot |F|^n$ some $c n \in Z^+$ all tautologies $F$ on DNF. (10)

A proof tree of the predicate calculus sequent $B \rightarrow F$ is computable (breadth-first) in polynomial $c \cdot |F|^n$ computing time. For reasons of space, the propositional atoms $U_{jk}$ and $T$ are used as short names for propositional instantiations of the $U$ and $T$ predicates of Axiom $B$.\(^8\) The names are enumerated by the indexes, where $j$ is the computing time, and $k$ is the searching time for a quintuple of the code, of the deterministic Turing machine $i$.

Proof tree 1 below is computed in polynomial computing time, i.e., in at most polynomially many head moves of $i$ (in the size of $F$) along a branch. The search time for a quintuple is less than the size $|i|$ of $i$.\(^9\)

---

\(^8\)The sizes of the instantiations of the predicates $U$ and $T$ of axiom $B$ are used for computing the size of a proof, not the sizes of the short names.

\(^9\)Proof tree 1 is computable in computing time $c \cdot |F|^n$ from axiom $B$, using Robinson resolution systems, by induction on the computing time.

**Proposition 1** If Turing machine $i$ for input $a$ computes output $u$ in polynomial computing time (in the size of $a$) then $B \vdash_r T(i, a, u)$ in polynomial computing time (in the size of $a$) all $i \in M a \in R u \in L$. 

5
Proof tree 1 A (condensed) proof of the sequent $B \rightarrow F$ in polynomial computing time $z$, i.e., $z \leq c \cdot |F|^n$, follows, $F \in \text{TAUT on DNF}$.

\[
\begin{array}{c}
B, U_{(z-1)k_2} \not\rightarrow U_{(z-1)k_2} & B, U_{z0} \not\rightarrow U_{z0} \\
\vdots & \\
B, U_{00} \not\rightarrow U_{00} & B \rightarrow U_{01} \\
B, T \not\rightarrow T & B, U_{01} \supset U_{00} \rightarrow U_{00} \\
B, F \not\rightarrow F & B, U_{0n} \supset T \rightarrow T, \ by \ \supset \rightarrow \ \forall \rightarrow \\
\vdots & \\
B, T(i, \neg F, \emptyset, \emptyset, 1) \supset F \rightarrow F, \ by \ (6) \\
B \rightarrow F
\end{array}
\]

The proof tree is computed breadth-first, i.e., all choices of Axiom $B$ are explored with no backtracking on the logic program $B$. The entire proof tree is closed at computing time $z \leq c \cdot |F|^n$ by the proposition $U_{z0}$.

A formal propositional proof $\Delta(c \cdot |F|^n)$ of $F$ is computable from $U_{z0}$, of the top leaf node $B, U_{z0} \not\rightarrow U_{z0}$, downward in the tree. However, the proof path from $U_{z0}$ to $F$ is not decidable until $U_{z0}$ is computed in computing time $c \cdot |F|^n$.

In summary, $\Delta(c \cdot |F|^n)$ is the sequence of the blue formulas, instantiations of axiom $B$ (except $F$ of (6)), \(^{10}\) it begins and ends as follows.

\[
U_{z0}, U_{z0} \supset U_{(z-1)k_2}, U_{(z-1)k_2}, U_{(z-1)k_2} \supset U_{(z-1)(k_2-1)}, \ldots,
\]

\[
U_{10}, U_{10} \supset U_{0k_1}, \ldots, U_{01} \supset U_{00}, U_{00},
\]

\[
U_{00} \supset T, T, T(t, F, \emptyset, \emptyset, 0) \supset F, F.
\]

More precisely, a formal propositional proof $\Delta(z)$ of $F$ in computing time $z$ is introduced in Definition 11, using (enumerations of) the propositional atoms $T$ and $U$ for instantiations of axiom $B$, cf. (11), $z \in \mathbb{Z}^+$, $F \in \text{TAUT on DNF}$.

If (10) then $\Delta(c \cdot |F|^n)$, by Definition 11, and induction on the computing time. Hence,

\[
\Delta(c \cdot |F|^n) \text{ some } c n \in \mathbb{Z}^+ \text{ all tautologies } F \text{ on DNF.} \tag{12}
\]

$\Delta(c \cdot |F|^n)$ is a formal propositional proof of $F$ (on DNF) in Robinson resolution systems, by (44) and induction on the computing time. Thus, if (12) then $\vdash F$ all tautologies $F$ on DNF, by (45). Therefore,

\[
\vdash F \text{ all tautologies } F \text{ on DNF.} \tag{13}
\]

It is sufficient to compute an upper bound on the size $|\Delta(z)|$ of the proof $\Delta(z)$ of $F$ for sufficiently large tautologies $F$, in Robinson resolution systems.

\(^{10}\) $\Delta(c \cdot |F|^n)$ is computable in computing time $c \cdot |F|^n$ from axiom $B$, using Robinson resolution systems, by induction on the computing time, Proposition 1 in footnote 9.
For any size of the deterministic Turing machine $i$, there are sufficiently large tautologies $F$ such that $|i| < |F|$. The size of each atomic propositional formula of $\Delta(z)$ has a polynomial upper bound $c \cdot |F|^n$. Moreover, $z \leq c \cdot |F|^n$ and $k_r < |F|$ for $0 \leq r \leq z$. Therefore, by Definition 11,

$$|\Delta(c \cdot |F|^n)| \in O(|F|^{2n+1})$$

all sufficiently large tautologies $F$ on DNF.

There is a resolution proof of $F$ of polynomial size (in the size of $F$) for all sufficiently large tautologies $F$, by (3), and (13)–(14). Therefore,

$$|\vdash_n F| \in O(|F|^{2n+1})$$

all sufficiently large tautologies $F$ on DNF.

Discharging the assumption (9) ends the proof, and gives Lemma 1.

If $SAT \in \mathcal{P}$ then $|\vdash_n F| \in O(|F|^{2n+1})$ all sufficiently large tautologies $F$ on DNF.

1.3.1 A proof of $SAT \notin \mathcal{P}$ and $\mathcal{P} \neq \mathcal{NP}$: first version

Haken’s theorem, directly, gives: it is not the case that each sufficiently large tautology $F$ (on DNF) has a proof of polynomial size (in the size of $F$) in Robinson resolution systems. This sentence is formalized in Corollary 6.

The sentence: $SAT$ is not in $\mathcal{P}$, has a reductio ad absurdum proof from Lemma 1 and Corollary 6. Therefore,

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

Thus, the Turing machine $i$ in (5) does not exist. However, $SAT \in \mathcal{NP}$, cf. Sipser. Hence, the sentence: $\mathcal{P}$ is not equal to $\mathcal{NP}$, has a reductio ad absurdum proof using Theorem 1. Therefore,

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

A theorem prover asked to prove Lemma 1, Theorem 1, and Theorem 2, would quite likely run into difficulties. Therefore, before Vampire is asked to prove these results in Section 5 more information is formalized in Hilbert’s proof theory.

Axiom $B$ is formalized in Section 2. It characterizes a universal Turing machine, and thus defines computing, but, it is also a logic program on a binary Horn clause form.

In addition, the concepts that Vampire uses are introduced in Sections 2–4.

2 Axiomatizing Turing’s theory of computing

The first-order theory $\mathcal{B}$ of Tärnlund [19, 20] axiomatizes Turing’s [23] theory of computing. It has a finite Axiom $B$, characterizing a universal Turing machine. $B$ is defined in Axiom 1.
In this section, Axiom $B$, the formulas, and the domains of theory $B$ are written down in predicate calculus sufficiently precisely, so that the formal deductions in theory $B$ can be constructed without any interpretation in Hilbert’s proof theory.

2.1 Theory $B$

In theory $B$, there are two predicates:

$$T(i, a, u) \quad \text{and} \quad U(x, s, q, j, i, u),$$

and one term:

$$\cdot,$$

which is used in infix notation, e.g., $x.y$ is a list of an element $x$ and a list $y$.

The domains of the predicates and terms are as follows.

Let $K$ be a finite set of constant symbols that a Turing machine may use, for example, the alphabetic symbols, the natural numbers, and the symbols of propositional logic. The following subset of symbols are included in predicate calculus to define a universal Turing machine,

$$\{\emptyset, 0, 1\} \subseteq K.$$  

The input symbols$^{11}$ of a Turing machine also belong to the set $K$.

In theory $B$, there are six sets for the Turing machines.

First, using the set $K$, a finite set $S$ of symbols is defined writing

$$S \text{ for } \{ u : u \in K \lor (u = r \cdot \emptyset \lor u = \emptyset \cdot r) \land r \in K \}.$$  

The two element symbols $\emptyset \cdot r$ and $r \cdot \emptyset$ (two one-element lists) $r \in K$, are used to grow the two-way tape with one element to the left or to the right, respectively.

Second, a finite set of states is defined writing

$$Q \text{ for } \{0, 1\} \text{, where } 0 \text{ is the halt state and } 1 \text{ is the start state.}$$  

Third, the set $D$ of moves of the tape head of a Turing machine is defined writing

$$D \text{ for } \{0, 1\},$$

where $0$ stands for a move to the left and $1$ for a move to the right.

There is a finite arbitrarily large two-way tape, with a left and right tape having an element between them at the tape head.$^{12}$ Initially, the two-way tape has an empty left tape, the input on the right tape, and the element between them has the symbol $\emptyset$. When a computation starts, the tape head reads the symbol $\emptyset$. The finite two-way tape is constructed of two lists.$^{13}$

$^{11}$For example, the set $F$ of propositional formulas of propositional and predicate calculus, including a propositional valid constant $\top$, and the empty symbol $\bot$, cf. Definition 3.

$^{12}$Turing [23] has a one-way tape. For a two-way tape see Post [13].

$^{13}$Historically, Turing [23] and Kleene [9] have a potentially infinite tape. In contrast, Davis [4] and Minsky [12] grow the arbitrarily large finite tape in the computation.
The list on the right tape grows to the right, if the symbol \( r \cdot \emptyset \in S \) replaces \( \emptyset \). The list on the left tape grows to the left, if the symbol \( \emptyset \cdot r \in S \) replaces \( \emptyset \). Therefore the size of the two-way tape grows one element at a time, controlled by the code of a Turing machine.

There are two sets of lists for the two-way tapes, \( R \) for the lists of right tapes, and \( L \) for the lists of left tapes. They are as follows.

A right tape is defined as a list of symbols,

\[
R(\emptyset) \land \forall (s \in S \land R(z) \supset R(s \cdot z)).
\]  
(23)

A left tape is defined as a list of symbols,

\[
L(\emptyset) \land \forall (s \in S \land L(z) \supset L(z \cdot s)).
\]  
(24)

Fourth, using (23), the set \( R \) of right tapes is defined writing

\[
R \text{ for } \{ u : R(u) \}. \]  
(25)

Fifth, using (24), the set \( L \) of left tapes is defined writing

\[
L \text{ for } \{ u : L(u) \}. \]  
(26)

The set \( M \subseteq R \) of codes of Turing machines is a subset of the set of right tapes. The code of a Turing machine, see Turing [23], is a finite list of quintuples,

\[
M(\emptyset) \land \forall (p q \in Q \land r s \in S \land d \in D \land M(z) \supset M(p \cdot s \cdot q \cdot r \cdot d \cdot z)).
\]  
(27)

Sixth, using (27), the set \( M \) of codes of Turing machines is defined writing

\[
M \text{ for } \{ u : M(u) \}. \]  
(28)

The formulas of theory \( B \) are constructed as usual in a first-order theory from the function symbol, predicate symbols, and the domains, see Kleene [9].

Axiom \( B \) defines the predicates \( T(i, a, u) \) and \( U(x, s, z, q, j, i, u) \) of theory \( B \), and a universal Turing machine.

Let \( i \) be the code of a Turing machine, \( a \) be the input list, and \( u \) be the output list, \( i \in M \ a \in R \ u \in L \). Informally,

\[
T(i, a, u) \text{ for } i \text{ with input } a \text{ computes output } u. \]  
(29)

Let \( x \cdot s \cdot z \) be a two-way tape, where \( x \) is the left tape, \( z \) is the right tape, \( s \) is the symbol between them (at the location of the tape head), and let \( j \) be an auxiliary code of Turing machine \( i \), \( i \ j \in M \ s \in S \ z \in R \ q \in Q \ x \ u \in L \). Informally,

\[
U(x, s, z, q, j, i, u) \text{ for } i \text{ with tape } x \cdot s \cdot z, \text{ state } q, \text{ code } j, \text{ computes } u. \]  
(30)

**Example 1** If a Turing machine \( i \) adds an element \( B \) to the input list \( A \cdot \emptyset \) then \( T(i, A \cdot \emptyset, u) \) computes the output list \( \emptyset \cdot A \cdot B \) i.e. \( T(i, A \cdot \emptyset, \emptyset \cdot A \cdot B) \).\(^{15}\)

This computation is done by the universal Turing machine as follows. It starts with the predicate \( U(\emptyset, \emptyset, A \cdot \emptyset, \emptyset, 1, i, i, u) \). Here the two-way tape is

---

\(^{14}\forall F \) for a free variable is universally quantified over the entire formula \( F \).

\(^{15}\)A formula \( A \sqsupset B \) is written \( A \cdot \sqsupset B \cdot \emptyset \) as an input list.
the left tape is the list \( \emptyset \), the symbol at the tape head is \( \emptyset \), the right tape is the list \( A \cdot \emptyset \cdot \emptyset \) that contains the input list \( A \cdot \emptyset \) i.e. a list on a list. This technique with a list on a list is a way to handle the growth of the two-way tape during a computation. Further, the initial state is 1, the code of the Turing machine is \( i \) and \( i \), and the output list is \( u \).\(^{16}\)

The output of the computation is the list \( \emptyset \cdot A \cdot B \) i.e. the halting predicate is \( U(\emptyset \cdot \emptyset \cdot A \cdot B \cdot y \cdot z \cdot 0 \cdot i \cdot i \cdot \emptyset \cdot A \cdot B) \) where \( \emptyset \) is the halt state, some \( y \in S \) \( z \in R \).

### 2.2 Axiom \( B \)

Theory \( B \) has a finite axiom \( B \) comprising a universal Turing machine that is defined in \((33) - (36)\) using the \( U \) predicate, cf. \((30)\). The useful predicate \( T(i, a, u) \), cf. \((29)\), is defined in \((31) - (32)\).

Axiom \( B \) is constructed of five parts as follows.\(^{17}\)

**Part 1.** A definition of \( T(i, a, u) \).

A Turing machine \( i \) with an input list \( a \) computes the output list \( u \) if and only if a universal Turing machine with a two-way tape \( \emptyset \cdot \emptyset \cdot a \cdot \emptyset \) in state 1 with the Turing machine codes \( i \) and \( i \) computes the output list \( u \). Equivalently, this is written in \((31) - (32)\).

\[
T(i, a, u) \supset U(\emptyset \cdot \emptyset \cdot a \cdot \emptyset \cdot 1 \cdot i \cdot i \cdot u) \quad i \in M \quad a \in R \quad u \in L. \tag{31}
\]

\[
U(\emptyset \cdot \emptyset \cdot a \cdot \emptyset \cdot 1 \cdot i \cdot i \cdot u) \supset T(i, a, u) \quad i \in M \quad a \in R \quad u \in L. \tag{32}
\]

The symbols \( \emptyset \), \( 0 \) and \( 1 \) are constants, and \( \cdot \) is an infix term.

**Part 2.** A universal Turing machine that has a two-way tape \( x, s, z \) (with the left tape \( x \)), the halt state \( 0 \), codes \( i \) and \( i \) of a Turing machine, computes the output \( x \), and halts.

\[
U(x, s, z, 0, i, i, x) \quad x \in L \quad s \in S \quad z \in R \quad i \in M. \tag{33}
\]

**Part 3.** If a universal Turing machine has a two-way tape \( x, v, r \cdot z \) (printing \( r \) and moving the head to the left in the previous state), a state \( p \), the codes \( i \) and \( i \), and computes \( u \) then it has a two-way tape \( v, s, z, \) the previous state \( q \) (reading \( s \)), a quintuple \( q, s \cdot p \cdot r \cdot 0 \) at the front of the code \( j \), the code \( i \), and computes \( u \).

\[
U(x, v, r \cdot z, p, i, i, u) \supset U(x \cdot v \cdot s \cdot z \cdot q \cdot q \cdot s \cdot p \cdot r \cdot 0 \cdot j, i, u) \quad x u \in L \quad v r s \in S \quad z \in R \quad p q \in Q \quad i j \in M. \tag{34}
\]

**Part 4.** If a universal Turing machine has a two-way tape \( x \cdot r, v, z \) (printing \( r \) and moving the head to the right in the previous state), a state \( p \), the codes \( i \) and \( i \), and computes \( u \) then it has a two-way tape \( x, s, v \cdot z, \) the previous state \( q \) (reading \( s \)), a quintuple \( q, s \cdot p \cdot r \cdot 1 \) at the front of the code \( j \), the code \( i \), and computes \( u \).

\[
U(x \cdot r, v, z, p, i, i, u) \supset U(x \cdot s \cdot v \cdot z \cdot q \cdot q \cdot s \cdot p \cdot r \cdot 1 \cdot j, i, u) \quad x u \in L \quad v r s \in S \quad z \in R \quad p q \in Q \quad i j \in M. \tag{35}
\]

\(^{16}\)The input list \( A \cdot \emptyset \) is on the right tape \( A \cdot \emptyset \cdot \emptyset \) of the two-way tape \( \emptyset \cdot \emptyset \cdot A \cdot \emptyset \cdot \emptyset \). Thus, the input list is enclosed with the symbol \( \emptyset \). Moreover, the beginning and the end of the two-way tape itself are marked with \( \emptyset \).

\(^{17}\)In this section the colour blue is used for highlighting.
Part 5. If a universal Turing machine has a two-way tape \( x, s, z \), a state \( q \), the codes \( j \) and \( i \), and computes \( u \) then it has a two-way tape \( x, s, z \), the state \( q \) (reading \( s \)), a quintuple \( q', s', p, r, d \) at the front of the code \( j \), the code \( i \), and computes \( u \).

\[
U(x, s, z, q, j, i, u) \supset U(x, s, z, q, q', s', p, r, d, j, i, u)
\]

\( xu \in L \land ss' \in S \land z \in R \land ppq' \in Q \land d \in D \land ij \in M \).

Axiom \( B \), cf. Tärnlund [19], comprises the five parts (31)–(36).

**Axiom 1 \( B \) for**

\[
T(i, a, u) \supset U(\emptyset, \emptyset, a, \emptyset, 1, i, i, u) \quad i \in M \land a \in R \land u \in L.
\]

\[
U(\emptyset, \emptyset, a \cdot \emptyset, 1, i, i, u) \supset T(i, a, u) \quad i \in M \land a \in R \land u \in L.
\]

\[
U(x, s, z, 0, i, i, x) \land x \in L \land s \in S \land z \in R \land i \in M.
\]

\[
U(x, v, r \cdot z, p, i, i, u) \supset U(x, v, s, z, q, q \cdot s \cdot p \cdot r \cdot 0, j, i, u) \quad xu \in L
\]

\[
vrs \in S \land z \in R \land ppq \in Q \land ij \in M.
\]

\[
U(x, r, v, z, p, i, i, u) \supset U(x, s, v \cdot z, q, q \cdot s \cdot p \cdot r \cdot 1, j, i, u) \quad xu \in L
\]

\[
vr \in S \land z \in R \land ppq \in Q \land ij \in M.
\]

\[
U(x, s, z, q, j, i, u) \supset U(x, s, z, q, q' \cdot s', p, r, d, j, i, u) \quad xu \in L
\]

\[
rvs' \in S \land z \in R \land ppq' \in Q \land d \in D \land ij \in M.
\]

Here, \( \emptyset, 0 \) and \( 1 \) are constants, and \( \cdot \) is an infix term.

The free variables have the generality interpretation.

The domains of axiom \( B \) are: the set \( S \) of symbols (20), the set \( Q \) of states (21), the set \( D \) of head moves (22), the set \( R \) of right tapes (25), the set \( L \) of left tapes (26), and the set \( M \) of codes of Turing machines (28).

It is convenient to define the domains using set abstraction. Without set abstraction, the computing theory could be developed similarly to the programming theory of Clark and Tärnlund [1].

### 2.3 Deterministic and nondeterministic Turing machines

The set \( D \) of deterministic Turing machines is a subset of the set \( M \) (of codes) of Turing machines.

Let \( t \) and \( t' \) be quintuples of a Turing machine \( i \in M \).

**Definition 1 \( D \) for**

\[
\{ i : i \in M \land t = q \cdot s \cdot p \cdot r \cdot d \land t' = q' \cdot s' \cdot p' \cdot r' \cdot d' \land q \land p \land p' \in Q \land s \land r \land r' \land d \land d' \in D \} \text{ no quintuples } t \text{ and } t' \text{ of } i \in M.
\]

A nondeterministic Turing machine belongs to the set \( M \) of Turing machines, but not to the set \( D \) of deterministic Turing machines.

As a logic program, cf. Kowalski [10], axiom \( B \) can be run by interpreters and compilers, for example, using Prolog.\(^{18,19}\)

Writing a few Turing machines \( i \) with input \( a \), and formally proving (running) \( \exists u T(i, a, u) \) from \( B \), by a theorem prover, is a fast way to comprehend axiom \( B \). There are some examples in Tärnlund [19].

\(^{18}\)A Prolog interpreter was first developed by Colmerauer, Kanou, Pasero and Roussel [2]. Warren [24] has developed a Prolog system including a compiler that runs logic programs (more) efficiently.

\(^{19}\)Logic programs are computationally efficient, cf. Proposition 1 in footnote 9.
3 Computing time in theory B

A Turing machine computation is a formal deduction in theory B. The computing time is the number of moves of the head of a Turing machine in a formal deduction (computation) in theory B.

The formal deduction of $\exists u T(i, a, u)$ from axiom $B$ that exists in computing time less or equal to $z$, in Kleene G4 systems, is defined as follows, $i \in M, a \in R, z \in Z^+$.\(^6\)

**Definition 2** $\vdash B \rightarrow \exists u T(i, a, u)$ in $z$ for there exists a formal deduction of $\exists u T(i, a, u)$ from $B$ in $G4$, and the number of moves of the tape head of $i$ is less or equal to $z$ in the formal deduction all $i \in M, a \in R, z \in Z^+$.

A few useful definitions follow.

**Definition 3** $\mathcal{F}$ for the set of propositional formulas of propositional and predicate calculus.

**Definition 4** $\text{SAT}$ for $\{ F : F \in \mathcal{F} \land \not \models \neg F \}$.

**Definition 5** $\text{TAUT}$ for $\{ F : F \in \mathcal{F} \land \models F \}$.

3.1 Computing satisfiability deterministically

Let $s$ be a deterministic Turing machine that exists and computes whether a proposition $F$ is satisfiable, and outputs $\emptyset.0$, or unsatisfiable, and outputs $\emptyset.1$, $s \in \mathcal{D}, F \in \mathcal{F}$. Formally,

**Definition 6** $T(s, F.\emptyset, u)$ for $(u = \emptyset.0 \land \not \models \neg F) \lor (u = \emptyset.1 \land \models \neg F))$ all $F \in \mathcal{F}, u \in L$.

Let $\mathcal{U}$ be the set of deterministic Turing machines that compute the same output as Turing machine $s$ does for input $F.\emptyset$ all $F \in \mathcal{F}$, by Definition 1 and Definition 6.

**Definition 7** $\mathcal{U}$ for $\{ i : i \in \mathcal{D} \land \exists u (T(i, F.\emptyset, u) \land T(s, F.\emptyset, u)) \}$ all $F \in \mathcal{F}$.

In Theory B, if $T(i, \neg F.\emptyset, \emptyset.1)$ then $F, i \in \mathcal{U}$ $F \in \mathcal{F}$, using Axiom $B$ and Definitions 6–7, cf. (6). Formally,

**Corollary 1** $T(i, \neg F.\emptyset, \emptyset.1) \supset F$ all $i \in \mathcal{U}$ $F \in \mathcal{F}$.

Vampire’s automated proof of Corollary 1 is in Section 5.1.

Theory B is simply consistent in $\mathcal{U}$, that is, there is no contradiction. Formally,

**Corollary 2** $\not \vdash B \rightarrow (\exists u T(i, F.\emptyset, u) \land \neg \exists u T(i, F.\emptyset, u))$ all $i \in \mathcal{U}$ $F \in \mathcal{F}$.

Let $E$ be an expression (term or formula) of predicate calculus (including propositional calculus).

**Definition 8** $|| : E \rightarrow N$ for a function that computes the size (the number of symbols) of $E$. 

12
3.2 A definition of $\text{SAT} \in \mathcal{P}$ in theory $B$

Next, $\text{SAT} \in \mathcal{P}$ is introduced in theory $B$, using Definition 2 and Definition 7.

$\text{SAT} \in \mathcal{P}$ for there exists a formal deduction of $\exists u T(i, F, \emptyset, u)$ from axiom $B$ in polynomial computing time $c |F|^n$ in Kleene G4 systems, some $i \in \mathcal{U}$ $n$ $c \in Z^+$ all $F \in \mathcal{F}$. Formally,

**Definition 9** $\text{SAT} \in \mathcal{P}$ for there exists a formal deduction of $T(i, \neg F, \emptyset, \emptyset, 1)$ from axiom $B$ in polynomial computing time, in the size of $F$, for some deterministic Turing machine $i \in \mathcal{U}$ for all tautologies $F \in \text{TAUT}$.

**Corollary 3** $\text{SAT} \in \mathcal{P}$ for $\exists \, \exists u T(i, F, \emptyset, u)$ in $c |F|^n$ some $i \in \mathcal{U}$ $n$ $c \in Z^+$ all $F \in \mathcal{F}$.

Vampire’s automated proof of Corollary 3 is in Section 5.2.

4 A resolution deduction

Let $A, B, C,$ and $D$ be disjunctions of any (propositional) atoms or negated atoms (including the blank formula $\bot$), and let $P$ be any atom, $A B C D \sqcup P \in \mathcal{F}$. The propositional inference rule of Robinson resolution systems is defined as follows.

\[
\begin{array}{c}
A \lor P \lor B, \quad C \lor \neg P \lor D \\
\hline
A \lor B \lor C \lor D
\end{array}
\]

(43)

Let $F_1, \ldots, F_n$ be a finite sequence of propositional formulas in $\mathcal{F}$. A formal deduction of Robinson resolution systems can be defined as follows. Writing,

\[
\begin{array}{c}
\text{a formal deduction of Robinson resolution systems for a sequence} \\
F_1, F_2, \ldots, F_n \text{ of formulas. } F_i \text{ and } F_j \text{ are an assumption, an axiom, or} \\
a deduced formula. } F_k \text{ follows from } F_i \text{ and } F_j \text{ by the inference} \\
rule (43). } F_n \text{ is the deduced formula, } 1 \leq i, j < k \leq n.
\end{array}
\]

Let $\Gamma$ be a sequence $A_1, \ldots, A_m$ of assumption formulas. Writing,

\[
\Gamma \vdash_R F_n \text{ for there exists a formal deduction } F_1, \ldots, F_n \text{ from} \\
\text{the assumption } \Gamma \text{ in Robinson resolution systems.}
\]

If $\Gamma$ is the empty sequence in (45), that is, if there is no assumption formula (i.e. $m < 1$), then $\vdash_R F_n$.\(^{20}\)

\[^{20}\text{In Robinson resolution systems a formal deduction of } F_n \text{ is, often, an indirect deduction, that is, } F_n \text{ is on disjunctive normal form, } F_{n-1} \text{ is the blank formula } \bot, \text{ and the assumption } F^* \text{, on conjunctive normal form, is a negation of } F_n. \text{ Hence,} \\
\text{if } \Gamma, F^* \vdash_R \bot \text{ then } \vdash_R F_n.
\]

If $\Gamma$ is the empty sequence, there is no assumption, thus, if $F^* \vdash_R \bot$ then $\vdash_R F_n$.\(^{20}\)
4.1 A formal deduction $\Delta(z)$ in computing time $z$

For reasons of space, the propositional atoms $U_jk$ and $T$ are used as short names for propositional instantiations of the $U$ and $T$ predicates of Axiom $B$. The names are enumerated by the indexes, where $j$ is the computing time, and $k$ is the searching time for a quintuple of the code, of a deterministic Turing machine, cf. Section 1.3.

Then, the formal propositional proof $\Delta(z)$ of $F$ in computing time $z$ is defined as follows, cf. (11), $z \in Z^+, F \in TAUT$ on DNF.

First, a proof $h(z)$ of $(z-1)h$ in computing time from $z$ to $(z-1)$.

Definition 10 $h(z) = < U_{z0} \supset U_{z0} \supset U_{z(k-1)}k^1, U_{z(k-1)}k^2, U_{z(k-1)}(k_{n-1}) \supset U_{z(k_{n-1})}, U_{z(k_{n-1})} \supset U_{z(k_{n-1})} >$ for $0 \leq n < k_z$ all $z \in Z^+$ some $k_z \in N$.

Second, a proof $\Delta(z)$ of $F$ in computing time $z$, defined inductively.

Definition 11 $\Delta(z) = h(z), \Delta(z-1) > all z \in Z^+$.

$\Delta(0) = U_{00} \supset T(i, \neg F, 0, 0, 1), T(i, \neg F, 0, 0, 1) \supset c_i < F, F) all i \in \mathcal{U} F \in TAUT$ on DNF.

If there is a formal deduction of $F$ from $B$ and $T(i, \neg F, 0, 0, 1) \supset F$ in computing time $z$, in Kleene G4 systems then $\Delta(z)$, and $\Delta(z)$ is a propositional formal deduction of $F$ in Robinson resolution systems $i \in \mathcal{U} F \in TAUT$ on DNF $z \in Z^+$. By Definition 11, (43)–(45), and induction on the computing time.

Corollary 4 $\vdash B, T(i, \neg F, 0, 0, 1) \supset F \rightarrow F$ in $z \supset \Delta(z) \land (\vdash F \subset \Delta(z)) all i \in \mathcal{U} F \in TAUT$ on DNF $z \in Z^+$.

Vampire proves Corollary 4 by induction on the computing time, see Section 5.3.

4.2 The size of a formal deduction

The size $|F_1,\ldots,F_n|$ of a formal deduction $F_1,\ldots,F_n$ is defined as the sum of the number of symbols of the formulas in the formal deduction writing

$$|F_1,\ldots,F_n| \text{ for } \sum_{1 \leq k \leq n} |F_k| \text{ all } n \in Z^+. \quad (47)$$

For any size of each deterministic Turing machine $i \in \mathcal{U}$, there are sufficiently large tautologies $F$ such that $i < |F|$. The size of each atomic propositional formula of $\Delta(z)$ has a polynomial upper bound $c \cdot |F|^n$. Moreover, $z \leq c \cdot |F|^n$, and $k_r < |F|$ for $0 \leq r \leq z$, using Definition 11. Hence,

If $\Delta(c \cdot |F|^n)$ in computing time $c \cdot |F|^m$ then $|\Delta(c \cdot |F|^m)|$ has a polynomial upper bound $O(|F|^{2m+1})$, all $c m \in Z^+$ sufficiently large $F \in TAUT$ on DNF.

Corollary 5 $\Delta(c \cdot |F|^m) \supset |\Delta(c \cdot |F|^m)| \in O(|F|^{2m+1}) all c m \in Z^+$ sufficiently large $F \in TAUT$ on DNF.

Vampire’s automated proof of Corollary 5 is in Section 5.4.

There is a polynomial upper bound (in the size of $F$) on the size of a proof of $F$ that exists in Robinson resolution systems if $\vdash F$ follows from the formal deduction $\Delta(c \cdot |F|^m)$ and the size $|\Delta(c \cdot |F|^m)|$ of $\Delta(c \cdot |F|^m)$ has a polynomial upper bound (in the size of $F$), $c m n \in Z^+$ sufficiently large tautologies $F$, cf. (3). Formally,
Definition 12 \( |\vdash_n F | \in O(|F|^n) \) if \( (\vdash_n F \in \Delta(c \cdot |F|^m)) \wedge |\Delta(c \cdot |F|^m)| \in O(|F|^n) \) all \( c \) \( m \) \( n \in Z^+ \) sufficiently large \( F \in \text{TAUT on DNF} \).

4.3 Proofs of Lemma 1 and \( \mathcal{P} \neq \mathcal{NP} \): second version

If \( \text{SAT} \in \mathcal{P} \) then the size of a formal deduction of \( F \) that exists is in \( O(|F|^{2n+1}) \) in Robinson resolution systems, some \( n \in Z^+ \) all sufficiently large \( F \in \text{TAUT} \) on disjunctive normal form (DNF). Formally,

Lemma 1 \( \text{SAT} \in \mathcal{P} \supset \ |\vdash_n F | \in O(|F|^{2n+1}) \) some \( n \in Z^+ \) all sufficiently large \( F \in \text{TAUT on DNF} \).

Proof.

\[ \begin{align*}
\star & \quad \text{SAT} \in \mathcal{P}, \text{ assumption} \quad (48) \\
\star & \quad \vdash B \rightarrow T(i, \neg F, \emptyset, \emptyset, 1) \text{ in } c \cdot |F|^n \text{ some } i \in \mathcal{U} \text{ } c \text{ } n \in Z^+ \text{ all } \quad (49) \\
\quad & \quad F \in \text{TAUT on DNF}, (48) \text{ and Corollary 3} \\
\star & \quad \vdash B, T(i, \neg F, \emptyset, \emptyset, 1) \supset F \rightarrow F \text{ in } c \cdot |F|^n, (49) \text{ and Corollary 1} \quad (50) \\
\star & \quad \Delta(c \cdot |F|^n) \wedge \vdash_n F \subset \Delta(c \cdot |F|^n), (50) \text{ and Corollary 4} \quad (51) \\
\star & \quad |\Delta(c \cdot |F|^n)| \in O(|F|^{2n+1}) \text{ all sufficiently large } F \in \text{TAUT on DNF}, (51) \text{ and Corollary 5} \quad (52) \\
\star & \quad |\vdash_n F | \in O(|F|^{2n+1}) \text{ all sufficiently large } F \in \text{TAUT on DNF}, (53) \text{ (52) and Corollary 5} \\
\quad & \quad \text{SAT} \in \mathcal{P} \supset |\vdash_n F | \in O(|F|^{2n+1}) \text{ some } n \in Z^+ \text{ all sufficiently large } \quad (54) \\
\quad & \quad F \in \text{TAUT on DNF}.
\end{align*} \]

Vampire’s automated proof of Lemma 1 is in section 5.5.

Haken’s theorem, directly, gives: it is not the case that all sufficiently large tautologies \( F \) have a formal deduction of polynomial size in the size of \( F \) on disjunctive normal form (DNF) in Robinson resolution systems. Formally,

Corollary 6 \( \neg(|\vdash_n F | \in O(|F|^m) \text{ all sufficiently large } F \in \text{TAUT on DNF some } m \in Z^+ \}). \)

The statement: \( \text{SAT} \) is not in \( \mathcal{P} \), has an indirect proof from Lemma 1 and Corollary 6, in Hilbert’s proof theory.

Theorem 1 \( \text{SAT} \notin \mathcal{P} \) in a simply consistent conservative extension of theory \( B \).

Proof.

\[ \begin{align*}
\star & \quad \text{SAT} \in \mathcal{P} \text{ in a simply consistent conservative extension of } B \quad (55) \\
\star & \quad |\vdash_n F | \in O(|F|^{2n+1}) \text{ some } n \in Z^+ \text{ all sufficiently large } \quad (56) \\
\quad & \quad F \in \text{TAUT on DNF}, (55) \text{ and Lemma 1} \\
\star & \quad \neg(|\vdash_n F | \in O(|F|^m) \text{ all sufficiently large } F \in \text{TAUT on DNF some } m \in Z^+ \}), \text{ Corollary 6} \quad (57) \\
\quad & \quad \text{SAT} \notin \mathcal{P} \text{ in a simply consistent conservative extension of } B, \text{ contradiction by (56) and (57).} \quad (58)
\end{align*} \]
Vampire’s proof of Theorem 1 is in Section 5.6. It is written,

\[ fof(\text{theorem1}, \text{axiom}, (\sim \text{in}(\text{sat}, p))). \]  

(59)

The next definition is justified in theory \( \mathcal{B} \) by the Cook-Levin theorem.

**Definition 13** \( \mathcal{P} = \mathcal{NP} \supset \text{SAT} \in \mathcal{P} \).

In Vampire, Definition 13 is written,

\[ fof(\text{def13}, \text{axiom}, (p = \text{np} \Rightarrow \text{in}(\text{sat}, p))). \]  

(60)

In Hilbert’s proof theory, Theorem 1 and Definition 13 give next result.

**Theorem 2** \( \mathcal{P} \neq \mathcal{NP} \) in a simply consistent conservative extension of theory \( \mathcal{B} \).

Proof.

\* \( \mathcal{P} = \mathcal{NP} \) in a simply consistent conservative extension of \( \mathcal{B} \)  

\* \( \text{SAT} \in \mathcal{P} \land \text{SAT} \notin \mathcal{P} \), Definition 13 and Theorem 1  

\* \( \mathcal{P} \neq \mathcal{NP} \) in a simply consistent conservative extension of \( \mathcal{B} \),  

contradiction by (62).

Vampire’s proof of Theorem 2 is in Section 5.7. It is written as follows.

\[ fof(\text{theorem2}, \text{axiom}, (p! = np)). \]  

(64)

5 Vampire’s proofs

The information in Sections 2–4 that is used by Vampire, of Riazanov and Voronkov [14], is re-written in the syntax of Vampire. This gives a third version of proofs of Lemma 1, Theorem 1, and Theorem 2 in Theory \( \mathcal{B} \).

The proof system of Vampire is a Robinson resolution system. A proof has a conjecture that Vampire tries to prove by assuming its negation and then deduce a contradiction. If a contradiction is deduced then Vampire has an indirect proof of the conjecture.

The proofs of Vampire are computed on the TPTP-site of Sutcliffe [17].

5.1 Vampire’s proof of Corollary 1

A few definitions are used to prove Corollary 1.

Let \( F \) be any propositional formula in \( \mathcal{F} \), see Definition 3.

\[ n(F) \text{ for } \lnot F. \]  

(65)

The list \( \emptyset \) . 1 stands for unsatisfiable in Definition 6.

\[ u \text{ for } \emptyset \) . 1. \]  

(66)
If $\text{SAT} \in \mathcal{P}$ then there exists a deterministic Turing machine $i \in \mathcal{U}$ that computes whether $\neg f$ is unsatisfiable in polynomial computing time for any tautology $f$. Let $i$ be the name of such a deterministic Turing machine, and let $f$ be any tautology on DNF, cf. (5).

$$t(i, n(f), u) \text{ for } T(i, \neg f, \emptyset, \emptyset).$$

The conjecture of Corollary 1 in the TPTP syntax of Vampire,

$$\text{fof}(\text{cor}1, \text{conjecture}, (t(i, n(f), u) \Rightarrow f)).$$

The proof of Corollary 1 uses Definition 6, but first a few more definitions.

$$\text{val}(F) \text{ for } \models F.$$ The list $\emptyset$ stands for satisfiable in Definition 6.

$s$ for $\emptyset$.0.

Instantiating Definition 6 with Turing machine $i$ that exists cf. (5), gives an auxiliary corollary in Vampire,

$$\text{fof}(\text{cor}10, \text{axiom}, ([!F, U] : (t(i, F, U) \iff ((U = s \& \sim \text{val}(n(F))) \| (U = u \& \text{val}(n(F)))))).$$

Two auxiliary corollaries follow.

A valid double negation of a formula implies that formula,

$$\text{fof}(\text{cor}11, \text{axiom}, (\text{val}(n(n(f))) \Rightarrow f)).$$

The names $s$ and $u$ are different,

$$\text{fof}(\text{cor}12, \text{axiom}, (u! = s)).$$

5.1.1 Proof of Corollary 1

Vampire’s proof of conjecture (68) is reversed i.e. the contradiction comes first.

Proof.

Refutation found.

$$\text{fof}(f272, \text{plain}, ($false$),$$

inference(merge, [], [f21, f27]).

$$\text{fof}(f271, \text{plain}, ($false | $bdd1),$$

inference(subsumption, resolution, [], [f261, f13])

$$\text{fof}(f13, \text{plain}, (u! = s),$$

inference(conversion, [], [f5]).

$$\text{fof}(f5, \text{axiom}, (u! = s),$$

file(/tmp/SystemOnTPTPFormReply27368/SOTy.N4dfT', cor12))

$$\text{fof}(f261, \text{plain}, (u = s | $bdd1),$$

inference(resolution, [], [f241, f11]).
\[fof(f11, plain, (t(i, n(f), u)),
\text{inference}(cnf, transformation, [], \{f7\})).\]

\[fof(f7, plain, ((t(i, n(f), u) \& \sim f), \text{inference}(cnf, transformation, [], \{f2\})).\]

\[fof(f2, negated, conjecture, (\sim (t(i, n(f), u) => f)), \text{inference}(negated, conjecture, [], \{f1\})).\]

\[fof(f1, conjecture, ((t(i, n(f), u) => f)), \text{file}('/tmp/SystemOnTPTPFormReply23549/SOT\_N4df7', cor1)).\]

\[fof(f21, plain, (\sim val(n(n(f))) | \$bdd1), \text{inference}(definition, folding, [], \{f14, fbd1\})).\]

\[fof(f14, plain, (f | \sim val(n(n(f)))), \text{inference}(cnf, transformation, [], \{f8\})).\]

\[fof(f8, plain, (\sim val(n(n(f))) | f), \text{inference}(cnf, transformation, [], \{f4\})).\]

\[fof(f4, axiom, (val(n(n(f))) => f), \text{file}('/tmp/SystemOnTPTPFormReply23549/SOT\_N4df7', cor11)).\]

\[fof(f17, plain, (\sim t(i, X0, X1)), \text{inference}(cnf, transformation, [], \{f10\})).\]

\[fof(f10, plain, (([i][X0, X1] : (val(n(X0)) | s = X1 | t(i, X0, X1))), \text{inference}(cnf, transformation, [], \{f9\})).\]

\[fof(f9, plain, (([i][X0, X1] : ((\sim t(i, X0, X1) | (s = X1 & \sim val(n(X0))) | (u = X1 & val(n(X0)))), \text{inference}(flattening, [], \{f6\}))).\]

\[fof(f6, plain, (([i][X0, X1] : (\sim t(i, X0, X1) | (s = X1 & \sim val(n(X0)))) | (u = X1 & val(n(X0)))), \text{inference}(flattening, [], \{f3\}))).\]

\[fof(f3, axiom, (\sim t(i, X0, X1) | (s = X1 & \sim val(n(X0)))), \text{file}('/tmp/SystemOnTPTPFormReply23549/SOT\_N4df7', cor10)).\]

\[fof(f21, plain, (\sim \$false | \sim \$bdd1), \text{inference}(definition, folding, [], \{f12, fbd1\})).\]

\[fof(f12, plain, (\sim f), \text{inference}(cnf, transformation, [], \{f7\})).\]
5.2 Vampire’s proof of Corollary 3

A few definitions follow.

\[ \text{in(sat, } p) \text{ for } SAT \in \mathcal{P}. \]  (74)

\[ a(b) \text{ for assuming axiom } B. \]  (75)

Let \( f \) be any tautology on DNF, and let \( c \) and \( n \) be some natural numbers.

\[ t(f, n) \text{ for } c \cdot |f|^n \text{ all } n \text{ some } c. \]  (76)

A Vampire notation for the existence of a formal deduction of \( T(i, \neg f, \theta, \emptyset, 1) \) in computing time \( c \cdot |f|^n \) in Kleene G4 systems is defined,

\[ g(a(b), t(i, n(f), u), t(f, n)) \text{ for } \vdash B \rightarrow T(i, \neg f, \theta, \emptyset, 1) \text{ in } c \cdot |f|^n \text{ all } n \text{ some } c. \]  (77)

Recalling, if \( SAT \in \mathcal{P} \) then there exists a Turing machine \( i \in \mathcal{U} \) that computes whether \( \neg f \) is unsatisfiable in polynomial computing time for any tautology \( f \). Let \( i \) be the name of such a deterministic Turing machine, cf. (5), let \( n \) be any natural number, let \( f \) be any tautology on DNF, and let \( b \) be axiom \( B \). The conjecture of Corollary 3 in TPTP syntax,

\[ fof(\text{cor3, conjecture}, (\text{in(sat, } p) \Rightarrow g(a(b), t(i, n(f), u), t(f, n)))), \]  (78)

Two auxiliary corollaries that are used in Corollary 3 follow. They use Definitions (65)–(67).

\[ fof(\text{cor31, axiom}, (\neg g(a(b), t(i, n(f), u), t(f, n)) \Rightarrow \neg t(i, n(f), u)))), \]  (79)

A double negation of a tautology \( f \) is valid if and only if \( f \) or \( \neg f \).

\[ fof(\text{cor32, axiom}, (\text{val}(n(n(f)))) \Leftrightarrow (f \Rightarrow \neg f))). \]  (80)

5.2.1 Proof of Corollary 3

Vampire's proof of conjecture (78) is reversed i.e., the contradiction comes first.

**Proof.**

Refutation found.

\[ fof(f402, \text{plain, } (\text{false}), \text{inference(merge, } [], [f41, f401]))). \]
\[ fof(f401, \text{plain, } (\text{false} | \sim \text{bdd}1), \text{inference(resolution, } [], [f39, f281]))). \]
\[ fof(fbd1, \text{plain, } (f \Leftrightarrow \text{bdd}1), \text{introduced(bddzation, } [])). \]
\[ fof(f281, \text{plain, } (\text{val}(n(n(f)))) | \sim \text{bdd}1), \text{inference(definitionfolding, } [], [f19, fbd1]))). \]
\[ fof(f19, \text{plain, } (\text{val}(n(n(f)))) | \sim f), \text{inference(definitiontransformation, } [], [f13]))). \]
\[ fof(f13, \text{plain, } ((\sim \text{val}(n(n(f)))) | f) \sim f) \& ((\sim f & f) | \text{val}(n(n(f))))), \text{inference(flattening, } [], [f12]))). \]
\begin{verbatim}
of(f12, plain, ((\sim\ val(n(f))) \land (f \sim f)) \land ((f \land f) \land \val(n(f))))).
of(f7, plain, (\val(n(f))) \Rightarrow (f \sim f)).
of(f5, axiom, (\val(n(f))) \Rightarrow (f \sim f)).

\textbf{File (/tmp/SystemOnTPTPFormReply25148/SOT\_v6H6Iw', cor32)).}
of(f39, plain, (\sim\ val(n(f)))).
of(f35, plain, (\sim\ val(n(f))) | u = u).
of(f30, plain, (\sim t(i, n(f), u)).
of(f16, plain, (\sim g(a(b), t(i, n(f), u), t(f, n))).
of(f10, plain, (\sim g(a(b), t(i, n(f), u), t(f, n))).
of(f9, plain, (\sim g(a(b), t(i, n(f), u), t(f, n))).
of(f2, negated, conjecture, (\sim (in(sat, p) \Rightarrow g(a(b), t(i, n(f), u), t(f, n)))).
of(f1, conjecture, (in(sat, p) \Rightarrow g(a(b), t(i, n(f), u), t(f, n))).
of(f17, plain, (\sim t(i, n(f), u) | g(a(b), t(i, n(f), u), t(f, n))).
of(f11, plain, (g(a(b), t(i, n(f), u), t(f, n)) | \sim t(i, n(f), u)).
of(f6, plain, (\sim g(a(b), t(i, n(f), u), t(f, n)) \Rightarrow t(i, n(f), u)).
of(f4, axiom, (\sim g(a(b), t(i, n(f), u), t(f, n)) \Rightarrow t(i, n(f), u)).
of(f26, plain, (\sim t(i, X0, X1) \land \val(n(X0)) | u = X1)).
of(f15, plain, (\sim t(i, X0, X1) \land \val(n(X0)) | (u = X1 & \sim \val(n(X0)))).
of(f14, plain, (\sim t(i, X0, X1) \land \val(n(X0)) | (u = X1 & \sim \val(n(X0))) \land (u = X1 & \val(n(X0))) & (t(i, X0, X1))).
\end{verbatim}
5.3 Vampire’s proof of Corollary 4

A few definitions used in Corollary 4 follow. Corollary 4 uses Corollary 1, cf. (68) in Section 5.1, which needs a different Vampire syntax in Corollary 4 writing

\[ \text{imp}(t(i, n(f), u), f) \text{ for } t(i, n(f), u) \Rightarrow f. \quad (81) \]

Let \( b \) be Axiom B.

\[ a(b, \text{imp}(t(i, n(f), u), f)) \text{ for assuming } b \text{ and } \text{imp}(t(i, n(f), u), f). \quad (82) \]

Let \( z \) be a natural number, let \( Z \) be a name for \( z \) in Vampire, and let \( \Delta(z) \) be as in Definition 11.

\[ d(Z) \text{ for } \Delta(z). \quad (83) \]

A Vampire notation for the existence of a formal deduction of \( f \) (with no assumption) in Robinson resolution systems is defined

\[ r(0, f) \text{ for } \vdash_{\alpha} f. \quad (84) \]

Using the notation in Definition (77),

\[ g(a(b, \text{imp}(t(i, n(f), u), f)), f, Z) \text{ for } \vdash B, T(i, \neg f, \emptyset, 1) \supset f \rightarrow f \text{ in } z. \quad (85) \]

Definitions (81)–(85) are used in Corollary 4.

Let \( Z \) be a positive integer. Corollary 4 in TPTP syntax is written

\[ \text{fof}\text{(cor4, axiom, }([Z] : (g(a(b, \text{imp}(t(i, n(f), u), f)), f, Z) \Rightarrow (d(Z) \& r(0, f) \leq d(Z))))). \quad (86) \]

Vampire uses induction on computing time \( z \) to prove Corollary 4, cf. Section 1.3.

Recall the sequence \( h(z) \), that is, a formal deduction of \( U_{(z-1)}0 \) in computing time from \( z \) to \( z - 1 \), cf. Definition 10. It is used in the induction proofs of Vampire.
Let $F \in \mathcal{F}$ be a propositional formula of propositional predicate calculus, see Definition 3. A formal deduction of $F_n$ as a sequence, cf. Definition (44), is written

$$\text{seq}(F_1, \ldots, F_n) \text{ for } < F_1, \ldots, F_n >.$$  \hfill (87)

### 5.3.1 Induction base

First a definition.

$$s_n \text{ for } n+1, n \in \mathbb{N}.$$  \hfill (88)

Let $f$ be any tautology on DNF. The conjecture of the induction base with computing time 2, see footnote 11,

$$fof(\text{cor}410, \text{conjecture}, (g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1) => (d(s1) \& (r(0, f) \leq d(s1))))).$$  \hfill (89)

Axiom 1, Definition 10, Definition 11, Definition (84), and Definition (87) give an auxiliary corollary,

$$fof(\text{cor}411, \text{axiom}, (g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1) => \text{seq}(h(s1), d(1)))).$$  \hfill (90)

Applying Definition 10, Definition 11, and Definitions (43)–(45), give an auxiliary corollary,

$$fof(\text{cor}412, \text{axiom}, (\text{seq}(h(s1), d(1))) \Rightarrow (d(s1) \& (r(0, f) \leq d(s1))))).$$  \hfill (91)

### 5.3.2 Proof of the induction base.

Vampire’s proof of conjecture (89) is reversed i.e. the contradiction comes first. **Proof.**

Refutation found.

$$fof(f20, \text{plain}, \langle \text{false} \rangle),$$  
-ingference(\text{global subsumption}, [], [f13, f8, f191])).

$$fof(f191, \text{plain}, (\sim \text{seq}(h(s1), d(1))),$$  
-ingference(\text{global subsumption}, [], [f11, f181])).

$$fof(f181, \text{plain}, (\sim d(s1) \mid \sim \text{seq}(h(s1), d(1))),$$  
-ingference(\text{global subsumption}, [], [f12, f10])).

$$fof(f10, \text{plain}, (\sim r(0, f) \mid d(s1)),$$  
-ingference(\text{cnf transformation}, [], [f5])).

$$fof(f5, \text{plain}, (g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1) \& (\sim d(s1) \mid (d(s1) \& \sim r(0, f)))$$  
-ingference(\text{cnf transformation}, [], [f2])).

$$fof(f2, \text{negated conjecture}, (\sim (g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1) => (d(s1) \& (d(s1) \Rightarrow r(0, f))))),$$  
-ingference(\text{negated conjecture}, [], [f1])).

22
Applying Definitions (43)–(45), Definition 10, and Definition 11 give an auxiliary corollary, using Definition 10, the induction hypothesis gives an auxiliary corollary, computing time

Let \( f \) be any tautology on DNF. The conjecture for computing time \( n + 1 \).

\[
\text{fof}(f1, \text{conjecture}, (g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1) => (d(s1) & (d(s1) => r(0, f))))),
\]

\[
\text{fof}(f12, \text{plain}, (r(0, f) | ~ d(s1) | ~ \text{seq}(h(s1), d(1))),
\]

\[
\text{inference(}\text{cnf,transformation}, [\pi], [f6])).
\]

\[
\text{fof}(f6, \text{plain}, (\sim \text{seq}(h(s1), d(1)) | (d(s1) & (\sim d(s1) | r(0, f)))),
\]

\[
\text{inference(}\text{cnf,transformation}, [\pi], [f4])).
\]

\[
\text{fof}(f4, \text{axiom}, (\text{seq}(h(s1), d(1)) => (d(s1) & (d(s1) => r(0, f)))),
\]

\[
\text{file('}/\text{tmp/}\text{SystemOnTPTPFormReply801}/\text{SOT}_O^{VG0UA'}, \text{cor}410)))).
\]

\[
\text{fof}(f11, \text{plain}, (d(s1) | ~ \text{seq}(h(s1), d(1))),
\]

\[
\text{inference(}\text{cnf,transformation}, [\pi], [f6])).
\]

\[
\text{fof}(f8, \text{plain}, (g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1)),
\]

\[
\text{inference(}\text{cnf,transformation}, [\pi], [f5])).
\]

\[
\text{fof}(f13, \text{plain}, (\sim g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1)) | ~ \text{seq}(h(s1), d(1)) | ~ g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1)),
\]

\[
\text{inference(}\text{cnf,transformation}, [\pi], [f7])).
\]

\[
\text{fof}(f7, \text{plain}, (\sim g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1)) | \text{seq}(h(s1), d(1))),
\]

\[
\text{inference(}\text{cnf,transformation}, [\pi], [f3])).
\]

\[
\text{fof}(f3, \text{axiom}, (g(a(b, \text{imp}(t(i, n(f), u), f)), f, s1) => \text{seq}(h(s1), d(1))),
\]

\[
\text{file('}/\text{tmp/}\text{SystemOnTPTPFormReply801}/\text{SOT}_O^{VG0UA'}, \text{cor}411)))).
\]

### 5.3.3 Induction step

Let \( f \) be any tautology on DNF. The conjecture for computing time \( n + 1 \).

\[
\text{fof}(\text{cor}421, \text{conjecture}, (g(a(b, \text{imp}(t(i, n(f), u), f)), f, sn) => (d(sn) & (r(0, f) <= d(sn))))).
\]

The induction hypothesis of computing time \( n \),

\[
\text{fof}(\text{cor}420, \text{axiom}, (g(a(b, \text{imp}(t(i, n(f), u), f)), f, n) => (d(n) & (r(0, f) <= d(n))))).
\]

The induction hypothesis gives an auxiliary corollary, using Definition 10,

\[
\text{fof}(\text{cor}422, \text{axiom}, (g(a(b, \text{imp}(t(i, n(f), u), f)), f, sn) => (\text{seq}(h(sn)) & g(a(b, \text{imp}(t(i, n(f), u), f)), f, n)))),
\]

Applying Definitions (43)–(45), Definition 10, and Definition 11 give an auxiliary corollary,

\[
\text{fof}(\text{cor}423, \text{axiom}, (\text{seq}(h(sn)) & d(n) & (r(0, f) <= d(n)) => (d(sn) & (r(0, f) <= d(sn))))).
\]
5.3.4 Proof of the induction step.

Vampire’s proof of conjecture (92) is reversed i.e. the contradiction comes first.

Proof.

Refutation found.

\[ \text{fof}(\text{f241}, \text{plain}, (\text{false}), \text{inference}(\text{global}, \text{assumption}, [], [\text{f19}, \text{f16}, \text{f11}, \text{f14}, \text{f17}, \text{f15}, \text{f13}])). \]

\[ \text{fof}(\text{f13}, \text{plain}, (\sim r(0,f) \mid \sim d(sn)), \text{inference}(\text{cnf}, \text{transformation}, [], [\text{f6}])). \]

\[ \text{fof}(\text{f6}, \text{plain}, (g(a(b, \text{impl}(t(i,n(f),u), f)), f, sn) \& (\sim d(sn) | (d(sn) \& \sim r(0,f)))), \text{inference}(\text{cnf}, \text{transformation}, [], [\text{f2}])). \]

\[ \text{fof}(\text{f2}, \text{negated}, \text{conjecture}, (~ (g(a(b, \text{impl}(t(i,n(f),u), f)), f, sn) => (d(sn) \& (d(sn) => r(0,f)))), \text{inference}(\text{negated}, \text{conjecture}, [], [\text{f1}])). \]

\[ \text{fof}(\text{f1}, \text{conjecture}, (g(a(b, \text{impl}(t(i,n(f),u), f)), f, sn) => (d(sn) \& (d(sn) => r(0,f)))), \text{file}(''/\text{tmp}/\text{SystemOnTPTPFormReply10657/SOT/myVJR', cor420)). \]

\[ \text{fof}(\text{f15}, \text{plain}, (r(0,f) \mid \sim d(n) | ~ g(a(b, \text{impl}(t(i,n(f),u), f)), f, n)), \text{inference}(\text{cnf}, \text{transformation}, [], [\text{f7}])). \]

\[ \text{fof}(\text{f7}, \text{plain}, (~ g(a(b, \text{impl}(t(i,n(f),u), f)), f, n) | (d(n) \& (\sim d(n) | r(0,f)))), \text{inference}(\text{cnf}, \text{transformation}, [], [\text{f3}])). \]

\[ \text{fof}(\text{f3}, \text{axiom}, (g(a(b, \text{impl}(t(i,n(f),u), f)), f, n) => (d(n) \& (d(n) => r(0,f)))), \text{file}(''/\text{tmp}/\text{SystemOnTPTPFormReply10657/SOT/myVJR', cor421)). \]

\[ \text{fof}(\text{f17}, \text{plain}, (g(a(b, \text{impl}(t(i,n(f),u), f)), f, n) | ~ g(a(b, \text{impl}(t(i,n(f),u), f)), f, sn)), \text{inference}(\text{cnf}, \text{transformation}, [], [\text{f8}])). \]

\[ \text{fof}(\text{f8}, \text{plain}, (~ g(a(b, \text{impl}(t(i,n(f),u), f)), f, sn) | (\text{seq}(h(sn)) \& g(a(b, \text{impl}(t(i,n(f),u), f)), f, n))), \text{inference}(\text{cnf}, \text{transformation}, [], [\text{f4}])). \]

\[ \text{fof}(\text{f4}, \text{axiom}, (g(a(b, \text{impl}(t(i,n(f),u), f)), f, sn) => (\text{seq}(h(sn)) \& g(a(b, \text{impl}(t(i,n(f),u), f)), f, n))), \text{file}(''/\text{tmp}/\text{SystemOnTPTPFormReply10657/SOT/myVJR', cor422)). \]

\[ \text{fof}(\text{f14}, \text{plain}, (d(n) | ~ g(a(b, \text{impl}(t(i,n(f),u), f)), f, n)), \text{inference}(\text{cnf}, \text{transformation}, [], [\text{f7}])). \]

\[ \text{fof}(\text{f11}, \text{plain}, (g(a(b, \text{impl}(t(i,n(f),u), f)), f, sn)), \text{inference}(\text{cnf}, \text{transformation}, [], [\text{f6}])). \]

\[ \text{fof}(\text{f16}, \text{plain}, (\text{seq}(h(sn)) |) \]
\[ g(a, b, \text{imp}(t(i, n(f), u), f), f, s_n) \]

\[ \text{inference(cnftransformation, \text{[]}., [f8]).} \]

\[ fof(f19, plain, (d(s_n) \sim r(0, f)) \sim d(n)) \]
\[ \sim \text{seq}(h(s_n)) \]

\[ \text{inference(cnftransformation, \text{[]}., [f10]).} \]

\[ faf(f10, plain, (\sim \text{seq}(h(s_n))) \sim d(n)) \]
\[ (d(n) \& \sim r(0, f)) \& (d(s_n) \& \sim (d(s_n) \& r(0, f)))) \]

\[ \text{inference(flattening, \text{[]}., [f9]).} \]

\[ fof(f9, plain, ((\sim \text{seq}(h(s_n))) \sim d(n) \& \sim r(0, f))) \| (d(s_n) \& (\sim d(s_n) \& r(0, f))) \]

\[ \text{inference(cnftransformation, \text{[]}., [f5]).} \]

\[ fof(f5, axiom, ((\text{seq}(h(s_n))) \& d(n) \& (d(n) \Rightarrow r(0, f))) \Rightarrow (d(s_n) \& (d(s_n) \Rightarrow r(0, f)))) \]

\[ \text{inference(ennftransformation, \text{[]}., [f5]).} \]

\[ fof(f5, axiom, ((\text{seq}(h(s_n))) \& d(n) \& (d(n) \Rightarrow r(0, f))) \Rightarrow (d(s_n) \& (d(s_n) \Rightarrow r(0, f)))) \]

\[ \text{file(’/tmp/SystemOnTPTPFormReply10657/SOT;myVJR’, cor423).} \]

### 5.4 Vampire’s proof of Corollary 5

A few definitions follow.

Let \( M \) be a natural number.

\[ \text{plus(mul(2, M), 1) for } 2M + 1. \]  

Let \( f \) be any tautology on DNF, and let \( n \) be some natural number. A Vampire notation for the big \( O \)-notation is defined.

\[ o(t(f, \text{plus(mul}(2, n), 1))) \text{ for } O(|f|^{2n+1}). \]  

Let \( m \) be any natural number, and let \( M \) be a name for \( m \) in Vampire.

\[ \text{size(d(t(f, M)), o(t(f, plus(mul}(2, M), 1)))} \text{ for } | \Delta(c \cdot |f|^m) | \in O(|f|^{2m+1}). \]  

Let \( f \) be any sufficiently large tautology on DNF. Using Definition (83), Definition (76), and Definitions (96)–(98) the conjecture of Corollary 5 in TPTP syntax becomes,

\[ fof(cor5, conjecture, (![M] : (d(t(f, M)) \Rightarrow \text{size(d(t(f, M)), o(t(f, plus(mul(2, M), 1))))}). \]  

Informally, the computing time \( c \cdot |f|^m \) and the size \( |f| \) of a tautology \( f \) give an upper bound on the size of the formal deduction \( \Delta(c \cdot |f|^m) \) (in Vampire syntax \( d(t(f, N)) \)), for a sufficiently large tautology \( f \) on DNF. Clearly, \( f \) should not be smaller than the Turing machine \( i \in \mathcal{U} \) that exists, from the assumption \( SAT \in \mathcal{P} \), cf. Section 1.3.
If in a proof \((\Delta(c \cdot |f|^m))\) of \(f\) the size of each formula is polynomial in the size of \(f\), the searching time (called sizesearch) is polynomial in the size of \(f\), and the computing time is polynomial in the size of \(f\) then the size of a formal deduction \(\Delta(c \cdot |f|^m)\) of \(f\) is polynomial in the size of \(f\) for a sufficiently large tautology \(f\) on DNF.

\[
fof(cor51, axiom, (![F, M] : ((sizeformula(o(t(f, M)))) &
sizecomptime(o(t(f, M)))) =>
size(d(t(f, M)), o(t(F, plus(mul(2, M), 1)))))),
\]

(100)

If there is a formal deduction \(\Delta(c \cdot |f|^m)\) in computing time \(c \cdot |f|^m\) then the upper bound on the size of each formula in \(\Delta(c \cdot |f|^m)\) is in \(O(|f|^m)\), a sufficiently large tautology \(f\) on DNF. Formally in TPTP syntax,

\[
fof(cor52, axiom, (![M] : (d(t(f, M)) => sizeformula(o(t(f, M)))))),
\]

(101)

If there is a formal deduction \(\Delta(c \cdot |f|^m)\) in computing time \(c \cdot |f|^m\) then the searching time (called sizesearch) of a quintuple is not greater than the size of \(|f|\) for a sufficiently large tautology \(f\) on DNF. Formally in TPTP syntax,

\[
fof(cor53, axiom, (![M] : (d(t(f, M)) => sizesearch(t(f, 1))))),
\]

(102)

If there is a formal deduction \(\Delta(c \cdot |f|^m)\) then the computing time (called size-comptime) is in \(O(|f|^m)\), for a sufficiently large tautology \(f\) on DNF. Formally in TPTP syntax,

\[
fof(cor54, axiom, (![M] : (d(t(f, M)) => sizecomptime(o(t(f, M)))))),
\]

(103)

5.4.1 Proof of Corollary 5

Vampire’s proof of conjecture (99) is reversed i.e. the contradiction comes first.

Proof.

Refutation found.

\[
fof(f28, plain, ($false),
inference(unit, esulting, esolution, [], [f23, f22, f16, f24])).
fof(f24, plain, ((![X0, X1] : (size(d(t(f, X1))), o(t(X0, plus(mul(2, X1), 1))))) |
~ sizecomptime(o(t(f, X1))) | ~ sizeformula(o(t(f, X1))))),
inference(subsumption, esolution, [], [f20, f21])).
fof(f21, plain, (sizesearch(t(f, 1))),
inference(unit, esulting, esolution, [], [f15, f17])).
fof(f17, plain, ((![X0] : (sizesearch(t(f, 1)) | ~ d(t(f, X0))))),
inference(cnft, ransformation, [], [f9])).
fof(f9, plain, ([!X0] : (~ d(t(f, X0)) | sizesearch(t(f, 1)))),
inference(ennf, ransformation, [], [f5])).
fof(f5, axiom, ([!X0] : (d(t(f, X0)) => sizesearch(t(f, 1)))),
file('/tmp/SystemOnTPTPFormReply29058/SOT_awDta\', cor53)).
\]

26
\[fof(f15, plain, (d(t(f, sK))).
\]
\[\text{inference}(\text{cnftransformation, []}, [f14]).
\]
\[fof(f14, plain, (d(t(f, sK))) &
\]
\[\sim \text{size}(d(t(f, sK)), o(t(f, plus(mul(2, sK), 1)))).
\]
\[\text{inference}(\text{skolemisation, [status(es)], [F8]}).
\]
\[fof(f8, plain, (?[X0] : (d(t(f, X0))) &
\]
\[\sim \text{size}(d(t(f, X0)), o(t(f, plus(mul(2, X0), 1)))).
\]
\[\text{inference}(\text{ennftransformation, []}, [f2])).
\]
\[fof(f2, negatedonjecture, (~[X0] : (d(t(f, X0)) =>
\]
\[size(d(t(f, X0)), o(t(f, plus(mul(2, X0), 1))))).
\]
\[\text{inference}(\text{negatedonjecture, []}, [f1])).
\]
\[fof(f1, conjecture, ([X0] : (d(t(f, X0)) =>
\]
\[size(d(t(f, X0)), o(t(f, plus(mul(2, X0), 1))))).
\]
\[\text{file}('\text{tmp/SystemOnTPTPFormReply29058/SOT}_1_{AzDtw}', \text{cor5})).
\]
\[fof(f20, plain, ([X0, X1] : (size(d(t(f, X1)), o(t(X0, plus(mul(2, X1), 1)))) |
\]
\[\sim \text{sizecomptime}(o(t(f, X1))) | \sim \text{sizecomptime}(t(f, 1)) | 
\]
\[\sim \text{sizeformula}(o(t(f, X1))))).
\]
\[\text{inference}(\text{enftransformation, []}, [f13]).
\]
\[fof(f13, plain, ([X0, X1] : ((\sim \text{sizeformula}(o(t(f, X1)))) |
\]
\[\sim \text{sizesearch}(t(f, 1)) | \sim \text{sizecomptime}(o(t(f, X1))) | 
\]
\[size(d(t(f, X1)), o(t(X0, plus(mul(2, X1), 1))))).
\]
\[\text{inference}(\text{flattening, []}, [f12]).
\]
\[fof(f12, plain, ([X0, X1] : ((\sim \text{sizeformula}(o(t(f, X1)))) |
\]
\[\sim \text{sizesearch}(t(f, 1)) | \sim \text{sizecomptime}(o(t(f, X1))) | 
\]
\[size(d(t(f, X1)), o(t(X0, plus(mul(2, X1), 1))))).
\]
\[\text{inference}(\text{ennftransformation, []}, [f7])).
\]
\[fof(f7, plain, ([X0, X1] : ((\text{sizeformula}(o(t(f, X1))) &
\]
\[\text{sizesearch}(t(f, 1)) & \text{sizecomptime}(o(t(f, X1)))) =>
\]
\[size(d(t(f, X1)), o(t(X0, plus(mul(2, X1), 1))))).
\]
\[\text{inference}(\text{rectify, []}, [f3])).
\]
\[fof(f3, axiom, ([X1, X0] : ((\text{sizeformula}(o(t(f, X0))) &
\]
\[\text{sizesearch}(t(f, 1)) & \text{sizecomptime}(o(t(f, X0)))) =>
\]
\[size(d(t(f, X0)), o(t(X1, plus(mul(2, X0), 1))))).
\]
\[\text{file}('\text{tmp/SystemOnTPTPFormReply29058/SOT}_1_{AzDtw}', \text{cor51})).
\]
\[fof(f16, plain, (\sim \text{size}(d(t(f, sK)), o(t(f, plus(mul(2, sK), 1))))).
\]
\[\text{inference}(\text{cnftransformation, []}, [f14]).
\]
\[fof(f22, plain, (\text{sizecomptime}(o(t(f, sK)))).
\]
\[\text{inference}(\text{unitresultingresolution, []}, [f15, f18]).
\]
\[fof(f18, plain, ([X0] : (\text{sizecomptime}(o(t(f, X0)))) | 
\]
\[\sim d(t(f, X0))).\text{inference}(\text{cnftransformation, []}, [f10]).
\]
\( \text{fof}(f10, \text{plain}, ([!X0] : (\sim d(t(f, X0)) | \text{sizecomptime}(o(t(f, X0))))),
\text{inference}(\text{ennftransformation}, [], [f6])). \)

\( \text{fof}(f6, \text{axiom}, ([!X0] : (d(t(f, X0)) => 
\text{sizecomptime}(o(t(f, X0)))))), 
\text{file}('/tmp/SystemOnTPTPFormReply29058/SOT_uAxDtw', cor54)). \)

\( \text{fof}(f23, \text{plain}, (\text{sizeformula}(o(t(f, sK)))), 
\text{inference}(\text{unitresultingresolution}, [], [f15, f19])). \)

\( \text{fof}(f19, \text{plain}, ([!X0] : (\text{sizeformula}(o(t(f, X0)) | 
\sim d(t(f, X0))))), 
\text{inference}(\text{ennftransformation}, [], [f11])). \)

\( \text{fof}(f11, \text{plain}, ([!X0] : (\sim d(t(f, X0)) | \text{sizeformula}(o(t(f, X0))))), 
\text{inference}(\text{ennftransformation}, [], [f4])). \)

\( \text{fof}(f4, \text{axiom}, ([!X0] : (d(t(f, X0)) => \text{sizeformula}(o(t(f, X0))))), 
\text{file}('/tmp/SystemOnTPTPFormReply29058/SOT_uAxDtw', cor52)). \)

### 5.5 Vampire’s proof of Lemma 1: third version

The first proof of Lemma 1, in Section 1.3, gives the concepts of the proof. Here follows Vampire’s proof in Hilbert’s proof theory.

Using the big O-notation of Definition (97), a polynomial upper bound on the size of a formal deduction of \( f \) in Robinson resolution systems is defined in TPTP syntax,

\[
\text{size}(r(0, f), o(t(f, \text{plus}(\text{mul}(2, n), 1)))) \text{ for } |\vdash_n f| \in O(|f|^{2n+1}). \tag{104}
\]

Let \( n \) be some natural number, and let \( f \) be any sufficiently large propositional tautology on disjunctive normal form (DNF). The conjecture of Lemma 1,

\( \text{fof}(\text{lemma1}, \text{conjecture}, \text{in}(\text{sat}, p) => 
\text{size}(r(0, f), o(t(f, \text{plus}(\text{mul}(2, n), 1))))). \)

Vampire’s proof of Lemma 1 uses Corollary 1, see Conjecture (68) in Section 5.1, Corollary 3, see Conjecture (78) in Section 5.2, and Corollary 4, see (86) in Section 5.3, and Corollary 5, see Conjecture (99) in Section 5.4.

Vampire also uses Definition 12.

The size of a Robinson resolution proof \( r(0, f) \) of \( f \) is polynomial in \( f \) if the size of \( d(t(f, M)) \) is polynomial in \( f \) and \( r(0, f) \) if \( d(t(f, M)) \) (a proof of \( f \)) for a sufficiently large tautology \( f \) on DNF.

\( \text{fof}(\text{def}12, \text{axiom}, ([!M, N] : (\text{size}(r(0, f), o(t(f, N))) <= 
(\text{size}(d(t(f, M)), o(t(f, N))) \& r(0, f) <= d(t(f, M)))))). \)

In addition, the proof of Lemma 1 also uses the following corollary.

Let \( f \) be any tautology.

If \( t(i, n(f), u) \) follows from \( b \) in computing time \( Z \) and \( t(i, n(f), u) => f \) then \( f \) follows from both \( b \) and \( t(i, n(f), u) => f \) in computing time \( Z \) in Kleene G4 systems, where \( Z \) is a positive integer.

\( \text{fof}(\text{cor}14, \text{axiom}, ([!Z] : ((g(a(b), t(i, n(f), u), Z) \& t(i, n(f), u) => f) 
=> g(a(b, \text{imp}(t(i, n(f), u), f)), f, Z))))). \)

28
5.5.1 Proof of Lemma 1

Vampire’s proof of conjecture (105) is reversed i.e. the contradiction comes first.

**Proof.**

Refutation found.

\[ \text{fof(f59, plain, (ffalse), } \]
\[ \text{inference(splitting, [], } [\text{f58, f30D, f41D, f37D, f26D, f38D, f36D, f45D}])]. \]
\[ \text{fof(f45, plain, (ffalse } | \text{ fsplit4), } \]
\[ \text{inference(subsumption, esolution, [], } [\text{f311, f44}]). \]
\[ \text{fof(f44, plain, (f!}[X0] : } \]
\[ \text{(~ g(a,b), t(i,n(f), u), X0)) | fsplit4), } \]
\[ \text{inference(subsumption, esolution, [], } [\text{f401, f42}]). \]
\[ \text{fof(f42, plain, (f!}[X0] : } \]
\[ \text{(~ g(a,b, imp(t(i,n(f), u), f)), f, X0)) | fsplit4), } \]
\[ \text{inference(subsumption, esolution, [], } [\text{f36, f25}]). \]
\[ \text{fof(f25, plain, (f!}[X0] : } \]
\[ \text{(~ g(a,b, imp(t(i,n(f), u), f)), f, X0) | d(X0))), } \]
\[ \text{inference(cnftransformation, [], } [\text{f15}]). \]
\[ \text{fof(f15, plain, (f!}[X0] : } \]
\[ \text{(~ g(a,b, imp(t(i,n(f), u), f)), f, X0) | (d(X0) & } \]
\[ \text{(~ d(X0) | r(0, f))))}, } \]
\[ \text{inference(enntransformation, [], } [\text{f5}]). \]
\[ \text{fof(f5, axiom, (f!}[X0] : } \]
\[ \text{(g(a,b, imp(t(i,n(f), u), f)), f, X0) => (d(X0) & (d(X0) => } \]
\[ \text{r(0, f))))}, } \]
\[ \text{file('}/tmp/SystemOnTPTPFormReply31569/SOTzDCdrl', cor4)).} \]
\[ \text{fof(f401, plain, (f!}[X0] : } \]
\[ \text{(g(a,b, imp(t(i,n(f), u), f)), f, X0) | ~ g(a,b, t(i,n(f), u), X0))), } \]
\[ \text{inference(global, subsumption, [], } [\text{f23, f27, f28}]). \]
\[ \text{fof(f28, plain, (f!}[X0] : } \]
\[ \text{(g(a,b, imp(t(i,n(f), u), f)), f, X0) | ~ f | ~ g(a,b, t(i,n(f), u), X0))), } \]
\[ \text{inference(cnftransformation, [], } [\text{f17}]).} \]
\[ \text{fof(f17, plain, (f!}[X0] : } \]
\[ \text{(~ g(a,b, t(i,n(f), u), X0) | } \]
\[ \text{(t(i,n(f), u) & ~ f) | g(a,b, imp(t(i,n(f), u), f)), f, X0))), } \]
\[ \text{inference(flattening, [], } [\text{f16}]).} \]
\[ \text{fof(f16, plain, (f!}[X0] : } \]
\[ \text{(~ g(a,b, t(i,n(f), u), X0) | } \]
\[ \text{(t(i,n(f), u) & ~ f) | g(a,b, imp(t(i,n(f), u), f)), f, X0))), } \]
\[ \text{inference(enntransformation, [], } [\text{f8}]).} \]
\[ \text{fof(f8, axiom, (f!}[X0] : } \]
\[ \text{(g(a,b, t(i,n(f), u), X0) & } \]
\[ \text{(t(i,n(f), u) => f)) => g(a,b, imp(t(i,n(f), u), f)), f, X0))), } \]
\[ \text{file('}/tmp/SystemOnTPTPFormReply31569/SOTzDCdrl', cor14)).} \]

29
fof(f27,plain,((!![X0] : (g(a(b, imp(t(i, n(f), u), f)), f, X0)) | t(i, n(f), u) | \sim g(a(b), t(i, n(f), u), X0)))).

fof(f23,plain, (f \mid \sim t(i, n(f), u)),

\text{inference}(\text{cnf transformation}, [], [f17])).

fof(f20,plain, (in(sat, p)),

\text{inference}(\text{cnf transformation}, [], [f11])).

fof(f11,plain, (in(sat, p) \&

\sim \text{size}(r(0, f), o(t(f, \text{plus}(\text{mul}(2, n), 1))))),

\text{inference}(\text{ennf transformation}, [], [f2])).

fof(f2, negated, onjecture, (\sim (in(sat, p) \Rightarrow

\text{size}(r(0, f), o(t(f, \text{plus}(\text{mul}(2, n), 1))))),

\text{inference}(\text{negated, onjecture}, [], [f1])).

fof(f1, conjecture, (in(sat, p) \Rightarrow

\text{size}(r(0, f), o(t(f, \text{plus}(\text{mul}(2, n), 1))))),

\text{file}(\text{/tmp/SystemOnTPTPFormReply31569/SOT''DCdr'', cor1})).

fof(f31,plain, (g(a(b), t(i, n(f), u), t(f, n))),

\text{inference}(\text{global, ubsumption}, [], [f22, f20])).

fof(f22, plain, (g(a(b), t(i, n(f), u), t(f, n))) | \sim \text{sat, p}),

\text{inference}(\text{cnf transformation}, [], [f12])).

fof(f12, plain, (\sim (in(sat, p)) \mid g(a(b), t(i, n(f), u), t(f, n))),

\text{inference}(\text{ennf transformation}, [], [f3])).

fof(f3, axiom, (in(sat, p) \Rightarrow g(a(b), t(i, n(f), u), t(f, n))),

\text{file}(\text{/tmp/SystemOnTPTPFormReply31569/SOT''DCdr'', lemma1})).

fof(f36, plain, (\text{[]} : ((!![X0] : (\sim d(X0) | \sim g(a(b, imp(t(i, n(f), u), f)), f, X0)) | \$sp(4)),

\text{inference}(\text{cnf transformation}, [], [f36D])).

fof(f36D, plain, ((!![X0] : (\sim d(X0) |

\sim g(a(b, imp(t(i, n(f), u), f)), f, X0)) \text{\Rightarrow} \$bdd\text{140444476}),

\text{introduced}(\text{splitting, onponent, ntrduction, []})).

fof(f38, plain, (r(0, f) | \$sp(6)),

\text{inference}(\text{cnf transformation}, [], [f38D])).

fof(f38D, plain, (r(0, f) \text{\Rightarrow} \$bdd\text{140444476}),

\text{introduced}(\text{splitting, onponent, ntrduction, []})).

fof(f26, plain, (\text{[]} : (r(0, f) | \sim d(X0) | \sim g(a(b, imp(t(i, n(f), u), f)), f, X0)))),
inference(cnfs transformation, [], [f15]).
fof(f37, plain, (~ r(0, f) $spl7), inference(cnfs transformation, [], [f37p])).
fof(f37p, plain, (~ r(0, f) <= $bdd140445124), introduced(splitting, component, introduction, [])].
fof(f41, plain, (\[X0, X1 : (~ size(d(t(f, X0)), o(t(f, X1)))) | size(r(0, f), o(t(f, X1)))) | $spl10),
inference(cnfs transformation, [], [f41p])).
fof(f41p, plain, (\[X0, X1 : (~ size(d(t(f, X0)), o(t(f, X1)))) | size(r(0, f), o(t(f, X1))))) <= $bdd140445124),
inference(cnfs transformation, [], [f41p])).
fof(f30, plain, (\[X0, X1 : (size(r(0, f), o(t(f, X1)))) | ~ r(0, f) | ~ size(d(t(f, X0)), o(t(f, X1))))),
inference(cnfs transformation, [], [f19])).
fof(f19, plain, (\[X0, X1 : (~ size(d(t(f, X0)), o(t(f, X1)))) | (d(t(f, X0)) & ~ r(0, f)) | size(r(0, f), o(t(f, X1)))))
inference(flattening, [], [f18])).
fof(f18, plain, (\[X0, X1 : ((~ size(d(t(f, X0)), o(t(f, X1)))) | (d(t(f, X0)) & ~ r(0, f)) | size(r(0, f), o(t(f, X1))))),
inference(cnfs transformation, [], [f10])).
fof(f10, plain, (\[X0, X1 : ((size(r(0, f), o(t(f, X1)))) & (d(t(f, X0)) => r(0, f)) => size(r(0, f), o(t(f, X1))))),
inference(rectify, [], [f7])).
fof(f7, axiom, (\[X1, X2 : ((size(d(t(f, X1)), o(t(f, X2)))) & (d(t(f, X1)) => r(0, f)) => size(r(0, f), o(t(f, X2)))))), file('/tmp/SystemOnTPTPFormReply31569/SOT_2DCdr1, def12)).
fof(f58, plain, ($false | $spl10),
inference(subsumption, esolution, [], [f56, f31i])).
fof(f56, plain, (~ g(a(b), t(i, n(f), u), t(f, n)) | $spl10),
inference(resolution, [], [f53, f40i])).
fof(f53, plain, (~ g(a(b, imp(t(i, n(f), u), f)), t(f, n)) | $spl10),
inference(unit, esolution, esolution, [], [f50, f25])).
fof(f50, plain, (~ d(t(f, n)) | $spl10),
inference(unit, esolution, esolution, [], [f46, f24])).
fof(f24, plain, (\[X0 : (size(d(t(f, X0)), o(t(f, plus(mul(2, X0), 1)))) | ~ d(t(f, X0)))), inference(cnfs transformation, [], [f14])).
fof(f14, plain, (\[X0 : (~ d(t(f, X0)) | size(d(t(f, X0)), o(t(f, plus(mul(2, X0), 1)))))),
inference(cnfs transformation, [], [f9])).
fof(f9, plain, (\[X0 :
The conjecture of Theorem 1 in TPTP syntax,
\[ \text{fof}(theorem1, conjecture, (\sim in(sat, p))). \]  \tag{108} 

In the proof of Theorem 1, Vampire uses Lemma 1, see Conjecture (105), and Corollary 6.

Let \( M \) be a natural number.

It is not the case that each sufficiently large tautology \( f \) has a formal deduction of polynomial size in the size of \( f \) on DNF in Robinson resolution systems.

\[ \text{fof}(cor6, axiom, (?[M] : size(r(0, f), o(t(f, M))))) \]  \tag{109} 

### 5.6.1 Proof of Theorem 1

Vampire's proof of conjecture (108) is reversed, that is, the contradiction comes first.

\text{Proof.} 

Refutation found.
\[ \text{fof}(f12, plain, ($false), \text{inference(subsumption, resolution, [], [f11, f10])}). \]
\[ \text{fof}(f10, plain, ([!!X0] : (\sim size(r(0, f), o(t(f, X0)))))), \text{inference(cnfttransformation, [], [f7])}). \]
\[ \text{fof}(f7, plain, ([!!X0] : size(r(0, f), o(t(f, X0)))), \text{inference(ennfttransformation, [], [f4])}). \]
\[ \text{fof}(f4, axiom, (\sim?[X0] : size(r(0, f), o(t(f, X0)))), \text{file('/')[-tmp/SystemOnTPTPFormReply10277/SOT_XoPTVB', cor7}). \]
\[ \text{fof}(f11, plain, (size(r(0, f), o(t(f, plus(mul(2, n), 1))))), \text{inference(global,subsumption, [], [f9, f8])}). \]
\[ \text{fof}(f8, plain, (in(sat, p)), \text{inference(cnfttransformation, [], [f5])}). \]
\[ \text{fof}(f5, plain, (in(sat, p)), \text{inference(cnfttransformation, [], [f5])}). \]
5.7 Vampire's proof of Theorem 2: third version

The conjecture of Theorem 2 in TPTP syntax,

\[ \text{fof}(\text{theorem2}, \text{conjecture}, (p \neq \text{np})). \tag{110} \]

Vampire uses Definition 13 and Theorem 1, see Conjecture (108).

Definition 13 in TPTP syntax,

\[ \text{fof}(\text{def13}, \text{axiom}, (p = \text{np} \implies \text{in}(\text{sat}, p))). \tag{111} \]

5.7.1 Proof of Theorem 2

Vampire's proof of conjecture (110) is reversed i.e. the contradiction comes first. 

**Proof.**

Refutation found.

\[ \text{fof}(f12, \text{plain}, ($false), \text{inference(subsumption,solution,} [], [f10, f11])). \]

\[ \text{fof}(f11, \text{plain}, ( \text{in}(\text{sat}, p)), \text{inference(subsumption,solution,} [], [f9, f8])). \]

\[ \text{fof}(f8, \text{plain}, (p = \text{np}), \text{inference(cnf transformation,} [], [f5])). \]

\[ \text{fof}(f5, \text{plain}, ((p = \text{np})), \text{inference(flattening,} [], [f2])). \]

\[ \text{fof}(f2, \text{negated,conjecture}, (~ p! = \text{np}), \text{inference(negated,conjecture,} [], [f1])). \]

\[ \text{fof}(f1, \text{conjecture}, (p! = \text{np})), \text{file('/tmp/SystemOnTPTPFormReply11853/SOT_8B3Un', theorem2))). \]

\[ \text{fof}(f9, \text{plain}, (\text{in}(\text{sat}, p) \mid p! = \text{np}), \text{file('/tmp/SystemOnTPTPFormReply10277/SOT_0aPTV_B', theorem1))). \]
inference(cnftransformation, [], [f7]).
fof(f7,plain,(p = np in(sat,p)).
inference(ennftransformation, [], [f4]).
fof(f4,axiom,(p = np => in(sat,p)).
file('/tmp/SystemOnTPTPFormReply11853/SOT68B3Un', def14)).
fof(f10,plain, (~ in(sat,p)).
inference(cnftransformation, [], [f6]).
fof(f6,plain,( ~ in(sat,p)).
inference(flattening, [], [f3]).
fof(f3,axiom,( ~ in(sat,p)).
file('/tmp/SystemOnTPTPFormReply11853/SOT68B3Un', theorem1)).

6 Conclusion

The first-order theory B has the single finite Axiom B that characterizes a
universal Turing [23] machine, and thus defines computing.

Vampire, cf. Riazanov and Voronkov [14], proves, in Hilbert’s [7, 8] proof
theory, the following results in a simply consistent conservative extension of
Theory B.

First, Vampire proves Corollary 1 and Corollaries 3–5, cf. Sections 5.1–5.4.
Vampire uses formalized information in Sections 2–4, e.g., Definition 11, cf.
Tärnlund [19, 20].

Second, Vampire proves, third version, Lemma 1 using Corollary 1 and
Corollaries 3–5, cf. Section 5.5.

Third, Vampire proves, third version, Theorem 1: SAT $\not\subset P$ from Lemma 1,
and Corollary 6 that follows directly from Haken’s theorem, cf. Section 5.6.

Fourth, Vampire proves, third version, Theorem 2: $P = \not\subset NP$ from Theo-
rem 1: SAT $\not\subset P$ and Definition 13: $P = \not\subset NP = SAT \subset P$, cf. the Cook-Levin
theorem [3, 11], cf. Section 5.7.

Fifth, Vampire thereby verifies, in Hilbert’s proof theory, the former results,
cf. Tärnlund [19, 20], that Theorem 1 and Theorem 2 have proofs in Theory B.

Acknowledgment

Frank Brown, Keith Clark, Hanna-Nina Ekelund, Niklas Ekelund, Andreas
Hamfelt, Torsten Palm, Bo Steinholtz, and Carl-Anton Tärnlund—thank you all.

References

[1] Keith L. Clark and Sten-Åke Tärnlund. A first order theory of data and
programs. In Bruce Gilchrist, editor, Information Processing 77, volume 7,

Communication Homme-machine en Français. Technical report, Groupe
Intelligence Artificielle, Université d’Aix-Marseille, Luminy, 1973.


35


[22] Sten-Åke Tärlund. The tautology problem is not in $\mathcal{NP}$. April 2013.
