A new viewpoint of the Gödel's incompleteness theorem and it's applications

Tianheng Tsui

May 9, 2018

Abstract

A new viewpoint of the Gödel's incompleteness theorem be given in this article which reveals the deep relationship between the logic and computation. Upon the results of these studies, an algorithm be given which shows how to search a proof of statement in first order logic from finite concrete examples, and an approach be proposed to improve searching mathematical proof by neural network.

1 Informal Introduction

Gödel's incompleteness theorem are the most famous result in modern logic. In addition to the Gödel's original proof, there are some other proofs of this theorem [12]. In this article, a new viewpoint to interpret the Gödel's incompleteness theorem will be given, which lead to some interesting applications and a deeper understanding of the relation between logic and computation.

The notion of "proof" plays a central role in mathematics as the means by which the truth or falsity of mathematical statements is established. The main difficulty in mathematical "proof" is how to prove "infinte objects" serve the given mathematical statement. If a mathematical statement talk about finite objects, for example, the statement: $\forall n \in N((0 < n < 100) \to (n^2 < 10000))$, we can test it one by one on the finite objects to prove or disprove the statement. That means there are only finite cases have to be tested for the statement. But if a mathematical statement talk about infinite objects, for example, the statement: $\forall n((n > 2) \to (n^2 > 2n))$, we cannot prove it or disprove it by test it one by one, but we have to classify all the infinite objects into essentially finite different cases, in each case there is a corresponding independent reason for the statement to be true or to be not true, thus we can prove or disprove the mathematical proposition about infinite objects. So we get the not rigorous but heuristic intuition observatons:

Observation 1.1. A true mathematical statement can be proved within finite steps in a consistent effective formal system if and only if, the domain of the

statement can be split into essentially finite different classes, in each class there is a corresponding independent reason to govern the members to serve the statement, i.e., there are only essentially finite different independent reasons to make the statement to be true.

Let a statement φ be true in natrual numbers and unprovable in the formal Peano system. So just within the formal Peano system, only we can do is to test it one by one on the standard natrual number $0, 1, 2, 3, \cdots$. if $\varphi(0)$ is true, we get a true sample, if $\varphi(1)$ is true, we get another true sample, ... The reason to make a natrual number m satisfys φ may be different and independent from other numbers'. There are infinite different reasons to make the statement φ to be true. Since essentially each number m has a particular reason to make the statement $\varphi(m)$ to be true and φ is not logical consequence of Peano Arithmetic, before it is tested: "computed" the $\varphi(m)$ and "abserved" the result in the Peano system, nobody can predict true or false of $\varphi(m)$ just within Peano system. Thus when we check the value of the true but unprovable statement φ on the natrual numbers, the results seem that you are tossing a coin and every time the result is up! Although it is not logically impossible, it's probablilty is 0 from prpbability theory. So we get the second intuition observatons:

Observation 1.2. Let $\mathcal{L}$ be a language, T is an $\mathcal{L} - theory$, $\mathcal{M}$ is a model of T. A mathematical statement φ is true in $\mathcal{M}$, expressible in $\mathcal{L}$, but it cannot be proved from T, if and only if, the domain of the statement in the model $\mathcal{M}$ cannot be split into essentially finite different classes, in each class there is a corresponding independent reason by T, which is expressible in $\mathcal{L}$, govern the members to serve the statement, i.e., there are essentially infinite different independent reasons govern the whole domain to serve the unprovable true statement, but any finite reasons do not.

The rigorous expression of the **observation** 1.2 is the **Theorem** 3.2.

2   Preliminaries and Notations

This section is devoted to the exposition of basic preliminary material, notations and conventions which be used throughout of this article.

The notion of algorithm can be defined in terms of Turing machines by means of the ChurchTuring thesis, so the sentence "There is an algorithm ..." means "There is a Turing machine to compute ...", and sometimes Turing machine algorithms be described in very high level. If a function or a map is recursive, it means that the function or the map can be computed by a Turing machine.

It is well known that there are character encoding system **ASCII** and language encoding system **LATEX 2$_\varepsilon$**. Therefore, throughout this article, we assume all the mathematical objects be encoded by these fixed encoding systems, and the length of a mathematical object is the number binary bits to represent the

object. For example, the symbols "t", "t" and "t_{298}" is represented as one character: "t", three characters: "t" and nine characters "t_{298}" in LaTeX 2_ε, each charater be encoded by seven bits in **ASCII**, therefore the binary length of these objects are 7, 21 and 63 respectively.

It should be noticed that the symbol "t_{298}" can be represented as "t_{298}" and "$ t_{298} $", we take the shortest representation to calculate its length.

Definition 2.1. Let s is a **ASCII** string, the ASCII length of s, written **asciilen**(s), abbreviated $\|s\|_{as}$, is the number of characters that it contains, and
$$\textbf{binlen}(s) = \textbf{asciilen}(s) \times 7$$
named binary length of s.

Let the formal Zermelo-Fraenkel axiomatic set theory is denoted by ZF, and ZFC denotes the theory ZF with the Axiom of Choice, and ω represents the natural number set N in the formal ZFC system.

Definition 2.2. A Turing machine **M** is a 5-tuple, $(Q, \Gamma, \delta, q_0, q_{halt})$
Q is a finite set of states, i.e., $\exists i (i \in \omega \wedge \|Q\| = i)$,
Γ is the tape alphabet containing the blank symbol $\sqcup$, and the left end symbol $\triangleright$,
$\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, S, R\}$ is the transition function,
if $\delta(q, \triangleright) = (p, s, b)$, then $(s = \triangleright) \wedge (b = R)$,
if $\delta(q, a) = (p, s, b)$ and $a \neq \triangleright$, then $s \neq \triangleright$,
$q_0 \in Q$ is the start state,
$q_{halt} \in Q$ is the halt state, that is $\forall a \in \Gamma$:
$\delta(q_{halt}, a) = (q_{halt}, a, S)$, and
$\forall q \in Q (\forall a \in \Gamma (\delta(q, a) = \delta(q_{halt}, a)) \rightarrow (q = q_{halt}))$.

Unless otherwise indicated, it will always be assumed that the tape alphabet $\Gamma = \{0, 1, \sqcup, \triangleright\}$ throughout this article, and we assume the basic notions and results of mathematical logic, such as formula, sentence, the set of all formulas is recursive ..., etc.

Definition 2.3. (**time complexity**) Let M be a Turing machine that halts on all inputs. The running time or time complexity of M, denoted by t_M, is the function
$$t_M : \ \omega \rightarrow \omega$$
where $t_M(n)$ is the maximum number of steps that M uses on any input of length n.

In computational complexity theory, a reasonable assumption $t_M(n) \geq n$ is to allow the algorithm have time to read its input. But in this section a property of the machine M with running time $t_M(n) < n$ be given, and the relationship between it and the provability of statement in consistent effective formal system will be revealed in later.

Theorem 2.1. Let M be a Turing machine, the length of input string s be denoted as $\|s\|$. If there exist a number K for any input s,

$$\|s\| \geq K \to t_M(\|s\|) < \|s\|$$

then $\forall r(\|r\| \geq K) \to (t_M(r) < K)$ and if $(\|s\| \leq K \to M(s) = 1)$ then we can prove $\forall s M(s) = 1$ in ZFC.

Proof. Let $\|s\| = K$, from the assumption $t_M(\|s\|) < \|s\|$, the machine M halts before it reads the last bit of the input s i.e., it never reach to the end boundary of the input, the bits following the $(K-1)$th bit have no effect on computation.

Therefore if input $\|r\| \geq K$, and the first $K-1$ bits are the same as a string s with $\|s\| = K$, the machine does not discriminate r from s, when computing on r, it return the same result as computing on s, and it halts after the same steps, i.e., $M(r) = M(s)$ and $t_M(\|r\|) = t_M(\|s\|) < \|s\| = K$.

The number of string s with $\|s\| \leq K$ is finite. From the explaination above and if $(\|s\| \leq K \to M(s) = 1)$, obviously we can prove $\forall s M(s) = 1$ in finite steps in ZFC.

$\square$

Definition 2.4. Let the set of all formulas is denoted by **Frm**, and let **Frmsq** denotes the set of all finite formula sequences, i.e., $sq \in$ **Frmsq** if and only if sq is a finite formula sequence:

$$sq = \langle s_0, s_1, \ldots, s_r \rangle, r \in \omega$$

For more rigorous, sq is a map from $r + 1$ to **Frm** such that

$$sq(i) \in \textbf{Frm}, \ \forall i < r + 1.$$

Definition 2.5. RTheory $= \{\langle T, al \rangle \mid T \subseteq \textbf{Frm}$ and al is an algorithm which decide whether $\varphi \in T$, for any formula φ i.e., T is recursive$\}$. If T is a finite set, we assume that T is a formula sequence: $\langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$ i.e., $T \in$ **Frmsq**.

Definition 2.6. The set Λ of logical axioms are arranged in seven groups:

1. Tautologies;

2. $\forall x \alpha \to \alpha_t^x$, where t is substitutable for x in α;

3. $\forall x(\alpha \to \beta) \to (\forall x \alpha \to \forall x \beta)$;

4. $\alpha \to \forall x \alpha$, where x does not occur free in α;

5. $x = x$;

6. $(x = y) \to (\alpha \to \beta)$, if α and β are atomic formulas and β is obtained from α by replacing an occurrence of x in α by y;

4

7. $\alpha_t^x \to \exists x \alpha$, where t is substitutable for x in α.

Definition 2.7. Let $\langle T, al \rangle \in$ **RTheory**, a proof π of a statement φ from T in ZFC is a finite sequence $\langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$ of formulas such that φ_n is φ and for each $i \le n$ one of the following conditions holds:

1. $\varphi_i \in \Lambda$;

2. $\varphi_i \in$ ZFC;

3. $\varphi_i \in T$;

4. $\exists j, k < i$ such that $\varphi_j = \varphi_k \to \varphi_i$;

5. $\exists j < i \exists k \in \omega (\varphi_i = \forall x_k \varphi_j)$.

and denoted as

$$\langle T, al \rangle \vdash^\pi \varphi$$

From the **definition** 2.4, $\pi(i) = \varphi_i$, and it is easy to see that there are algorithms decide the corresponding conditions such as:

1. $\mathbf{G}_\Lambda(\pi(i)) = 1$ if $\pi(i) \in \Lambda$, otherwise, 0.

2. $\mathbf{G}_{\text{ZFC}}(\pi(i)) = 1$ if $\pi(i) \in$ ZFC, otherwise, 0.

3. $\mathbf{G}_{\text{IN}}(\langle T, al \rangle, \pi(i)) = 1$ if $\pi(i) \in T$, otherwise, 0, note that $\mathbf{G}_{\text{IN}}$ use the algorithm al to decide whether $\pi(i) \in T$.

4. $\mathbf{G}_\to(\pi, i, j, k) = 1$ if $(j, k < i)$ and $\pi(j) = \pi(k) \to \pi(i)$, otherwise, 0.

5. $\mathbf{G}_\forall(\pi, i, j, k) = 1$ if $j < i$ and $k \in \omega(\pi(i) = \forall x_k \pi(j))$, otherwise, 0.

Let the set of above five verification algorithms is

$$\mathbf{G}_{\text{check}} = \{\mathbf{G}_\Lambda, \mathbf{G}_{\text{ZFC}}, \mathbf{G}_{\text{IN}}, \mathbf{G}_\to, \mathbf{G}_\forall\}$$

and its member is called checker.

Definition 2.8. (**proof type**) Let $\langle T, al \rangle \in$ **RTheory** and $\pi \in$ **Frmsq**, $\pi = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$, the proof type of $\langle \langle T, al \rangle, \pi \rangle$ denoted by

$$\textbf{prooftype}(\langle T, al \rangle, \pi)$$

such that: If **not** $\langle T, al \rangle \vdash^\pi \varphi_n$,

$$\textbf{prooftype}(\langle T, al \rangle, \pi) \text{ is the empty set.}$$

else if $\langle T, al \rangle \vdash^\pi \varphi_n$, then **prooftype**$(\langle T, al \rangle, \pi)$ is also called the proof type of $\langle T, al \rangle \vdash^\pi \varphi_n$, and it is a same length sequence $G = \langle g_0, g_1, g_2 \ldots, g_n \rangle$ of checkers, such that:

1. If $\varphi_i \in \Lambda$, then the corresponding g_i is a recursive function on **Frmsq** such that $\forall sq \in$ **Frmsq**:
$$g_i(sq) = \mathbf{G}_\Lambda(sq(i))$$
Say that g_i is a $\mathbf{G}_\Lambda$ type checker.

2. If $\varphi_i \in$ ZFC, then the corresponding g_i is a recursive function on **Frmsq** such that $\forall sq \in$ **Frmsq**:
$$g_i(sq) = \mathbf{G}_{\mathrm{ZFC}}(sq(i))$$
Say that g_i is a $\mathbf{G}_{\mathrm{ZFC}}$ type checker.

3. If $\varphi_i \in T$, then the corresponding g_i is a recursive function on **RTheory** $\times$ **Frmsq** such that $\forall sq \in$ **Frmsq** and $S = \langle ST, al_s \rangle \in$ **RTheory**:
$$g_i(S, sq) = \mathbf{G}_{\mathrm{IN}}(S, sq(i))$$
Say that g_i is a $\mathbf{G}_{\mathrm{IN}}$ type checker.

4. If $\exists j, k < i$ such that $\varphi_j = \varphi_k \to \varphi_i$, then the corresponding g_i is a recursive function on **Frmsq** such that $\forall sq \in$ **Frmsq**:
$$g_i(sq) = \mathbf{G}_\to(sq, i, j, k)$$
Say that g_i is a $\mathbf{G}_\to$ type checker.

5. If $\exists j < i$ and $k \in \omega(\varphi_i = \forall x_k \varphi_j)$, then the corresponding g_i is a recursive function on **Frmsq** such that $\forall sq \in$ **Frmsq**:
$$g_i(sq) = \mathbf{G}_\forall(sq, i, j, k)$$
Say that g_i is a $\mathbf{G}_\forall$ type checker.

G is also called the adjoint check sequence of π.

Definition 2.9. Let $\langle T_1, al_1 \rangle \vdash^{\pi_1} \varphi$, the adjoint check sequence of π_1 is G_1, $\langle T_2, al_2 \rangle \vdash^{\pi_2} \phi$, the adjoint check sequence of π_2 is G_2. Say that the proof type of $\langle T_1, al_1 \rangle \vdash^{\pi_1} \varphi$ is the same as the proof type of $\langle T_2, al_2 \rangle \vdash^{\pi_2} \phi$ if $G_1 = G_2$.

Definition 2.10. (adjoint checker) It is not hard to see that $G = \langle g_0, g_1, g_2, \ldots, g_n \rangle$, the proof type of $\langle T, al \rangle \vdash^\pi \varphi$, can be easily converted to an algorithm which decide whether a proof have the same type, denote the algorithm by $\mathbf{CK}_{(\langle T, al \rangle, \pi)}$, abbreviated $\mathbf{CK}_\pi$, and it is called the adjoint checker of $\langle T, al \rangle \vdash^\pi \varphi$, which on input
$$(\langle U, b \rangle, \sigma), \text{ where } \sigma = \{\phi_0, \phi_1, \phi_2, \ldots, \phi_m\}$$
it does:

firstly, it compare m to n, if $m \neq n$, return 0 and stop, else it does the following operations:

for all $0 \leq i \leq n$ it compute g_i such as:

1. if $g_i = \mathbf{G}_\Lambda(sq(i))$, then it compute $\mathbf{G}_\Lambda(\sigma(i))$;

2. if $g_i = \mathbf{G}_{\mathrm{ZFC}}(sq(i))$, then it compute $\mathbf{G}_{\mathrm{ZFC}}(\sigma(i))$;

3. if $g_i = \mathbf{G}_{\mathrm{IN}}(S, sq(i))$, then it compute $\mathbf{G}_{\mathrm{IN}}(\langle U, b\rangle, \sigma(i))$;

4. if $g_i = \mathbf{G}_\rightarrow(sq, i, j, k)$, then it compute $\mathbf{G}_\rightarrow(\sigma, i, j, k)$;

5. if $g_i = \mathbf{G}_\forall(sq, i, j, k)$, then it compute $\mathbf{G}_\forall(\sigma, i, j, k)$.

If all of the computations of g_i, $0 \leq i \leq n$, return 1, the $\mathbf{CK}_{(\langle T, al\rangle, \pi)}$ return 1 and stop, else return 0 and stop, therefore

$$\mathbf{CK}_{(\langle T, al\rangle, \pi)}(\langle U, b\rangle, \sigma) = \begin{cases} 1, & \text{if } \mathbf{prooftype}(\langle T, al\rangle, \pi) = \mathbf{prooftype}(\langle U, b\rangle, \sigma) \\ 0, & \text{otherwise} \end{cases}$$

Indeed, the algorithm $\mathbf{CK}_{(\langle T, al\rangle, \pi)}$ is described by a group of checkers g_i, it is only depend on the proof sequence π, it is therefore abbreviated to $\mathbf{CK}_\pi$.

Theorem 2.2. $\mathbf{CK}_{(\langle T, al\rangle, \pi)}((\langle T, al\rangle, \pi) = 1$, i.e.,

$$\text{If } \langle T, al\rangle \vdash^\pi \varphi \text{ then } \mathbf{CK}_\pi((\langle T, al\rangle, \pi) = 1$$

Proof. It is obvious from the definition 2.10. $\qquad\qquad\square$

3   A theorem of provability and an algorithm of proof

In order to prove a statement, we may enumerate formula sequences, and verify the sequences, one by one, whether or not it is a proof sequence of the statement. But it is not a practical method. In practice, mathematicians often have computed lots of concrete examples before proposing a conjecture by intuition, and searching a proof of it guided by intuition. In this section, the prove process will be studied from the computational viewpoint, and give a rigorous expression of the following statement: "There are essentially infinite different independent reasons govern the whole domain to serve the unprovable true statement"(**Theorem** 3.2), and give an algorithm which explain some aspects of practical prove activities.

Definition 3.1. Let M is a Turing machine: $(Q, \Gamma, \delta, q_0, q_{halt})$,
    t is a computation tape square, or simply tape square, if $t \in \Gamma \times Q \times \{0, 1\}$,
    $T_M = \{t | t = (t_0, t_1, t_2, \cdots), \forall i \in \omega \ t_i \in \Gamma \times Q \times \{0, 1\}\}$,
    $Table_M = \{table | table = (tape_0, tape_1, tape_2, \cdots), \text{where } \forall i \in \omega(tape_i \in T_M)\}$.
    π_M is a projection function from $\Gamma \times Q \times \{0, 1\}$ to Γ as: $\pi_M(s, p, v) = s$.

Remark 3.1. A table of the Turing machine M can be describe as following figure.

t_{00}	t_{01}	t_{02}	t_{03}	$\cdots$
t_{10}	t_{11}	t_{12}	t_{13}	$\cdots$
t_{20}	t_{21}	t_{22}	t_{23}	$\cdots$
t_{30}	t_{31}	t_{32}	t_{33}	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

The rows are tape configurations of the Turing machine M which satisfy some conditions.

In ZFC system, a tape configuration of the Turing machine M:

$$t = (t_0, t_1, t_2, \cdots), (\forall i \in \omega \ t_i \in \Gamma \times Q \times \{0, 1\})$$

can be defined as function from ω to $\Gamma \times Q \times \{0, 1\}$, and a table can be defined as function tb from ω to T_M satisfys extra conditions which will be shown later.

Therefore the $t_{i,j}$ as above figure can be represented as $t(i)(j)$ in ZFC formal system. For convenience later, let "$t_{i,j}$" is the abbreviation of "$t(i)(j)$" i.e., t is a function from ω to T_M, $t(i)$ is a function from ω to $\Gamma \times Q \times \{0, 1\}$ and $t_{i,j} = t(i)(j) \in \Gamma \times Q \times \{0, 1\}$, a computation tape square.

If $t_{i,j} = t(i)(j) \in \Gamma \times Q \times \{1\}$, we say the machine M is reading the jth square of the tape at step i.

Definition 3.2. It is easy to see that the relation $t \in Table_M$ and π_M can be defined within finite formulas in ZFC. We say that t is a table of M computing on input s, if $t \in Table_M$ and satisfys extra conditions such as:

- $\forall i(t_{i,0} \in \{\triangleright\} \times Q \times \{0, 1\})$, this means the leftmost end of a tape is always markered by $\triangleright$.

- $\forall i \exists k(t_{i,k} \in \Gamma \times Q \times \{1\}) \wedge (\forall j(j \neq k) \rightarrow (t_{i,j} \in \Gamma \times Q \times \{0\}))$, this formula means that there is one and only one square be reading at any time by the machine.

and some interpretations of the transition function $\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, S, R\}$ such as:

- if $\delta(q, a) = (p, b, L)$, then the corresponding formula is
  $\forall i \forall j(t_{i,j} = (a, q, 1) \rightarrow (t_{(i+1),j} = (b, p, 0)) \wedge (t_{(i+1),(j-1)} = (\pi_M(t_{i,(j-1)}), p, 1)) \wedge (\forall k(k \neq j \wedge k \neq j - 1) \rightarrow t_{(i+1),k} = (\pi_M(t_{i,k}), p, 0)))$.

8

- if $\delta(q,a) = (p,b,S)$, then the corresponding formula is

  $\forall i \forall j (t_{i,j} = (a,q,1) \rightarrow (t_{(i+1),j} = (b,p,1)) \wedge (\forall k(k \neq j) \rightarrow t_{(i+1),k} = (\pi_M(t_{i,k}), p, 0)))$.

- if $\delta(q,a) = (p,b,R)$, then the corresponding formula is

  $\forall i \forall j (t_{i,j} = (a,q,1) \rightarrow (t_{(i+1),j} = (b,p,0)) \wedge (t_{(i+1),(j+1)} = (\pi_M(t_{i,(j+1)}), p, 1)) \wedge (\forall k(k \neq j \wedge k \neq j+1) \rightarrow t_{(i+1),k} = (\pi_M(t_{i,k}), p, 0)))$.

- ...

- ...

- ...

- $\exists m(\forall j (j < m) \rightarrow t_{0,j} \neq (\sqcup, q_0, 0)) \wedge (\forall j(j \geq m) \rightarrow t_{0,j} = (\sqcup, q_0, 0))$,

Let the set of these formulas are arranged as formula sequence: $\langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_k \rangle$, and denoted by $\mathbf{def}_M$.

If $s = s_0 s_1 s_2 \ldots s_l$, $s_i \in \{0,1\}, 0 \leq i \leq l$ then there are $l+3$ formulas to describe the input s on M such as:

- φ_{k+1}: $\forall j (j > l+1) \rightarrow t_{0j} = (\sqcup, q_0, 0)$,

- φ_{k+2}: $t_{0,0} = (\triangleright, q_0, 1)$,

- φ_{k+3}: $t_{0,1} = (s_0, q_0, 0)$,

- φ_{k+4}: $t_{0,2} = (s_1, q_0, 0)$,

- φ_{k+5}: $t_{0,3} = (s_2, q_0, 0)$,

- $\ldots$,

- φ_{k+l+3}: $t_{0,(l+1)} = (s_l, q_0, 0)$.

Let the set of these $l+3$ formulas be denoted by $\mathbf{input}_s$.

It is not hard to see that $\mathbf{def}_M$ is unchanged if the machine M is fixed, and if $\|s\| = l$ then $\mathbf{input}_s$ have $l+3$ formulas.

$\mathbf{T}_{\langle M,s \rangle}$ is a formula sequence such that

$$\mathbf{T}_{\langle M,s \rangle} = \langle \mathbf{def}_M \cup \mathbf{input}_s \rangle = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_{k+l+3} \rangle$$

where we arrange its members as: the first $k+1$ formulas belonging to $\mathbf{def}_M$, the following $l+3$ formulas belonging to $\mathbf{input}_s$, and keep the order as above described, i.e.,

- φ_0,

- φ_1,

9

- φ_2,

- $\ldots$,

- φ_k is $\exists m (\forall j (j < m) \rightarrow t_{0,j} \neq (\sqcup, q_0, 0)) \wedge (\forall j (j \geq m) \rightarrow t_{0,j} = (\sqcup, q_0, 0))$,

- φ_{k+1} is $\forall j (j > l+1) \rightarrow t_{0,j} = (\sqcup, q_0, 0)$,

- φ_{k+2} is $t_{0,0} = (\triangleright, q_0, 1)$,

- φ_{k+3} is $t_{0,1} = (s_0, q_0, 0)$,

- φ_{k+4} is $t_{0,2} = (s_1, q_0, 0)$,

- φ_{k+5} is $t_{0,3} = (s_2, q_0, 0)$,

- $\ldots$,

- φ_{k+l+3} is $t_{0,(l+1)} = (s_l, q_0, 0)$.

Thus $\mathbf{T}_{\langle M,s \rangle}$ is a $k+l+4$ formulas sequence as above.

Thus we can define an algorithm $\mathbf{AL}$ whose input is $\langle \varphi, \mathbf{T}_{\langle M,s \rangle} \rangle$ and decide whether $\varphi \in \mathbf{T}_{\langle M,s \rangle}$, such as:

Definition 3.3. ($\mathbf{AL}$) The algorithm on input $\langle \varphi, \mathbf{T}_{\langle M,s \rangle} \rangle$, it compare φ to each φ_i in $\mathbf{T}_{\langle M,s \rangle}$

- it compare φ to φ_0, if they match, return yes and halt, else:

- it compare φ to φ_1, if they match, return yes and halt, else:

- it compare φ to φ_2, if they match, return yes and halt, else:

- $\ldots$

- it compare φ to φ_{k+l+3}, if they match, return yes and halt, else return no and halt.

denoted such algorithm by $\mathbf{AL}$ throughout this article.

Definition 3.4. (**normal proof**) Let M is a Turing machine $(Q, \Gamma, \delta, q_0, q_{halt})$, $v \in \{0,1\}, s \in \{0,1\}^*$, and $\mathbf{AL}$ is the algorithm as the **definition** 3.3. A normal proof of $M(s) = v$ in ZFC is a formula sequence $\pi = \{\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_r\}$ such that:

$$\varphi_r \text{ is the formula: } \exists i t_{i,1} = (v, q_{halt}, 1) \text{ and } \langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle \vdash^{\pi} \varphi_r$$

The ASCII length of the normal proof be denoted by $\|\pi\|_{as}$, defined as

$$\|\pi\|_{as} = \sum_{i=0}^{r} \mathbf{asciilen}(\varphi_i)$$

Definition 3.5. (**string order**) Let x and y are two ASCII strings, we say that x precede y, written $x <_s y$, if $\|x\|_{as} < \|y\|_{as}$, or $\|x\|_{as} = \|y\|_{as}$ and x precede y in dictionary order.

Definition 3.6. Let x and y are two ASCII strings, and $\|x\|_{as} = n$, $\|y\|_{as} = m$, the concatenation of x and y, written $x \circ y$, is the string obtained by appending y to the end of x, i.e., $x \circ y = x_1 \cdots x_n y_1 \cdots y_m$.

Definition 3.7. Let S_1 and S_2 are two finite formula sequences, such that:

$$S_1 = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$$

$$S_2 = \langle \sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_m \rangle$$

the concatenation of S_1 and S_2, written $S_1 + S_2$, is the finite formula sequence obtained by appending S_2 to the end of S_1:

$$S_1 + S_2 = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n, \sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_m \rangle$$

In general, $S_1 + S_2 \neq S_2 + S_1$, and it is obvious that the operation "+" satisfy associative law. So if $S_0, S_1, S_2, \ldots, S_n$ are all finite formula sequences, we can define

$$S_0 + S_1 + S_2 + \ldots + S_n = (\ldots((S_0 + S_1) + S_2) + \ldots) + S_n$$

denoted by

$$\sum_{i=0}^{n} S_i$$

Definition 3.8. (**sequence order**) Let S_1 and S_2 are two finite formula sequences, such that:

$$S_1 = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$$

$$S_2 = \langle \sigma_0, \sigma_1, \sigma_2, \ldots, \sigma_m \rangle$$

and we take each formula as an ASCII string, say that S_1 precede S_2, written $S_1 <_s S_2$, if and only if

$$\varphi_0 \circ \varphi_1 \circ \varphi_2 \ldots, \circ \varphi_n <_s \sigma_0 \circ \sigma_1 \circ \sigma_2 \ldots \circ \sigma_m$$

Remark 3.2. Let π_1 and π_2 are two normal form proofs of $M(s) = v$, if $\|\pi_1\|_{as} < \|\pi_2\|_{as}$, we say π_1 is shorter than π_2. Indeed, if $M(s) = v$, the process of M computing on s can be easily converted to a normal proof π such that $\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle \vdash^{\pi} \exists i \ (t_{i,1} = (v, q_{halt}, 1))$. Let its ASCII length is $\|\pi\|_{as} = n$. Since the normal proofs of $M(s) = v$ with ASCII length shorter than n are finite, we can enumerate formula sequences in sequence order, and check whether it is a normal proof of $M(s) = v$. Hence it is easy to see that there is an algorithm, for any $M(s) = v$, it give the minimum of the set $S = \{k | \ k = \|\pi\|_{as}, \ \pi$ is a proof of $M(s) = v$ in ZFC $\}$.

Definition 3.9. (**FS**) Fix an algorithm which can find the shortest ASCII length of normal proofs as in the **remark** 3.2 , throughout this article denote it by $\mathbf{FS}(M, s, v)$, abbreviated $\mathbf{FS}(M, s)$, if the machine M halts on all inputs, i.e., for any $M(s) = v$,

$\mathbf{FS}(M, s, v) = \mathbf{min}\{k|\ k = \|\pi\|_{as},\ \langle\mathbf{T}_{\langle M,s \rangle}, \mathbf{AL}\rangle \vdash^{\pi} \exists i\ (t_{i,1} = (v, q_{halt}, 1))\}$

$\mathbf{FS}(M, s)=$ "On input $\langle M, s \rangle$, an encoding of a machine M and a string s:

1. Using the description of M and s, compute M on s and get result v.

2. Enumerate formula sequence in the sequence order, every time that $\mathbf{FS}(M, s)$ outputs a sequence S, verify whether it is satisfies

$$\langle\mathbf{T}_{\langle M,s \rangle}, \mathbf{AL}\rangle \vdash^{S} \exists i\ (t_{i,1} = (v, q_{halt}, 1))$$

3. Let the first formula sequence π satisfies

$$\langle\mathbf{T}_{\langle M,s \rangle}, \mathbf{AL}\rangle \vdash^{\pi} \exists i\ (t_{i,1} = (v, q_{halt}, 1))$$

then return the result $\|\pi\|_{as}$"

Definition 3.10. (**adjoin proof complexity**) Let M be a Turing machine that halts on all inputs. The adjoint proof complexity of M is the function $f : \omega \to \omega$, where $f(n)$ is the maximum number of the set: $\{\mathbf{FS}(M, r)|\ \|r\| = n\}$, denote such function by $\mathbf{apf}_M$, i.e., if $\|s\| = n$ then

$$\mathbf{apf}_M(\|s\|) = \mathbf{apf}_M(n) = f(n) = \mathbf{max}\{\mathbf{FS}(M, r)|\ \|r\| = \|s\|\}$$

Let M be a Turing machine which compute a function

$$g : \{0, 1\}^* \to \{0, 1\}$$

and $M(s) = v$, $\|s\| = n$. t is the table of M computing on input s and the time complexity of M is $f(n)$, then the process of M computing on input s can be converted to a special normal proof of $M(s) = v$ as the following.

Definition 3.11. Let M be a Turing machine which compute a function

$$g : \{0, 1\}^* \to \{0, 1\}$$

and $M(s) = v$, $\|s\| = n$, "t" is the table of M computing on input s, the time complexity of M is $f(n)$. It is obvious that there are only finite tape squares be affected by the computation, exactly not exceed $(f(n) + 1) \times (f(n) + 1)$ tape squares.

The content of each tape square is determined by certain squares in the preceding row. If we know the values at $t_{(i-1),(j-1)}, t_{(i-1),(j)}$, and $t_{(i-1),(j+1)}$, we can obtain the value at $t_{i,j}$ with M's transition function. For example:

Let $100 < f(n) + 1$, and if we have proved the formula:

$$t_{99,100} = (0, q, 1)$$

and a transition rule is:

$$\forall i \forall j (t_{i,j} = (0, q, 1) \rightarrow$$
$$(t_{(i+1),j} = (1, p, 0))$$
$$\wedge (t_{(i+1),(j+1)} = (\pi_M(t_{i,(j+1)}), p, 1))$$
$$\wedge (\forall k(k \neq j \wedge k \neq j+1) \rightarrow t_{(i+1),k} = (\pi_M(t_{i,k}), p, 0))) \tag{1}$$

Then we can prove the formula: $t_{100,100} = (1, p, 0)$ from the above two formulas, let γ denotes the formula (1), π denotes a formula sequence, a special normal proof of $M(s) = v$, the section of proving $t_{100,100} = (1, p, 0)$ as following:

$\pi(n_1)$: $t_{99,100} = (0, q, 1)$,
     (previously proved)

  . . .

  . . .

  . . .

$\pi(c)$: γ,
     ($\mathbf{G}_{\mathrm{IN}}(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi(c)) = 1$, i.e., $\pi(c) \in \mathbf{T}_{\langle M,s \rangle}$, indeed $\pi(c) \in \mathbf{def}_M$)

$\pi(c+1)$: $\gamma \rightarrow (\forall i \forall j t_{i,j} = (0, q, 1) \rightarrow (t_{(i+1),j} = (1, p, 0)))$,
     ($\mathbf{G}_\Lambda(\pi(c+1)) = 1$, i.e., $\pi(c+1) \in \Lambda$, indeed $\pi(c+1)$ is a tautology)

$\pi(c+2)$: $\forall i \forall j t_{i,j} = (0, q, 1) \rightarrow (t_{(i+1),j} = (1, p, 0))$,
     ($\mathbf{G}_\rightarrow(\pi, c+2, c+1, c) = 1$, i.e., $\pi(c+2)$ is obtained by modus ponens from $\pi(c+1)$ and $\pi(c)$)

$\pi(c+3)$: $(\forall i \forall j t_{i,j} = (0, q, 1) \rightarrow (t_{(i+1),j} = (1, p, 0))) \rightarrow$
     $(t_{99,100} = (0, q, 1) \rightarrow (t_{100,100} = (1, p, 0)))$,
     ($\mathbf{G}_\Lambda(\pi(c+3)) = 1$, i.e., $\pi(c+3) \in \Lambda$)

$\pi(c+4)$: $t_{99,100} = (0, q, 1) \rightarrow (t_{100,100} = (1, p, 0))$,
     ($\mathbf{G}_\rightarrow(\pi, c+4, c+3, c+2) = 1$, i.e., $\pi(c+4)$ is obtained by modus ponens from $\pi(c+3)$ and $\pi(c+2)$)

$\pi(c+5)$: $t_{100,100} = (1, p, 0)$
     ($\mathbf{G}_\rightarrow(\pi, c+5, c+4, n_1) = 1$, i.e., $\pi(c+5)$ is obtained by modus ponens from $\pi(c+4)$ and $\pi(n_1)$)

The six formulas from $\pi(c)$ to $\pi(c+5)$ form a proof section of $t_{100,100} = (1, p, 0)$, denoted by $\mathbf{SEC}_\pi(t_{100,100} = (1, p, 0))$.

    $t_{100,100}$ is represented by \$t_{100,100}\$ and $t_{99,100}$ is represented by \$t_{99,100}\$ in LaTeX 2_ε, and it is obvious that $\|99\|_{as} < \|100\|_{as} < 100 < f(\|s\|) + 1 = f(n) + 1$
therefore:

$$\|t_{99,100}\|_{as} = \|\$t\_\{99,100\}\$\|_{as} < 2f(n) + 9$$

13

$$\|t_{100,100}\|_{as} = \|\$t\_\{100,100\}\$\|_{as} < 2f(n) + 9$$

Because $\mathbf{def}_M$ is a sequence formulas $\langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_k \rangle$ as in the **definition** 3.2, we can define

$$\|\mathbf{def}_M\|_{as} = \sum_{i=0}^{k} \|\varphi_i\|_{as}$$

and it is easy to see

1. $\pi(c) \in \mathbf{def}_M$, so $\|\pi(c)\|_{as} \leq \|\mathbf{def}_M\|_{as}$

2. $\|\pi(c+1)\|_{as} < 2\|\pi(c)\|_{as} \leq 2\|\mathbf{def}_M\|_{as}$

3. $\|\pi(c+2)\|_{as} < \|\pi(c)\|_{as} \leq \|\mathbf{def}_M\|_{as}$

4. $\|\pi(c+3)\|_{as} < 2\|\pi(c+2)\|_{as} + 4f(n) + 20 < 2\|\mathbf{def}_M\|_{as} + 4f(n) + 20$

5. $\|\pi(c+4)\|_{as} < \|\pi(c+3)\|_{as} < 2\|\mathbf{def}_M\|_{as} + 4f(n) + 20$

6. $\|\pi(c+5)\|_{as} < \|\pi(c+4)\|_{as} < 2\|\mathbf{def}_M\|_{as} + 4f(n) + 20$

$$\sum_{i=0}^{5} \|\pi(c+i)\|_{as} < 12f(n) + 10\|\mathbf{def}_M\|_{as} + 60$$

Using the same approach as proving $t_{100,100} = (1, p, 0)$ above, we can prove a formula $t_{a,b} = v_{ab}$ for each pair $\langle a, b \rangle$, $0 \leq a, b \leq f(n) + 1$, denoted by $\mathbf{SEC}_\pi(t_{a,b} = v_{ab})$, is called the proof section of $t_{a,b} = v_{ab}$ where the v_{ab} is the value of the tape square $t_{a,b}$ on the table of $M(s) = v$.

The idea behind this approach is simple, the proof formula sequence is just a description of M computing on input s: the tape configuration t_i determined by the preceding tape configuration t_{i-1} and an appropriate transition rule of the machine M. Note that

1. $\mathbf{SEC}_\pi(t_{0,0} = (\triangleright, q_0, 1))$ is just the formula itself, because from the **definition** 3.2

$$(t_{0,0} = (\triangleright, q_0, 1)) \in \mathbf{T}_{\langle M, s \rangle}$$

2. The same reasoning applies to any proof section of

$$t_{0,b} = (s_b, q_0, 0), \quad 0 < b \leq n + 1$$

i.e., $\mathbf{SEC}_\pi(t_{0,b} = (s_b, q_0, 0))$ is just one formula, itself.

3. From the **definition** 3.2, any $b > n+1$, $\mathbf{SEC}_\pi(t_{0,b} = (\sqcup, q_0, 0))$ is following formula sequence:

$$b > n + 1$$
$$\forall j (j > n + 1) \to t_{0j} = (\sqcup, q_0, 0)$$
$$(\forall j (j > n + 1) \to t_{0j} = (\sqcup, q_0, 0)) \to ((b > n + 1) \to t_{0,b} = (\sqcup, q_0, 0))$$
$$(b > n + 1) \to t_{0,b} = (\sqcup, q_0, 0)$$
$$t_{0,b} = (\sqcup, q_0, 0)$$

4. For any $a > 0$, the $\mathbf{SEC}_\pi(t_{a,b} = v_{ab})$ like the case $\mathbf{SEC}_\pi(t_{100,100} = (1,p,0))$ shown above, is a description of how the content of the tape square $t_{a,b}$ be determined by certain squares in the preceding row.

Hence it is easy to see that there exist two numbers K and C, independent of the input s, for all

$$0 \le a, b \le f(\|s\|) + 1 = f(n) + 1, \quad \|\mathbf{SEC}_\pi(t_{a,b} = v_{ab})\|_{as} < Kf(n) + C$$

Since t is the table of $M(s) = v$, there exists a number $d \le f(n) + 1$ satisfys $t_{d,1} = (v, q_{halt}, 1)$, and we can prove the formula $t_{d,1} = (v, q_{halt}, 1)$ like in the described situation $t_{99,100} = (0, q, 1)$ above. Then

$$(t_{d,1} = (v, q_{halt}, 1)) \to \exists i t_{i,1} = (v, q_{halt}, 1), \quad \text{(denoted by } \varphi_{r-1})$$

$$\exists i t_{i,1} = (v, q_{halt}, 1), \quad \text{(denoted by } \varphi_r)$$

is the proof of $\exists i t_{i,1} = (v, q_{halt}, 1)$ from $t_{d,1} = (v, q_{halt}, 1)$. Obviously,

$$\|\varphi_{r-1}\|_{as} + \|\varphi_r\|_{as} < Kf(n) + C$$

Thus there is a special normal proof of $M(s) = v$ in ZFC, such that:

1.

$$\pi = \{\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_r\} \quad = (\sum_{i=0}^{f(n)+1} \sum_{j=0}^{f(n)+1} \mathbf{SEC}_\pi(t_{i,j} = v_{ij})) + \langle \varphi_{r-1} \rangle + \langle \varphi_r \rangle$$

Note that the operation "+" and "$\sum$" on formula sequences are defined in **definition** 3.7.

2. φ_{r-1} is the formula: $(t_{d,1} = (v, q_{halt}, 1)) \to \exists i t_{i,1} = (v, q_{halt}, 1)$.

3. φ_r is the formula: $\exists i t_{i,1} = (v, q_{halt}, 1)$.

4. $\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle \vdash^\pi \varphi_r$

We denote this special normal proof of $M(s) = v$ as $\Pi_{\langle M,s \rangle}$. Therefore,

$$\|\Pi_{\langle M,s \rangle}\|_{as} = \sum_{i=0}^{f(\|s\|)+1} \sum_{j=0}^{f(\|s\|)+1} \|\mathbf{SEC}_\pi(t_{i,j} = v_{ij})\|_{as} + \|\varphi_{r-1}\|_{as} + \|\varphi_r\|_{as}$$

$$< [\sum_{i=0}^{f(\|s\|)+1} \sum_{j=0}^{f(\|s\|)+1} (Kf(\|s\|) + C)] + Kf(\|s\|) + C$$

$$= [(f(\|s\|) + 2)(f(\|s\|) + 2) + 1](Kf(\|s\|) + C)$$

Where the two numbers K and C are independent of the input s.

It is obvious that

$$\mathbf{FS}(M, s) \le \|\Pi_{\langle M, s \rangle}\|_{as} < [(f(\|s\|) + 2)(f(\|s\|) + 2) + 1](Kf(\|s\|) + C) \quad (2)$$

Therefore we get the following lemma:

Lemma 3.1. (**polynomial proof complexity**)
Let M be a polynomial time Turing machine. then its adjoint proof complexity is also a polynomial, i.e., $\mathbf{apf}_M$ is bounded by a polynomial.

Proof. Let the time complexity of M is a polynomial $f(n)$. From the **definition** 3.10, $\mathbf{apf}_M(\|s\|) = \mathbf{max}\{\mathbf{FS}(M, r)| \ \|r\| = \|s\|\}$. since the above inequality (2), we get

$$\mathbf{apf}_M(\|s\|) < [(f(\|s\|) + 2)(f(\|s\|) + 2) + 1](Kf(\|s\|) + C)$$

$\square$

Lemma 3.2. (**bounded running time**)
Let M is a Turing machine $(Q, \Gamma, \delta, q_0, q_{halt})$, $v \in \{0, 1\}, r \in \{0, 1\}^*$, and **AL** is the algorithm as the **definition** 3.3, $M(r) = v$, π is a formula sequence:

$$\langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$$

such that:

$$\varphi_n \text{ is the formula: } \exists it_{i,1} = (v, q_{halt}, 1) \text{ and } \langle \mathbf{T}_{\langle M, r \rangle}, \mathbf{AL} \rangle \vdash^\pi \varphi_n$$

That is π is a normal proof of $M(r) = v$, thus we can define a Turing machine on $s \in \{0, 1\}^*$ as:
$$\mathbf{f}(s) = \mathbf{CK}_\pi(\langle \mathbf{T}_{\langle M, s \rangle}, \mathbf{AL} \rangle, \pi)$$

then the time complexity of $\mathbf{f}$: $t_{\mathbf{f}}(n)$ is bounded, i.e., there exists a number K, for all $s \in \{0, 1\}^*$, $t_{\mathbf{f}}(\|s\|) < K$.

Proof. From the **definition** 2.10 $\mathbf{CK}_\pi$, the adjoint checker of π, be described by a group of checkers:
$$\{g_0, g_1, g_2, \ldots, g_n\}$$

therefore $f(s) = 1$ if and only if the formula sequence π is a normal proof of $M(s) = 1$. Indeed there are only five types of checkers:

1. $\mathbf{G}_\Lambda$ type;

2. $\mathbf{G}_{\mathrm{ZFC}}$ type;

3. $\mathbf{G}_{\mathrm{IN}}$ type;

4. $\mathbf{G}_\to$ type;

5. $\mathbf{G}_\forall$ type.

16

Since the formula sequence π is fixed

$$\pi = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$$

and $\langle \mathbf{T}_{\langle M,r \rangle}, \mathbf{AL} \rangle \vdash^\pi \varphi_n$, therefore from the **theorem** 2.2:

$$f(r) = \mathbf{CK}_\pi(\langle \mathbf{T}_{\langle M,r \rangle}, \mathbf{AL} \rangle, \pi) = 1$$

So only the $\mathbf{G}_{\mathrm{IN}}$ type checkers need to be computed, because:

1. if $g_i = \mathbf{G}_\Lambda(sq(i))$, from the **definition** 2.8, $\pi(i)$, i.e., φ_i must be in Λ, therefore $\mathbf{G}_\Lambda(\pi(i)) = 1$, $\mathbf{CK}_\pi$ need not to compute the checker $g_i = \mathbf{G}_\Lambda(sq(i))$ on input $(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi)$;

2. if $g_i = \mathbf{G}_{\mathrm{ZFC}}(sq(i))$, from the **definition** 2.8, $\pi(i)$, i.e., φ_i must be in ZFC, therefore $\mathbf{G}_{\mathrm{ZFC}}(\pi(i)) = 1$, $\mathbf{CK}_\pi$ need not to compute the checker $g_i = \mathbf{G}_{\mathrm{ZFC}}(sq(i))$ on input $(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi)$;

3. if $g_i = \mathbf{G}_\to(sq, i, j, k)$, from the **definition** 2.8, $\pi(i)$, i.e., φ_i must satisfys the following condition $j, k < i$, $\varphi_j = \varphi_k \to \varphi_i$ therefore $\mathbf{G}_\to(\pi, i, j, k) = 1$, $\mathbf{CK}_\pi$ need not to compute the checker $g_i = \mathbf{G}_\to(sq, i, j, k)$ on input $(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi)$;

4. if $g_i = \mathbf{G}_\forall(sq, i, j, k)$, from the **definition** 2.8, $\pi(i)$, i.e., φ_i must satisfys the following condition $j < i$, $k \in \omega$, $(\varphi_i = \forall x_k \varphi_j)$, therefore $\mathbf{G}_\forall(\pi, i, j, k) = 1$, $\mathbf{CK}_\pi$ need not to compute the checker $g_i = \mathbf{G}_\forall(sq, i, j, k)$ on input $(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi)$;

Thus the value of $\mathbf{CK}_\pi(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi)$, i.e., $\mathbf{f}(s)$ depends only on $\mathbf{G}_{\mathrm{IN}}$ type checkers.

Let $g_i = \mathbf{G}_{\mathrm{IN}}(S, sq(i))$, then from the **definition** 2.10, $\mathbf{CK}_\pi$ compute

$$\mathbf{G}_{\mathrm{IN}}(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi(i))$$

and from the **definition** 2.7, $\mathbf{G}_{\mathrm{IN}}$ use the algorithm $\mathbf{AL}$ to decide whether $\pi(i) \in \mathbf{T}_{\langle M,s \rangle}$. From the **definition** 3.2,

$$\mathbf{T}_{\langle M,r \rangle} = \langle \mathbf{def}_M \cup \mathbf{input}_r \rangle$$

$$\mathbf{T}_{\langle M,s \rangle} = \langle \mathbf{def}_M \cup \mathbf{input}_s \rangle$$

Let $\mathbf{def}_M$ are k formulas, and $r = r_0 r_1 r_2 \ldots r_l$, $r_i \in \{0,1\}, 0 \le i \le l$, therefore $\mathbf{T}_{\langle M,r \rangle}$ are $k + l + 4$ formulas as in **definition** 3.2.

Since $\langle \mathbf{T}_{\langle M,r \rangle}, \mathbf{AL} \rangle \vdash^\pi \varphi_n$ and g_i is a $\mathbf{G}_{\mathrm{IN}}$ type checker, from the **definition** 2.8, $\pi(i) \in \mathbf{T}_{\langle M,r \rangle}$.

Therefore, for each $s \in \{0,1\}^*$, when $\mathbf{G}_{\mathrm{IN}}$ use the algorithm $\mathbf{AL}$ to decide whether $\pi(i) \in \mathbf{T}_{\langle M,s \rangle}$, only the first $k + l + 4$ formulas of the $\mathbf{T}_{\langle M,s \rangle}$ need to be tested. That is the number of steps in compute a $\mathbf{G}_{\mathrm{IN}}$ type checker is less than a fixed number, denoted by C, and the number of the all $\mathbf{G}_{\mathrm{IN}}$ type checkers less than $n + 1$, compute all the all $\mathbf{G}_{\mathrm{IN}}$ type checkers less than $C \times (n+1)$ steps.

Hence there exists a number K, for all $s \in \{0,1\}^*$, the number of steps of compute $\mathbf{f}(s) = \mathbf{CK}_\pi(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi)$ is less than K, i.e., $\forall s \; s \in \{0,1\}^* \; t_{\mathbf{f}}(\|s\|) < K$. $\qquad \square$

In order to analyze proof procedure in more detail, we now consider the input of normal proof. Let M be a Turing machine on $\{0,1\}^*$, $r = r_0 r_1 r_2 \ldots r_l$, $r_i \in \{0,1\}, 0 \leq i \leq l$, $M(r) = 1$, and π is a normal proof of $M(r) = 1$, from the **definition** 3.4, we know

$$\langle \mathbf{T}_{\langle M,r \rangle}, \mathbf{AL} \rangle \vdash^\pi \exists i t_{i,1} = (1, q_{halt}, 1)$$

According to the **definition** 3.2 the $\langle \mathbf{T}_{\langle M,r \rangle}$ is:

- φ_0,

- φ_1,

- φ_2,

- $\ldots$,

- φ_k is $\exists m (\forall j (j < m) \to t_{0,j} \neq (\sqcup, q_0, 0)) \land (\forall j (j \geq m) \to t_{0,j} = (\sqcup, q_0, 0))$,

- φ_{k+1} is $\forall j (j > l+1) \to t_{0,j} = (\sqcup, q_0, 0)$,

- φ_{k+2} is $t_{0,0} = (\triangleright, q_0, 1)$,

- φ_{k+3} is $t_{0,1} = (r_0, q_0, 0)$,

- φ_{k+4} is $t_{0,2} = (r_1, q_0, 0)$,

- φ_{k+5} is $t_{0,3} = (r_2, q_0, 0)$,

- $\ldots$,

- φ_{k+l+3} is $t_{0,(l+1)} = (r_l, q_0, 0)$.

It is easy to see that for any $s \in \{0,1\}^*$, the first $k+1$ formulas of $\mathbf{T}_{\langle M,s \rangle}$ are the same formulas, i.e., the formula sequence $\mathbf{def}_M$. If $s \neq r$ the different formulas between $\mathbf{T}_{\langle M,s \rangle}$ and $\mathbf{T}_{\langle M,r \rangle}$ are all in

$$\mathbf{input}_s \cup \mathbf{input}_r$$

Therefore we have the following definition:

Definition 3.12. Let $\pi = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$ is a normal proof of $M(r) = 1$ as described above, the key information set of π is the formula set, denoted by $\mathbf{keyset}(\pi)$:

$$\{ \varphi | (\varphi \in \pi) \land (\varphi \in \mathbf{input}_r) \}$$

and the key information of π, is the formula obtained by connecting all the formulas of $\mathbf{keyset}(\pi)$ by $\land$ operations, and denoted by $\mathbf{keyinfo}(\pi)$:

$$\bigwedge_{\varphi \in \mathbf{keyset}(\pi)} \varphi$$

Corollary 3.1. Let $\pi = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$ is a normal proof of $M(r) = 1$, then $\mathbf{CK}_\pi(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi) = 1$ if and only if the input $s \in \{0,1\}^*$ satisfies $\mathbf{keyinfo}(\pi)$, i.e.,

$$\forall s((\mathbf{CK}_\pi(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi) = 1) \leftrightarrow \mathbf{keyinfo}(\pi))$$

Proof. From the proof of **lemma** 3.2, we know that in order to decide whether or not $\mathbf{CK}_\pi(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi) = 1$, only the $\mathbf{G}_{\mathrm{IN}}$ type checkers need to be computed, and from the above discussion, indeed, only the checkers corresponding to the formulas in $\mathbf{keyset}(\pi)$ need to be computed, therefore the lemma is proved. $\square$

Corollary 3.2. Let $\pi = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$ is a normal proof of $M(r) = 1$, if an input $s \in \{0,1\}^*$ satisfies $\mathbf{keyinfo}(\pi)$ then $M(s) = 1$ can be proved in ZFC, i.e., the following formula can be proved in ZFC:

$$\forall s(\mathbf{keyinfo}(\pi) \to (M(s) = 1))$$

Proof. It is obviously true from the above discussion. $\square$

Theorem 3.1. Let $\pi = \langle \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n \rangle$ is a normal proof of $M(r) = 1$, then the following formula can be proved in ZFC:

$$\forall s((\mathbf{CK}_\pi(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi) = 1) \to (M(s) = 1))$$

Proof. This theorem is obviously deduced from the **corollary** 3.1 and **corollary** 3.2. $\square$

Definition 3.13. Let M is a Turing machine $(Q, \Gamma, \delta, q_0, q_{halt})$, $v_i \in \{0,1\}$, $r_i \in \{0,1\}^*$, $i = 0, 1, \cdots, n.$, and $\mathbf{AL}$ is the algorithm as the **definition** 3.3, $M(r_i) = v_i$, π_i is a formula sequence:

$$\langle \varphi_{i_0}, \varphi_{i_1}, \varphi_{i_2}, \ldots, \varphi_{i_{k_i}} \rangle$$

such that:

$\varphi_{i_{k_i}}$ is the formula: $\exists j t_{j,1} = (v_i, q_{halt}, 1)$ and $\langle \mathbf{T}_{\langle M,r_i \rangle}, \mathbf{AL} \rangle \vdash^{\pi_i} \varphi_{i_{k_i}}$

That is π_i is a normal proof of $M(r_i) = v_i$, the corresponding adjoint checker of π_i is $\mathbf{CK}_{\pi_i}$, and let

$$C = \{\mathbf{CK}_{\pi_0}, \mathbf{CK}_{\pi_2}, \cdots, \mathbf{CK}_{\pi_n}\}$$

thus we can define a Turing machine $\mathbf{F}_C$ on $s \in \{0,1\}^*$ as:
$\mathbf{F}_C = $ "On input s, where $s \in \{0,1\}^*$:

1. for each i, $0 \le i \le n$ use $\mathbf{CK}_{\pi_i}$ to compute

$$\mathbf{CK}_{\pi_i}(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi_i)$$

2. If there is a checker return 1, i.e., there is a normal proof π_k, $\mathbf{CK}_{\pi_k}(\langle \mathbf{T}_{\langle M,s\rangle}, \mathbf{AL}\rangle, \pi_k) = 1$, the machine $\mathbf{F}_C$ return 1, and halts.

3. If all the computations of checkes return 0, i.e.,

$$\mathbf{CK}_{\pi_i}(\langle \mathbf{T}_{\langle M,s\rangle}, \mathbf{AL}\rangle, \pi_i) = 0, \text{ for all } 0 \leq i \leq n$$

the machine $\mathbf{F}_C$ return 0, and halts."

We call $\mathbf{F}_C$ the C generated verifier.

Corollary 3.3. Let the Turing machine $\mathbf{F}_C$ is the C generated verifier as the **definition** 3.13, then the time complexity of $\mathbf{F}_C$: $t_{\mathbf{F}_C}(n)$ is bounded, i.e., there exists a number K, for all $s \in \{0,1\}^*$, $t_{\mathbf{F}_C}(\|s\|) < K$.

Proof. From the **lemma** 3.2, we know that the time complexity of each

$$\mathbf{CK}_{\pi_i}(\langle \mathbf{T}_{\langle M,s\rangle}, \mathbf{AL}\rangle, \pi_i)$$

is bounded by a number K_i, hence the computation steps of $\mathbf{F}_C$ is no more than $\sum_{i=0}^{n} K_i + M$ where M is a large enough constant number. $\square$

Lemma 3.3. Let the Turing machine $\mathbf{F}_C$ is the C generated verifier as the **definition** 3.13, then the following formula can be proved in ZFC:

$$(\mathbf{F}_C(s) = 1) \leftrightarrow \left(\bigvee_{\mathbf{CK}_{\pi_i} \in C} (\mathbf{CK}_{\pi_i}(\langle \mathbf{T}_{\langle M,s\rangle}, \mathbf{AL}\rangle, \pi_i) = 1) \right)$$

Proof. From the definition of $\mathbf{F}_C$, it is obviously true. $\square$

Lemma 3.4. There exists a Turing machine M that it halts on every input and the following five formulas can be proved in ZFC:

1. $\forall r, s \in \{0,1\}^* \ (\ \|s\| = \|r\| \to (M(s) = M(r)) \ )$.

2. $\forall r, s \in \{0,1\}^* \ (\ \|s\| > \|r\|) \to (M(s) = 1 \to M(r) = 1)$.

3. $\forall r, s \in \{0,1\}^* \ (\ \|s\| > \|r\|) \to (M(r) = 0 \to M(s) = 0)$.

4. $\forall s \in \{0,1\}^* \ (\ M(s) = 0 \vee M(s) = 1)$.

5. $(\forall s(s \in \{0,1\}^*) \to (M(s) = 1)) \vee (\exists m(\|s\| < m \to M(s) = 1) \wedge (\|s\| \geq m \to M(s) = 0))$.

but the formula $\forall s(M(s) = 1)$ is independent of ZFC, i.e., it cannot be proved in ZFC and its negation is also unprovable in ZFC.

Proof. This lemma is just the **Corollary 3.2** in the paper [11]. $\square$

Theorem 3.2. Let M as in the **lemma** 3.4, i.e., $\forall s(M(s) = 1)$ is independent of ZFC, then $\mathbf{FS}(M, s, 1)$ is unbounded on $s \in \{0,1\}^*$, that is

$$\forall m \exists s \in \{0,1\}^* \ (\mathbf{FS}(M, s, 1) > m)$$

20

PROOF IDEA   From the **lemma** 3.4, $\forall s(M(s) = 1)$ is independent of ZFC. Therefore we cannot find a string $s \in \{0,1\}^*$ satisfying $M(s) = 0$, that is, for all $s \in \{0,1\}^*$, we use M to compute on s will returning 1, but we cannot prove $\forall s(M(s) = 1)$ in ZFC.

If $\mathbf{FS}(M, s, 1)$ is bounded on $s \in \{0,1\}^*$, then there exists a number n for all $s \in \{0,1\}^*$,

$$\mathbf{FS}(M, s, 1) < n$$

Let

$$S_n = \{\pi|\ \pi \text{ is normal proof of } M(s) = 1, \quad s \in \{0,1\}^*,\ \|\pi\|_{as} < n, \}$$

It is not hard to see that S_n is finite.

Since $\mathbf{FS}(M, s, 1) < n$, for each $s \in \{0,1\}^*$, there exists a normal proof sequence π_s of $M(s) = 1$ and $\|\pi_s\|_{as} < n$,

$$\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle \vdash^{\pi_s} (\exists it_{i,1} = (1, q_{halt}, 1))$$

therefore $\pi_s \in S_n$. Since S_n is finite, thus we can prove

$$\forall s(M(s) = 1)$$

in finite steps in ZFC, contradiction.

Proof. Let M as in the **lemma** 3.4, if we found a string $s \in \{0,1\}^*$, $M(s) = 0$, then it is obvious that we can prove $\neg(\forall s(M(s) = 1))$ in ZFC. But from the **lemma** 3.4:

$$\text{The statement } \forall s(M(s) = 1) \text{ is independent of ZFC.} \tag{3}$$

Therefore we cannot find such string, i.e., for all $s \in \{0,1\}^*$, we use M to compute on s will returning 1, but we cannot prove $\forall s(M(s) = 1)$ in ZFC.

Now we prove the statement $\forall m \exists s \in \{0,1\}^*\ (\mathbf{FS}(M, s, 1) > m)$ by contradiction.

First, we assume for the purpose of later obtaining a contradiction that $\mathbf{FS}(M, s, 1)$ is bounded on $s \in \{0,1\}^*$. Thus there exists a number n for all $s \in \{0,1\}^*$,

$$\mathbf{FS}(M, s, 1) < n. \tag{4}$$

Let ϕ is the formula $\exists it_{i,1} = (1, q_{halt}, 1)$ then define the S_n as:

$$S_n = \{\pi|\ \langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle \vdash^{\pi} \phi,\ s \in \{0,1\}^*,\ \|\pi\|_{as} < n, \}$$

It is not hard to see that S_n is finite, therefore let

$$S_n = \{\pi_0, \pi_1, \pi_2, \ldots, \pi_m\}$$

and the corresponding adjoint checkers set $C_n = \{c_\pi | \ c_\pi = \mathbf{CK}_\pi, \ \pi \in S_n\}$ is also finite, i.e.,

$$C_n = \{\mathbf{CK}_{\pi_0}, \mathbf{CK}_{\pi_1}, \mathbf{CK}_{\pi_2}, \dots, \mathbf{CK}_{\pi_m}\}$$

Let $\mathbf{F}_{C_n}$ is the C_n generated verifier(see the definition in **definition** 3.13).

From (4), for each $s \in \{0,1\}^*$ $\mathbf{FS}(M, s, 1) < n$, therefore there exists a normal proof sequence π_s, $\|\pi_s\|_{as} < n$,

$$\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle \vdash^{\pi_s} \phi$$

Therefore $\pi_s \in S_n$ and the corresponding adjoint checker $\mathbf{CK}_{\pi_s} \in C_n$ and from the **theorem** 2.2, we get

$$\mathbf{CK}_{\pi_s}(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi_s) = 1.$$

So when we compute the machine $\mathbf{F}_{C_n}$ on any $s \in \{0,1\}^*$, it will return 1.

Because the **corollary** 3.3, the computation steps on $\mathbf{F}_{C_n}$ is bounded, so there exists a number K, the computation complexity of $\mathbf{F}_{C_n}$ is bounded by K:

$$\forall s \in \{0,1\}^* \ t_{\mathbf{F}_{C_n}}(\|s\|) < K$$

and obviously, there are finite strings in $\{s| \ \|s\| \leq K\}$, so we can prove the following formula in ZFC:

$$\|s\| \leq K \to \mathbf{F}_{C_n}(s) = 1$$

Therefore we can prove the following two statements:

$$\forall s \in \{0,1\}^* \ \|s\| \geq K \to t_{\mathbf{F}_{C_n}}(\|s\|) < \|s\|$$

and

$$\|s\| \leq K \to \mathbf{F}_{C_n}(s) = 1$$

Since the **theorem** 2.1, we can prove $\forall s \mathbf{F}_{C_n}(s) = 1$ in ZFC, and from the **lemma** 3.3, we can prove

$$\forall s \left(\bigvee_{i=0}^{m} (\mathbf{CK}_{\pi_i}(\langle \mathbf{T}_{\langle M,s \rangle}, \mathbf{AL} \rangle, \pi_i) = 1) \right)$$

and since the **theorem** 3.1 we can prove $\forall s(M(s) = 1)$ in ZFC, contradicting the statemenet of (3): $\forall s(M(s) = 1)$ is independent of ZFC.

$\square$

Indeed, the **theorem** 3.2 is actually the rigorous expression of "there are essentially infinite different independent reasons govern the whole domain to serve the unprovable true statement". From this theorem we get the following corollary:

Corollary 3.4. The formula $\forall s(M(s) = 1)$ is provable in ZFC, if and only if $\mathbf{FS}(M, s, 1)$ is bounded on $s \in \{0, 1\}^*$, that is

$$\exists m \forall s \in \{0, 1\}^* \left(\mathbf{FS}(M, s, 1) < m\right)$$

Proof. This statement is obviously true from the **theorem** 3.2. $\qquad\square$

Indeed, the proof of **theorem** 3.2 has shown a procedure how to search a proof of a general conclusion(for example $\forall s \in \omega(M(s) = 1)$), from some finite concrete examples. That is, the procedure is to find a adjoint checkers set C_n which is large enough to satisfies the following two statements:

$$\forall s \in \{0, 1\}^* \; \|s\| \geq K \to t_{\mathbf{F}_{C_n}}(\|s\|) < \|s\|$$

and

$$\|s\| \leq K \to \mathbf{F}_{C_n}(s) = 1$$

Now we give this procedure as an explicit algorithm which will halts if and only if the formula $\forall s M(s) = 1$ is not independant of ZFC:

Algorithm

1. Begin with the checkers set $C = $ empty set,

2. Since the **corollary** 3.3, we can find a number, K, such that $t_{\mathbf{F}_C}(\|s\|) < K$ on any input s,

3. Compute $M(s)$ and $\mathbf{F}_C(s)$ on all $\|s\| \leq K$,

4. If $\forall s(\|s\| \leq K) \to M(s) = 1$ and $\|s\| \leq K \to \mathbf{F}_C(s) = 1$, then from the proof of **theorem** 3.2, $\forall s(M(s) = 1)$ can be proved in ZFC, halts,

5. Else if there exist $\|s\| \leq K$ and $M(s) \neq 1$, then we can prove

$$\neg \forall s(M(s) = 1)$$

   halts,

6. Else if $\forall s(\|s\| \leq K) \to M(s) = 1$, but $\exists r(\|r\| \leq K) \wedge (\mathbf{F}_C(r) = 0)$, then adding an adjoint checker $\mathbf{CK}_\pi$ to the checkers set C, where π is a shortest normal proof of $M(r) = 1$ and $\mathbf{CK}_\pi$ is the adjoint checker of π,

7. goto 2

In practice, we can improve this algorithm by using neural network technique at the step 6 to searching the shortest normal proof. The most interesting thing we will see in a later article is, that the algorithm seemingly to imply some sophisticated processes, such as training neural network, cannot be proved being effective in formal system, though it is practically effective.

References

[1] T. P. Baker, J. Gill, and R. Solovay, *Relativizations of the P=?NP question*, SIAM Journal on Computing 4(4):431-442, 1975.

[2] J. Hartmanis, *Feasible Computations and Provable Complexity Problems*, SIAM, 1978.

[3] J. Hartmanis and J. Hopcroft, *Independence results in computer science*, SIGACT News 8(4):13-24, 1976.

[4] J. Hartmanis, *Independence results about context-free languages and lower bounds*, Information Proc. Lett. 20(5):241-248, 1985.

[5] Harry R.Lewis and Christos H.Papadimitriou, *Elements of the Theory of Computation 2nd Ed*, Prentice-Hall, 1998.

[6] Michael Sipser, *Introduction to the Theory of Computation 3rd Ed*, Cengage Learning, 2012.

[7] C. C. Chang and H. J. Keislelr, *Model Theory*, North-Holland, Amsterdam, 1990.

[8] W. Hodges, *Model Theory*, Cambridge University Press, 1993.

[9] David Marker, *Model Theory: An Introduction*, Springer, 2002.

[10] Yu. I. Maninr, *A Course in Mathematical Logic*,(Graduate texts in mathematics; 53) Springer-Verlag, 1977.

[11] Tianheng. Tsui, *Two theorems about the P versus NP problem*, https://arxiv.org/pdf/1805.01755.pdf

[12] https://en.wikipedia.org/wiki/Gödel's_incompleteness_theorems