

An Automated Theorem Generator with Theoretical Foundation Based on Rectangular Standard Contradiction

Yang Xu^{1,4†}, Peiyao Liu^{2,4†}, Shuwei Chen^{1,4†}, Jun Liu^{3,4†}

¹School of Mathematics, Southwest Jiaotong University, Chengdu, 611756, Sichuan, China.

²School of Computer and Software Engineering, Xihua University, Chengdu, 610039, Sichuan, China.

³School of Computing, Ulster University, Belfast, BT15 1ED, Northern Ireland, UK.

⁴National-Local Joint Engineering Laboratory of System Credibility Automatic Verification, Chengdu, 611756, Sichuan, China.

Contributing authors: xuyang@swjtu.edu.cn;
liupeiyao@mail.xhu.edu.cn; swchen@swjtu.edu.cn; j.liu@ulster.ac.uk;

[†]All the authors are co-first authors.

Abstract

Currently, there is a lack of rigorous theoretical system for systematically generating non-trivial and logically valid theorems. Addressing this critical gap, this paper conducts research to propose a novel automated theorem generation theory and tool. Based on the concept of standard contradiction which possesses unique deductive advantages, this paper defines and proves, for the first time, a new logical structure known as rectangular standard contradiction. Centered on this structure, a complete Automated Theorem Generation (ATG) theory is put forward. Theoretical proofs clarify two core properties of rectangular standard contradiction: first, it is a standard contradiction (necessarily unsatisfiable); second, it exhibits non-redundancy (the remaining clause set becomes satisfiable after removing any clause). Leveraging these properties, this paper proves that partitioning a rectangular standard contradiction into a premise subset \mathbf{A} and negation of its complement \mathbf{H} , a valid theorem $\mathbf{A} \vdash \neg\mathbf{H}$ can be formed, and all such theorems are logically equivalent. To implement this theory, an efficient template-based ATG algorithm is designed, and a Rectangular Automated

Theorem Generator is developed. This research enables machines to transition from "verifiers" to "discoverers", opening up new avenues for fundamental research in the fields of logic and artificial intelligence.

Keywords: Automated Theorem Generation; Standard Contradiction; Rectangular Standard Contradiction

1 Introduction

As a core branch of artificial intelligence and computer science, automated reasoning is dedicated to enabling computers to simulate human logical reasoning capabilities [1]. However, for a long time, the research paradigm in the field of automated reasoning has mainly focused on the "proving" phase, i.e., verifying the validity of an existing theorem [2-5]. Regarding how to systematically and automatically "generate" or "discover" new, non-trivial, and logically necessary theorems starting from a set of basic literals, there is still a lack of a complete and rigorous theoretical system. This role transition from "verifier" to a "discoverer" is of crucial significance for enhancing the creativity of machine intelligence [6, 7].

To fill this theoretical gap, this paper proposes a complete theory of Automated Theorem Generation (ATG). The cornerstone of this theory is a novel logical structure - "Rectangular Standard Contradiction" - which we have proposed and rigorously defined for the first time. The theory presented in this paper is built upon the concept of Standard Contradiction[8]. A standard contradiction refers to an unsatisfiable clause set, whose structural characteristics offer unique advantages for logical deduction [9, 10]. In recent years, research on Standard Contradiction has achieved significant progress, leading to the formation series of systematic theories and algorithms [11, 12].

The core contribution of this paper lies in the establishment of a complete theory capable of directly generating correct theorems. Since the correctness of the theorem is guaranteed by their construction method and the inherent logical properties of Rectangular Standard Contradiction - with rigorous mathematical proofs provided in this paper - theorems generated through this method require no further verification by any theorem prover or manual effort.

First, this paper rigorously defines the "Rectangular Standard Contradiction" and conducts in-depth theoretical proofs regarding its properties. Theorem 1 proves that an n -level Rectangular Standard Contradiction constructed from n generation literals is a standard contradiction, i.e., a necessarily unsatisfiable clause set. More importantly, Theorem 2 and its corollaries reveal its key "non-redundant" property: if any one or more clauses are removed from a complete Rectangular Standard Contradiction, the remaining clause set becomes satisfiable. This profound property directly forms the logical basis for theorem generation, as elaborated in Theorem 3: taking any subset of the Rectangular Standard Contradiction as the premise (A), the negation of its complement ($\neg H$) constitutes a logically valid theorem ($A \vdash \neg H$). Theorem 4 further proves that all theorems generated by a set of generation literals are logically equivalent.

On the solid theoretical foundation, this paper further designs and implements an efficient Automated Theorem Generation (ATG) algorithm (Algorithm 2). By adopting an innovative “template-based construction” method (Algorithm 1), this algorithm avoids the complexity of naive construction method and reliance on the literal management. It can quickly construct the corresponding Rectangular Standard Contradiction from any given and qualified literal set, and automatically output a new theorem consisting of premises and a conclusion. Currently, we have released the first version of the automated theorem generator based on rectangular standard contradiction on GitHub (<https://github.com/lpy-2019/Automated-Theorem-Generator---Rectangle>), which readers can download and use.

The fundamental theoretical significance of this study lies in its that it provides the theoretical basis for computers to automatically “generating” or “discovering” paradigm. By providing a systematic method for constructing unsatisfiable sets and deriving theorems from them, this work opens up new perspectives for basic theoretical research in logic and artificial intelligence [13, 14], and explores the possibility of machines performing logical creation [15].

The structure of this paper is organized as follows: Section 2 reviews preliminary knowledge such as first-order/propositional logic and the concept of standard contradiction. Section 3 elaborates on the definition and properties of Rectangular Standard Contradiction, and presents a rigorous mathematical proof of its status as a standard contradiction. Section 4 expounds the complete theory of theorem generation based on the properties of Rectangular Standard Contradiction. Section 5 provides the implementation of an automated algorithm for theorem generation, from template construction to final theorem output. Finally, Section 6 summarizes the entire paper and outlines future research directions.

2 Preliminaries

This paper involves concepts from both first-order logic and propositional logic. This section briefly reviews the fundamental concepts of these two logical systems [16, 17] and introduces the key notion of the standard contradiction, which forms the theoretical basis of our work.

Definition 1 (Term, First-Order Logic) A **term** is a syntactic structure in first-order logic that denotes individual objects, defined recursively by the follows rules:

- Individual constants (e.g., a, b, c and variables (e.g., x, y, z) are terms.
- If f is an n -ary function symbol and t_1, t_2, \dots, t_n are terms, then $f(t_1, t_2, \dots, t_n)$ is also a term.

Terms are used to construct atomic formulas, describing the properties or relationship of individual objects.

Definition 2 (Literal) A **literal** is either an atomic formula (**Atom**) or its negation. In propositional logic, an atom is a propositional variable (e.g., p, x_1). In first-order logic, an atom is a predicate applied to terms (e.g., $P(x)$)

- **Positive literal:** The atomic formula itself (e.g., $P(x), p$).
- **Negative literal:** The negation of an atomic formula (e.g., $\neg Q(a, b), \neg p$).

Literals are the basic units for constructing clauses, used to express true or false assertions of propositions.

Definition 3 (Clause) A **clause** is a disjunction (logical OR, \vee) of a finite set of literals.

- **Empty clause:** A clause containing no literals, denoted as \square .

Clauses are the core structure of the automated deduction rule.

Definition 4 (CNF Formula) A **Conjunctive Normal Form** (in short, CNF) formula is a conjunction (logical AND, \wedge) of a finite set of clause, i.e., it has the form C_1, C_2, \dots, C_n , where C_i is a clause.

Next is the definitions of the standard contradiction.

Definition 5 (Standard Contradiction) [10] Suppose a clause set $\mathcal{S} = \{C_1, C_2, \dots, C_m\}$ in propositional or first-order logic. If $\forall (l_1, \dots, l_m) \in \prod_{i=1}^m C_i$, there exists at least one complementary pair among l_1, l_2, \dots, l_m , then $\mathcal{S} = \bigwedge_{i=1}^m C_i$ is called a **standard contradiction**.

For a more detailed introduction to standard contradiction, please refer to reference [8].

3 The Rectangular Standard Contradiction

This section introduces and elaborates on the core of our theory - Rectangular Standard Contradiction. This novel logical construction is not only necessarily unsatisfiable, but more importantly, it possesses a unique “non-redundant” property, laying a solid logical foundation for the subsequent theorem generation theory.

3.1 Concept

We first start from a more general concept of “maximal contradiction” and then introduce “rectangular standard contradiction” as its special case.

Definition 6 (Maximal Contradiction) Suppose a literal set $\mathcal{L} = \{l_1, l_2, \dots, l_n\}$ in propositional or first-order logic, where l_i is either an atom p_i or a negated atom $\neg p_i$. A clause

$C(l_1, l_2, \dots, l_n) = \bigvee_{i=1}^n l_i$, a clause set $\mathcal{S} = \{C(l_1, l_2, \dots, l_n) | l_i \in \{p_i, \neg p_i\}, i = 1, 2, \dots, n\}$. Then \mathcal{S} is a standard contradiction containing $n \times 2^n$ literals that includes \mathcal{L} , and \mathcal{S} is called the **maximal contradiction** generated by \mathcal{L} .

Maximal contradiction provides us with a complete unsatisfiable set that covers all possibilities. However, its structure is overly broad. To obtain more refined properties, we introduce a highly structured and symmetric special case - rectangular standard contradiction.

According to Definition 6, the length of each clause in the maximal contradiction \mathcal{S} generated by \mathcal{L} is $|\mathcal{L}|$ (that is, the number of literals in \mathcal{L}), and there are a total of 2^n clauses in \mathcal{S} . If each of these 2^n clauses is arranged vertically in sequence, the form of \mathcal{S} is a rectangle (with a length of 2^n literals and a height of n literals).

Definition 7 [Rectangular Standard Contradiction] Suppose a literal set $\mathcal{L} = \{l_1, l_2, \dots, l_n\}$ in propositional or first-order logic, and $\mathcal{R}_{\mathcal{L}}^n$ is the maximal contradiction generated by \mathcal{L} . When $\mathcal{R}_{\mathcal{L}}^n$ satisfies the following recursive rules, $\mathcal{R}_{\mathcal{L}}^n$ (here “ n ” means the number of literals in \mathcal{L}) is a **rectangular standard contradiction**.

- i) Let $\mathcal{L}' = \mathcal{L} \setminus \{l_n\}$, and $\mathcal{R}_{\mathcal{L}'}^{n-1}$ is the rectangular standard contradiction generated by \mathcal{L}' ;
- ii) The rectangular form of $\mathcal{R}_{\mathcal{L}}^n$ is as follows:

$$\begin{array}{c} \begin{array}{|c|c|} \hline \mathcal{L}' & \mathcal{L}' \\ \hline l_n \cdots l_n & \neg l_n \cdots \neg l_n \\ \hline \end{array} \\ \underbrace{\hspace{1.5cm}}_{2^{n-1}} \quad \underbrace{\hspace{1.5cm}}_{2^{n-1}} \end{array} \quad (1)$$

The literal set \mathcal{L} is called the **generation literal set** of the rectangular standard contradiction $\mathcal{R}_{\mathcal{L}}^n$. The rectangular standard contradiction $\mathcal{R}_{\mathcal{L}}^n$ is also called **n -level rectangular standard contradiction**.

The recursive definition of Rectangular Standard Contradiction is the origin of all its excellent properties. It expands an $(n-1)$ -level rectangular standard contradiction into an n -level structure in a systematic manner, ensuring a high degree of symmetry and analyzability of the whole.

Remark 1 The generation literal set \mathcal{L} must satisfy the rule that no identical predicate symbols exist. In particular, the equality symbol (“=”) is a special type of predicate symbol.

This rule is proposed to avoid the occurrence of complementary pairs in the clauses within a rectangular standard contradiction after substitution and to ensure that there are no redundant clauses in the rectangular standard contradiction $\mathcal{R}_{\mathcal{L}}^n$ generated by \mathcal{L} .

Each column of $\mathcal{R}_{\mathcal{L}}^n$ is a clause, and usually, the first column of $\mathcal{R}_{\mathcal{L}}^n$ is a clause formed by the generation literal set \mathcal{L} .

From Definition 7, it can be known that $Length(\mathcal{L}') = 2^{n-1}$ (literals) and $Height(\mathcal{L}') = n - 1$ (literals). Combined Formula (1), it can be directly derived that

$Length(\mathcal{L}) = 2 \times Length(\mathcal{L}')$ and $Height(\mathcal{L}) = Height(\mathcal{L}') + 1$, where “1” means the last row $(l_n, \dots, l_n, \neg l_n, \dots, \neg l_n)$ of $\mathcal{R}_{\mathcal{L}}^n$.

Notably, in the last row $(l_n, \dots, l_n, \neg l_n, \dots, \neg l_n)$, the first 2^{n-1} literals are l_n , and the last 2^{n-1} literals are $\neg l_n$.

Next, we give two examples to illustrate the rectangular standard contradiction in propositional logic (Example 1) and first-order logic (Example 2).

Example 1 Let $\mathcal{L} = \{w, x, y, z\}$ is a generation literal set in propositional logic. The 4-level rectangular standard contradiction $\mathcal{R}_{\mathcal{L}}^4$ generated by \mathcal{L} is shown as follows. ($Height(\mathcal{R}_{\mathcal{L}}^4) = 4$, and $Length(\mathcal{R}_{\mathcal{L}}^4) = 2^4 = 16$)

$$\begin{bmatrix} w & \neg w & w & \neg w & w & \neg w & w & \neg w & w & \neg w & w & \neg w & w & \neg w & w & \neg w \\ x & x & \neg x & \neg x & x & x & \neg x & \neg x & x & x & \neg x & \neg x & x & x & \neg x & \neg x \\ y & y & y & y & \neg y & \neg y & \neg y & \neg y & y & y & y & y & \neg y & \neg y & \neg y & \neg y \\ z & z & z & z & z & z & z & z & \neg z & \neg z & \neg z & \neg z & \neg z & \neg z & \neg z & \neg z \end{bmatrix} \quad (2)$$

Example 2 Let $\mathcal{L} = \{P_1(a), P_2(f(x)), P_3(g(y, a))\}$ is a generation literal set in first-order logic. The rectangular standard contradiction $\mathcal{R}_{\mathcal{L}}^3$ generated by \mathcal{L} is shown as follows. ($Height(\mathcal{R}_{\mathcal{L}}^3) = 3$, and $Length(\mathcal{R}_{\mathcal{L}}^3) = 2^3 = 8$)

$$\begin{bmatrix} P_1(a) & \neg P_1(a) & P_1(a) & \neg P_1(a) & P_1(a) & \neg P_1(a) & P_1(a) & \neg P_1(a) \\ P_2(f(x)) & P_2(f(x)) & \neg P_2(f(x)) & \neg P_2(f(x)) & P_2(f(x)) & P_2(f(x)) & \neg P_2(f(x)) & \neg P_2(f(x)) \\ P_3(g(y, a)) & P_3(g(y, a)) & P_3(g(y, a)) & P_3(g(y, a)) & \neg P_3(g(y, a)) & \neg P_3(g(y, a)) & \neg P_3(g(y, a)) & \neg P_3(g(y, a)) \end{bmatrix} \quad (3)$$

3.2 Naive Construction

To gain a more intuitive understanding of the recursive structure in Definition 7, we propose a step-by-step expansion construction method, namely the **naive construction method**. Starting from a single literal, this method iterates level by level until a complete n -level rectangular standard contradiction $\mathcal{R}_{\mathcal{L}}^n$ is constructed. This is not only a construction process, but more importantly, a procedural interpretation of the recursive definition. The steps of the naive method are follows.

Let $\mathcal{L} = \{x_1, x_2, \dots, x_n\}$ is a generation literal set. For the convenience of illustration, the following sets are defined:

- $\mathcal{L}_1 = \{x_1\}$ The element is the first literal in \mathcal{L} .
- $\mathcal{L}_2 = \{x_1, x_2\}$ The elements are the first 2 literals in \mathcal{L} .
- \vdots
- $\mathcal{L}_i = \{x_1, x_2, \dots, x_i\}$ The elements are the first i literals in \mathcal{L} .
- \vdots
- $\mathcal{L}_{n-1} = \{x_1, x_2, \dots, x_{n-1}\}$ The elements are the first $n - 1$ literals in \mathcal{L} .
- $\mathcal{L}_n = \{x_1, x_2, \dots, x_n\} = \mathcal{L}$. The elements are the first n literals in \mathcal{L} , i.e., \mathcal{L} itself.

Step 1: Using the \mathcal{L}_1 to construct a 1-level rectangular standard contradiction $\mathcal{R}_{\mathcal{L}_1}^1$ which contains 2^1 clauses, then

$$\mathcal{R}_{\mathcal{L}_1}^1 = [x_1 \neg x_1] \quad (4)$$

Step 2: Using the \mathcal{L}_2 and $\mathcal{R}_{\mathcal{L}_1}^1$ (from Step 1) to construct 2-level rectangular standard contradiction $\mathcal{R}_{\mathcal{L}_2}^2$ which contains 2^2 clauses, then

$$\mathcal{R}_{\mathcal{L}_2}^2 = \left[\begin{array}{cc|cc} \mathcal{R}_{\mathcal{L}_1}^1 & \mathcal{R}_{\mathcal{L}_1}^1 \\ \hline x_2 & x_2 & \neg x_2 & \neg x_2 \end{array} \right] = \left[\begin{array}{cccc} x_1 & \neg x_1 & x_1 & \neg x_1 \\ x_2 & x_2 & \neg x_2 & \neg x_2 \end{array} \right] \quad (5)$$

Step 3: Using the \mathcal{L}_3 and $\mathcal{R}_{\mathcal{L}_2}^2$ (from Step 2) to construct 3-level rectangular standard contradiction $\mathcal{R}_{\mathcal{L}_3}^3$ which contains 2^3 clauses, then

$$\mathcal{R}_{\mathcal{L}_3}^3 = \left[\begin{array}{cccc|cccc} \mathcal{R}_{\mathcal{L}_2}^2 & \mathcal{R}_{\mathcal{L}_2}^2 \\ \hline x_3 & x_3 & x_3 & x_3 & \neg x_3 & \neg x_3 & \neg x_3 & \neg x_3 \end{array} \right] = \left[\begin{array}{cccccccc} x_1 & \neg x_1 & x_1 & \neg x_1 & x_1 & \neg x_1 & x_1 & \neg x_1 \\ x_2 & x_2 & \neg x_2 & \neg x_2 & x_2 & x_2 & \neg x_2 & \neg x_2 \\ x_3 & x_3 & x_3 & x_3 & \neg x_3 & \neg x_3 & \neg x_3 & \neg x_3 \end{array} \right] \quad (6)$$

...

Step i : Using the \mathcal{L}_{i-1} and $\mathcal{R}_{\mathcal{L}_{i-1}}^{i-1}$ (from Step $i-1$) to construct i -level rectangular standard contradiction $\mathcal{R}_{\mathcal{L}_i}^i$ which contains 2^{i-1} clauses, then

$$\mathcal{R}_{\mathcal{L}_i}^i = \left[\begin{array}{cc|cc} \mathcal{R}_{\mathcal{L}_{i-1}}^{i-1} & \mathcal{R}_{\mathcal{L}_{i-1}}^{i-1} \\ \hline x_i & \cdots & x_i & \neg x_i & \cdots & \neg x_i \end{array} \right] \quad (7)$$

$\underbrace{\hspace{1.5cm}}_{2^{i-1}} \quad \underbrace{\hspace{1.5cm}}_{2^{i-1}}$

...

Step n : Using the \mathcal{L} and $\mathcal{R}_{\mathcal{L}_{n-1}}^{n-1}$ (from Step $n-1$) to construct n -level rectangular standard contradiction $\mathcal{R}_{\mathcal{L}}^n$ which contains 2^n clauses, then

$$\mathcal{R}_{\mathcal{L}}^n = \left[\begin{array}{cc|cc} \mathcal{R}_{\mathcal{L}_{n-1}}^{n-1} & \mathcal{R}_{\mathcal{L}_{n-1}}^{n-1} \\ \hline x_n & \cdots & x_n & \neg x_n & \cdots & \neg x_n \end{array} \right] \quad (8)$$

$\underbrace{\hspace{1.5cm}}_{2^{n-1}} \quad \underbrace{\hspace{1.5cm}}_{2^{n-1}}$

3.3 Proof

To establish Rectangular Standard Contradiction as the logical cornerstone of our theory, it is first necessary to rigorously prove its core property: it is a standard contradiction, i.e., a necessarily unsatisfiable clause set. The following theorem completes this critical proof using mathematical induction.

Theorem 1 *Let \mathcal{L} be a generation literal set containing n literals, then the rectangular standard contradiction $\mathcal{R}_{\mathcal{L}}^n$ generated by \mathcal{L} is a standard contradiction.*

Proof We use mathematical induction on the number of literals in \mathcal{L} .

Base Case: $n = 1$.

When $n = 1$, the generation literal set \mathcal{L}_1 contains exactly 1 literal, denoted $\mathcal{L}_1 = \{x\}$, then

$$\mathcal{R}_{\mathcal{L}_1}^1 = [x \neg x] .$$

Obviously, $\mathcal{R}_{\mathcal{L}_1}^1$ is a standard contradiction with simplest structure. Thus, the theorem holds when $n = 1$.

Inductive Hypothesis: Assume the theorem holds for any generation literal set \mathcal{L}_k with $n = k$ literals (where $k \geq 1$). That is, for any \mathcal{L}_k with $|\mathcal{L}_k| = k$, the rectangular standard contradiction $\mathcal{R}_{\mathcal{L}_k}^k$ generated by \mathcal{L}_k is a standard contradiction, denoted as

$$\mathcal{R}_{\mathcal{L}_k}^k = \bigwedge_{i=1}^{2^k} C_i = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1i} & \cdots & x_{12^k} \\ x_{21} & x_{22} & \cdots & x_{2i} & \cdots & x_{22^k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{ki} & \cdots & x_{k2^k} \end{bmatrix} .$$

There are k rows and 2^k columns in $\mathcal{R}_{\mathcal{L}_k}^k$, and any column $C_i = \{x_{1i}, x_{2i}, \dots, x_{ki}\}$ represents a clause. Thus, for any $(x^{[1]}, x^{[2]}, \dots, x^{[2^k]}) \in \prod_{i=1}^{2^k} C_i$, there exists one complementary pair among $\{x^{[1]}, x^{[2]}, \dots, x^{[2^k]}\}$. ($x^{[i]} \in C_i, i = 1, \dots, 2^k$)

Inductive Step: We now prove that the theorem holds for $n = k + 1$.

Let \mathcal{L}_{k+1} be a generation literal set with $n = k + 1$ literals. We can decompose \mathcal{L}_{k+1} as $\mathcal{L}_k \cup x$, where \mathcal{L}_k is a subset of \mathcal{L}_{k+1} with $|\mathcal{L}_k| = k$ literals, and $x \notin \mathcal{L}_k$ (the “new” literal added to \mathcal{L}_k). Then the rectangular standard contradiction $\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1}$ generated by \mathcal{L}_{k+1} is denoted as follows:

$$\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{12^k} & x_{11} & x_{12} & \cdots & x_{12^k} \\ x_{21} & x_{22} & \cdots & x_{22^k} & x_{21} & x_{22} & \cdots & x_{22^k} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{k2^k} & x_{k1} & x_{k2} & \cdots & x_{k2^k} \\ x & x & \cdots & x & \neg x & \neg x & \cdots & \neg x \end{bmatrix} = \underbrace{\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{12^k} \\ x_{21} & x_{22} & \cdots & x_{22^k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{k2^k} \end{bmatrix}}_{2^k} \underbrace{\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{12^k} \\ x_{21} & x_{22} & \cdots & x_{22^k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{k2^k} \end{bmatrix}}_{2^k} = \underbrace{\begin{bmatrix} \mathcal{R}_{\mathcal{L}_k}^k & \mathcal{R}_{\mathcal{L}_k}^k \\ x & x & \cdots & x & \neg x & \neg x & \cdots & \neg x \end{bmatrix}}_{2^k} .$$

for any $(x^{[1]}, x^{[2]}, \dots, x^{[2^k]}, y^{[1]}, y^{[2]}, \dots, y^{[2^k]}) \in \prod_{i=1}^{2^k} (C_i \cup \{x\}) \times \prod_{i=1}^{2^k} (C_i \cup \{\neg x\})$, four cases may arise, which we discuss one by one:

a) If there no $x \in \{x^{[1]}, x^{[2]}, \dots, x^{[2^k]}\}$, and exists $\neg x \in \{y^{[1]}, y^{[2]}, \dots, y^{[2^k]}\}$, then there exists at least one complementary pair among $\{x^{[1]}, x^{[2]}, \dots, x^{[2^k]}\}$. Thus, there exists one complementary pair among $(x^{[1]}, x^{[2]}, \dots, x^{[2^k]}, y^{[1]}, y^{[2]}, \dots, y^{[2^k]})$.

b) If there no $\neg x \in \{y^{[1]}, y^{[2]}, \dots, y^{[2^k]}\}$, and exists $x \in \{x^{[1]}, x^{[2]}, \dots, x^{[2^k]}\}$, then there exists at least one complementary pair among $\{y^{[1]}, y^{[2]}, \dots, y^{[2^k]}\}$. Thus, there exists one complementary pair among $(x^{[1]}, x^{[2]}, \dots, x^{[2^k]}, y^{[1]}, y^{[2]}, \dots, y^{[2^k]})$.

c) If there no $x \in \{x^{[1]}, x^{[2]}, \dots, x^{[2^k]}\}$, and exists $\neg x \in \{y^{[1]}, y^{[2]}, \dots, y^{[2^k]}\}$, then there exists one complementary pair $(x, \neg x)$ among $(x^{[1]}, x^{[2]}, \dots, x^{[2^k]}, y^{[1]}, y^{[2]}, \dots, y^{[2^k]})$.

d) If there no $x \in \{x^{[1]}, x^{[2]}, \dots, x^{[2^k]}\}$, and exists $\neg x \in \{y^{[1]}, y^{[2]}, \dots, y^{[2^k]}\}$, then there exists obviously one complementary pair among $(x^{[1]}, x^{[2]}, \dots, x^{[2^k]}, y^{[1]}, y^{[2]}, \dots, y^{[2^k]})$.

In summary, for any $(x^{[1]}, x^{[2]}, \dots, x^{[2^k]}, y^{[1]}, y^{[2]}, \dots, y^{[2^k]}) \in \prod_{i=1}^{2^k} (C_i \cup \{x\}) \times \prod_{i=1}^{2^k} (C_i \cup \{\neg x\})$, there exists one complementary pair within it; that is, $\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1}$ is a standard contradiction. Therefore, the theorem holds when $n = k + 1$.

By the principle of mathematical induction, this theorem is thus proven. \square

Each column of $\mathcal{R}_{\mathcal{L}}^n$ is a clause, $\mathcal{R}_{\mathcal{L}}^n$ is also a CNF formula. Furthermore, according to Theorem 1, it follows that $\mathcal{R}_{\mathcal{L}}^n$ is UNSAT. The proof of Theorem 1 is of crucial importance, as it establishes the status of Rectangular Standard Contradiction as a logically necessarily false (UNSAT) clause set. Taking this as a starting point, we will explore its deeper structural properties in the next section, and these properties directly lead to the automated generation of theorems.

4 Theorem Generation Based on Rectangular Standard Contradiction

In the previous section, we proved that Rectangular Standard Contradiction is a standard contradiction (UNSAT). This section will reveal its more profound "non-redundant" or "minimally unsatisfiable" property, and based on this, construct a complete theory of Automated Theorem Generation (ATG). This theory will clearly demonstrate how to systematically "extract" new, logically valid theorems from an unsatisfiable structure.

Definition 8 A rectangular standard contradiction is referred to as a **full rectangular standard contradiction**, meaning that each column in the rectangle is filled with literals.

Now, we introduce a key theorem that serves as a link between the preceding and subsequent content in our theoretical system, which reveals the non-redundancy of Rectangular Standard Contradiction.

Theorem 2 *There are no redundant clauses in a full rectangular standard contradiction (that is, the remaining clause set after removing any one clause from the full rectangular standard contradiction is satisfiable, i.e., SAT).*

Proof Let $\mathcal{R}_{\mathcal{L}}^n$ is a full rectangular standard contradiction where \mathcal{L} is a generation literal set containing n distinct literals. We still use mathematical induction on the number of literals in \mathcal{L} .

Base Case: $n = 1$

When $n = 1$, the generation literal set \mathcal{L}_1 contains exactly 1 literal, denoted as $\mathcal{L}_1 = \{x\}$, then

$$\mathcal{R}_{\mathcal{L}_1}^1 = [x \quad \neg x] .$$

It is obviously that there are no redundant clauses in $\mathcal{R}_{\mathcal{L}_1}^1$, Plus, the theorem holds when $n = 1$.

Inductive Hypothesis: Assume the theorem holds for any generation literal set \mathcal{L}_k with $n = k$ literals (where $k \geq 1$). That is, for any \mathcal{L}_k with $|\mathcal{L}_k| = k$, the rectangular standard contradiction $\mathcal{R}_{\mathcal{L}_k}^k$ generated by \mathcal{L}_k is a standard contradiction, denoted as

$$\mathcal{R}_{\mathcal{L}_k}^k = \bigwedge_{i=1}^{2^k} C_i = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1i} & \cdots & x_{12^k} \\ x_{21} & x_{22} & \cdots & x_{2i} & \cdots & x_{22^k} \\ \vdots & \vdots & & \vdots & & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{ki} & \cdots & x_{k2^k} \end{bmatrix}.$$

There are k rows and 2^k columns in $\mathcal{R}_{\mathcal{L}_k}^k$, and any column $C_i = \{x_{1i}, x_{2i}, \dots, x_{ki}\}$ is a clause. After arbitrarily extracting one clause from $\mathcal{R}_{\mathcal{L}_k}^k$, the remaining clause set is satisfiable.

Inductive Step: We now prove that the theorem holds for $n = k + 1$.

Let \mathcal{L}_{k+1} be a generation literal set with $n = k + 1$ literals. We can decompose \mathcal{L}_{k+1} as $\mathcal{L}_k \cup x$, where \mathcal{L}_k is a subset of \mathcal{L}_{k+1} with $|\mathcal{L}_k| = k$ literals, and $x \notin \mathcal{L}_k$ (the “new” literal added to \mathcal{L}_k). Then the rectangular standard contradiction $\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1}$ generated by \mathcal{L}_{k+1} is denoted as follows:

$$\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{12^k} & x_{11} & x_{12} & \cdots & x_{12^k} \\ x_{21} & x_{22} & \cdots & x_{22^k} & x_{21} & x_{22} & \cdots & x_{22^k} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{k2^k} & x_{k1} & x_{k2} & \cdots & x_{k2^k} \\ x & x & \cdots & x & \neg x & \neg x & \cdots & \neg x \end{bmatrix} = \underbrace{\begin{bmatrix} \mathcal{R}_{\mathcal{L}_k}^k & \mathcal{R}_{\mathcal{L}_k}^k \\ x & x & \cdots & x & \neg x & \neg x & \cdots & \neg x \end{bmatrix}}_{\substack{2^k \quad 2^k}}.$$

Notably, $\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1}$ is also a clause set, i.e., $\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1} = \{\bigwedge_{i=1}^{2^k} (C_i \vee x)\} \wedge \{\bigwedge_{i=1}^{2^k} (C_i \vee \neg x)\}$.

Arbitrarily extract a clause; for the sake of notational convenience, w.l.o.g, extract the first clause $C_1 \vee x$ from $\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1}$ (since all clauses in the clause set $\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1}$ are of equal status), denoted as follows:

$$\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1} \setminus \{C_1 \vee x\} = \begin{bmatrix} x_{12} & \cdots & x_{12^k} & x_{11} & x_{12} & \cdots & x_{12^k} \\ x_{22} & \cdots & x_{22^k} & x_{21} & x_{22} & \cdots & x_{22^k} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ x_{k2} & \cdots & x_{k2^k} & x_{k1} & x_{k2} & \cdots & x_{k2^k} \\ x & \cdots & x & \neg x & \neg x & \cdots & \neg x \end{bmatrix}.$$

$\underbrace{\hspace{10em}}_{2^k-1} \quad \underbrace{\hspace{10em}}_{2^k}$

By the inductive hypothesis, the clause set $\bigwedge_{i=2}^{2^k} C_i$ composed of all literals marked in yellow is SAT (where $C_i = \{x_{1i}, x_{2i}, \dots, x_{ki}\}$), meaning there exists a set of satisfying instances $Y = (y_2, \dots, y_{2^k})$ that makes $\bigwedge_{i=1}^{2^k} C_i$ satisfied. It is not difficult to deduce that this satisfying instances Y renders the clause set $\bigwedge_{i=2}^{2^k} (C_i \vee x)$ SAT. Furthermore, By adding 2^k instances of $\neg x$ to Y results in a new literal set $Y' = (y_2, \dots, y_{2^k}, \neg x, \neg x, \dots, \neg x)$, which renders the clause set $\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1} \setminus \{C_1 \vee x\}$ SAT.

It should also be noted that there are no complementary pairs in $Y = (y_2, \dots, y_{2^k})$, so there are no complementary pairs in $Y' = (y_2, \dots, y_{2^k}, \neg x, \neg x, \dots, \neg x)$ either. Moreover, the $2^{k+1} - 1 = 2^k - 1 + 2^k$ literals in Y' corresponding exactly to the literals in $C_2 \vee x, \dots, C_{2^k} \vee x, C_1 \vee \neg x, \dots, C_{2^k} \vee \neg x$, which means Y' is a satisfying instance of the clause set $\mathcal{R}_{\mathcal{L}_{k+1}}^{k+1} \setminus \{C_1 \vee x\} = \{C_2 \vee x, \dots, C_{2^k} \vee x, C_1 \vee \neg x, \dots, C_{2^k} \vee \neg x\}$.

Therefore, the theorem holds when $n = k + 1$.

By the principle of mathematical induction, this theorem is thus proven. \square

Theorem 2 reveals a profound dual property: a full rectangular standard contradiction as a whole is unsatisfiable, yet any proper subset of the clause set that constitutes the full rectangular standard contradiction is satisfiable. This property can be naturally extended to scenarios where multiple clauses are removed.

Corollary 1 *Let $\mathcal{R}_{\mathcal{L}^n}^n$ is a full rectangular standard contradiction. After removing any k ($k \leq 2^n$) clauses from $\mathcal{R}_{\mathcal{L}^n}^n$, the remaining clause set is satisfiable, i.e., SAT.*

Proof As can be seen from Theorem 2, after any one clause from a full rectangular standard contradiction $\mathcal{R}_{\mathcal{L}^n}^n$, remaining clause set is SAT (that is, there will be no complementary pairs in the literal tuple formed by arbitrarily selecting $2^n - 1$ literals from the remaining $2^n - 1$ clauses).

Then, after removing any k clauses from $\mathcal{R}_{\mathcal{L}^n}^n$, there also is no complementary pairs in the literal tuple formed by arbitrarily selecting $2^n - k$ literals from the remaining $2^n - k$ clauses. Therefore, after removing any k clauses from a rectangular standard contradiction, the remaining clause set is SAT.

When $k = 2^n$, the remaining clause set is an empty set, and the empty set generally considered SAT. \square

Theorem 1, Theorem 2, and Corollary 1 together form the logical foundation for theorem generation, and ensure that the theorems generated are meaningful - specifically, that such theorems are not derived from unsatisfiable clause sets.

Theorem 3 *For any full rectangular standard contradiction $\mathcal{R}_{\mathcal{L}^n}^n$, let $H = \{H_1, H_2, \dots, H_k\}$ ($k \leq 2^n$) be an arbitrary subset of $\mathcal{R}_{\mathcal{L}^n}^n$, and let $A = \mathcal{R}_{\mathcal{L}^n}^n \setminus H$. Then $A \vdash \neg H$ is theorem, where A is the premise and $\neg H$ is the hypothesis.*

Remark 2 $A \vdash \neg H$ is referred to as a theorem generated by $\mathcal{R}_{\mathcal{L}^n}^n$.

An intuitive interpretation of Theorem 3 is as follows: Since the premise A combined with the conclusion H forms a full rectangular standard contradiction ($A \wedge H \equiv \text{UNSAT}$), this is equivalent to $A \wedge (\neg \neg H) \equiv \text{UNSAT}$, which can also be expressed as $A \wedge \neg(\neg H) \equiv \text{UNSAT}$. According to the deduction theorem, this indicates that starting from the premise A , the hypothesis $\neg H$ can necessarily be derived.

Furthermore, we find that all theorems generated from the same rectangular contradictory are logically equivalent, which greatly simplifies the selection and representation of theorems.

Theorem 4 *Let \mathcal{L} is a generation literal set with n literals, and $\mathcal{R}_{\mathcal{L}^n}^n$ be a full rectangular standard contradiction generated by \mathcal{L} . Then the theorems generated by $\mathcal{R}_{\mathcal{L}^n}^n$ are mutually equivalent.*

Proof Let C is an arbitrary clause in $\mathcal{R}_{\mathcal{L}}^n$, for $\forall r, k \leq 2^n$, let $\{E_1, E_2, \dots, E_r\}$ and D_1, D_2, \dots, D_k be subsets of $\mathcal{R}_{\mathcal{L}}^n$. Then

$$\begin{aligned} \mathcal{R}_{\mathcal{L}}^n \setminus \{C\} \vdash \neg C &\text{ iff } \mathcal{R}_{\mathcal{L}}^n \text{ is unsatisfiable iff } S \setminus \{E_1, E_2, \dots, E_r\} \vdash \neg(E_1 \wedge E_2 \wedge \dots \wedge E_r), \\ \mathcal{R}_{\mathcal{L}}^n \setminus \{C\} \vdash \neg C &\text{ iff } \mathcal{R}_{\mathcal{L}}^n \text{ is unsatisfiable iff } S \setminus \{D_1, D_2, \dots, D_k\} \vdash \neg(D_1 \wedge D_2 \wedge \dots \wedge D_k), \\ \mathcal{R}_{\mathcal{L}}^n \setminus \{E_1, E_2, \dots, E_r\} \vdash \neg\{E_1, E_2, \dots, E_r\} &\text{ iff } S \setminus \{D_1, D_2, \dots, D_k\} \vdash \neg(D_1 \wedge D_2 \wedge \dots \wedge D_k). \end{aligned}$$

This theorem is thus proven. \square

Remark 3 It follows from Theorem 4 that among all theorems generated by a full rectangular standard contradiction $\mathcal{R}_{\mathcal{L}}^n$, only one theorem needs to be selected for representation, i.e., $S \setminus \{C\} \vdash \neg C$.

5 Automated Theorem Generation Algorithm

The previous two sections have laid a solid theoretical foundation for Automated Theorem Generation (ATG). This section puts the theory into practice by designing and implementing an efficient automated algorithm. Taking any valid set of generating literals as input, this algorithm can automatically construct the corresponding Rectangular Standard Contradiction and output a new theorem consisting of premises and a conclusion. The core lies in an innovative “template-based construction method” we proposed, which is significantly superior to the naive construction method described in Section 3.2.

5.1 Template-Based Construction Method

A close observation of the structure of an n -level rectangular standard contradiction $\mathcal{R}_{\mathcal{P}_n}^n$ reveals that its inherent pattern is independent of specific literals and only related to the polarity of the literals (positive or negative). For instance, an n -level rectangular standard contradiction can be viewed as two $(n-1)$ -level rectangular standard contradictions placed side by side, with a new row of literals and their negations appended below.

$$\mathcal{R}_{\mathcal{L}_n}^n = \left[\begin{array}{c|c} \mathcal{R}_{\mathcal{L}_{n-1}}^{n-1} & \mathcal{R}_{\mathcal{L}_{n-1}}^{n-1} \\ \hline x \ x \ \dots \ x & \neg x \ \neg x \ \dots \ \neg x \end{array} \right].$$

$\underbrace{\hspace{10em}}_{2^{n-1}} \quad \underbrace{\hspace{10em}}_{2^{n-1}}$

We can abstract this polarity structure, using “!” to represent positive literals and “?” to represent negative literals, thereby obtaining a general n -level Rectangular Standard Contradiction template. For any generating set containing n literals, its Rectangular Standard Contradiction follows the same n -level template.

The literals directly below $\mathcal{R}_{\mathcal{L}_{n-1}}^{n-1}$ on the left are 2^{n-1} generation literals x , and the literals directly below $\mathcal{R}_{\mathcal{L}_{n-1}}^{n-1}$ on the right are the opposite counterparts of these 2^{n-1} generation literals x , i.e., $\neg x$. We can abstract this polarity structure, using “!” to represent positive literals and “?” to represent negative literals, thereby obtaining a

general n -level rectangular standard contradiction template. For any set of generation literals containing n literals, its rectangular standard contradiction follows the same n -level template. The structure of $\mathcal{R}_{\mathcal{L}_n}^n$ can be transformed as follows:

$$\begin{bmatrix} ! & ? & \dots & ? & ! & ? & \dots & ? \\ ! & ! & \dots & ? & ! & ! & \dots & ? \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ ! & ! & \dots & ? & ! & ! & \dots & ? \\ ! & ! & \dots & ! & ? & ? & \dots & ? \end{bmatrix} \quad (9)$$

$\underbrace{\hspace{1.5cm}}_{2^{n-1}} \quad \underbrace{\hspace{1.5cm}}_{2^{n-1}}$

The structure in Formula (9) is called an **n -level rectangular standard contradiction template**. That is, the template of the rectangular standard contradiction generated by generation literal set containing n literals is the same n -level template.

Example 3 The 3-level and 4-level rectangular standard contradiction templates are as follows.

$$\begin{aligned} \text{3-level template: } & \begin{bmatrix} ! & ? & ! & ? & ? & ! & ? & ! & ? \\ ! & ! & ? & ? & ? & ! & ! & ? & ? \\ ! & ! & ! & ! & ? & ? & ? & ? & ? \end{bmatrix}; \\ \text{4-level template: } & \begin{bmatrix} ! & ? & ! & ? & ? & ! & ? & ! & ? & ? & ! & ? & ? & ! & ? \\ ! & ! & ? & ? & ? & ! & ? & ? & ! & ! & ? & ? & ? & ! & ? \\ ! & ! & ! & ! & ? & ? & ? & ? & ! & ! & ! & ? & ? & ? & ? \\ ! & ! & ! & ! & ! & ! & ! & ! & ? & ? & ? & ? & ? & ? & ? \end{bmatrix}. \end{aligned}$$

Compared with the naive construction method, the template-based method completely separates structure generation from content population. We first efficiently generate an abstract polarity template, and then "populate" the specific generating literal set into the template. This method avoids the complex management of the literals themselves during the construction process, making the algorithm logic clearer and the execution efficiency higher.

Algorithm 1 is the pseudo-code for generating an n -level template. By means of an iterative approach, the template is constructed incrementally from the 1-level template up to the n -level template. Each step only process the current level, resulting in extremely high efficiency and clear logic.

5.2 ATG Algorithm

Combined with Algorithm 1, we finally propose a complete Automated Theorem Generation (ATG) algorithm (i.e., Algorithm 2). This algorithm encapsulates the entire process from theory to practice: it receives a generating literal set, constructs a Rectangular Standard Contradiction via the template, automatically divides the premises and conclusion according to Theorem 3 and Remark 3, and finally outputs a brand-new theorem. Eventually, based on the ATG algorithm, we have implemented the **Automated Theorem Generator - Rectangle**, whose first version has been released on

Algorithm 1 n -level Template Construction Method

Input: A positive integer n indicating the template level.

Output: An n -level full rectangular standard contradiction template (2D array).

```
1: if  $n \leq 0$  then
2:   return empty list
3: end if
4:  $current\_template \leftarrow [ \text{"!"}, \text{"?"} ]$ 
5: for  $k \leftarrow 2$  to  $n$  do
6:    $extended\_rows \leftarrow \text{empty list}$ 
7:   for each  $row \in current\_template$  do
8:      $extended\_rows.append(row \parallel row)$ 
9:   end for
10:   $half\_length \leftarrow 2^{k-1}$ 
11:   $new\_row \leftarrow ([\text{"!"}] \times half\_length) \parallel ([\text{"?"}] \times half\_length)$ 
12:   $current\_template \leftarrow extended\_rows$ 
13:   $current\_template.APPEND(new\_row)$ 
14: end for
15: return  $current\_template$ 
```

▷ 1. Initialization: Start with the 1-level template
▷ 2. Iterative construction: Build up from 2-level to n -level
▷ a. Duplicate and extend: Copy and concatenate each row of the previous level template
▷ b. Generate new row: Create a new row with 2^{k-1} "!" followed by 2^{k-1} "?"
▷ c. Merge: Combine the extended rows and the new row to form the current k -level template
▷ 3. Return the final result

GitHub (<https://github.com/lpy-2019/Automated-Theorem-Generator---Rectangle>). This tool features a clear GUI, and anyone with basis mathematical logic knowledge can quickly get started with in.

6 Conclusions and Future Work

This paper introduces a complete and rigorous theory for Automated Theorem Generation (ATG), and focus on theorem discovery by formalizing the “Rectangular Standard Contradiction”, a novel, recursively defined logical structure. We have proven that this structure is a non-redundant unsatisfiable clause set, which provides a systematic method for generating new, valid theorems by partitioning the structure into premises and a conclusion.

The significance of this work lies in providing a complete theoretical framework, complemented by an efficient template-based algorithm, that empowers machines to transition from logical ‘verifiers’ to ‘discoverers’. This paradigm shift opens new possibilities for machine creativity and advances fundamental research in logic and artificial intelligence, establishing a solid foundation for the automated discovery of new knowledge.

Algorithm 2 Automated Theorem Generation (ATG)

Input: A literal set \mathcal{L} .**Output:** A theorem (clause set).

```
1:  $n \leftarrow |\mathcal{L}|$  ▷ Get the number of literals in  $\mathcal{L}$ 
2:  $num\_clauses \leftarrow 2^n$  ▷ Get the number of clauses in the rectangular standard contradiction
3:  $\mathcal{R} \leftarrow []$  ▷ Initialization of the rectangular standard contradiction (clause set form)
   ▷ Call Algorithm 1 to generate a  $n$ -level template (2D array form)
4:  $template \leftarrow \text{MAKE\_TEMPLATE}(n)$ 
5: for  $i \leftarrow 1$  to  $num\_clause$  do ▷ Constructed clauses column by column
6:    $col\_array \leftarrow template[i]$ 
7:    $clause \leftarrow []$  ▷ Initialization of a clause
8:   for  $j \leftarrow 1$  to  $n$  do ▷ Fill clauses row by by row
9:     if  $col\_array[j] \neq \text{"?"}$  then ▷ Fill in the corresponding positive literal
10:       $clause.APPEND(\text{GENERATING\_LITERAL}(\mathcal{R}[j]))$ 
11:     else ▷ Fill in the corresponding negative literal
12:        $clause.APPEND(\text{NEGATION}(\text{GENERATING\_LITERAL}(\mathcal{R}[j])))$ 
13:     end if
14:   end for
15:    $\mathcal{R}.APPEND(clause)$ 
16: end for
17:  $hypothesis \leftarrow \text{NEGATION}(\mathcal{R}[1])$  ▷ Negate  $\mathcal{R}$ 's 1st clause as the theorem's hypothesis
18:  $premises \leftarrow \{\mathcal{R}[2], \mathcal{R}[3], \dots, \mathcal{R}[2^n]\}$  ▷ Remaining clauses are the theorem's premises
19:  $theorem \leftarrow \{hypothesis, premises\}$  ▷ Get the theorem
20: return  $theorem$ 
```

As our conclusions remain valid when logical literals are replaced with first-order logic formulas, our primary focus for future work will be extending the automated theorem generator to accept first-order closed formulas as input, thereby significantly broadening its application scenarios. Additionally, we aim to deeply integrate domain expertise - particularly from mathematics - with our automated theorem generator to establish a technical pathway capable of generating framework with practical semantic value. Finally, there remains considerable scope for meaningful theoretical research in automated theorem generation, which constitutes another important direction for our future endeavors.

Declarations

Funding This work has been supported by the Key Project of Sichuan Science and Technology Innovation and Entrepreneurship Seeding Program (Grant No. 2024JDR0084).

(Competing interests) The authors declare no competing interests.

References

- [1] Russell, S., Norvig, P.: Artificial intelligence: a modern approach, 4th us ed. aima (2021)
- [2] Pantsar, M.: Theorem proving in artificial neural networks: new frontiers in mathematical ai. *European Journal for Philosophy of Science* **14**(1), 4 (2024)
- [3] Coward, S., Paulson, L., Drane, T., Morini, E.: Formal verification of transcendental fixed-and floating-point algorithms using an automatic theorem prover. *Formal Aspects of Computing* **34**(2), 1–22 (2022)
- [4] Stock, S., Dunkelau, J., Mashkoor, A.: Application of ai to formal methods—an analysis of current trends. *arXiv preprint arXiv:2411.14870* (2024)
- [5] Hozzová, P., Kovács, L., Norman, C., Voronkov, A.: Program synthesis in saturation. In: *International Conference on Automated Deduction*, pp. 307–324 (2023). Springer
- [6] Delgrande, J.P., Glimm, B., Meyer, T., Truszczyński, M., Wolter, F.: Current and future challenges in knowledge representation and reasoning (dagstuhl perspectives workshop 22282). *Dagstuhl Manifestos* **10**(1), 1–61 (2024)
- [7] Lin, X., Cao, Q., Huang, Y., Yang, Z., Liu, Z., Li, Z., Liang, X.: Atg: Benchmarking automated theorem generation for generative language models. *arXiv preprint arXiv:2405.06677* (2024)
- [8] Xu, Y., Chen, S., Zhong, X., Liu, J., He, X.: Contradictions. *arXiv preprint arXiv:2509.07026* (2025)
- [9] Xu, Y., Chen, S., Liu, J., Zhong, X., He, X.: Distinctive features of the contradiction separation based dynamic automated deduction. In: *Data Science and Knowledge Engineering for Sensing Decision Support: Proceedings of the 13th International FLINS Conference (FLINS 2018)*, pp. 725–732 (2018). World Scientific
- [10] Xu, Y., Liu, J., Chen, S., Zhong, X., He, X.: Contradiction separation based dynamic multi-clause synergized automated deduction. *Information Sciences* **462**, 93–113 (2018)
- [11] Xu, Y., Chen, S., Liu, J., Cao, F., He, X.: Extended triangular method: A generalized algorithm for contradiction separation based automated deduction. *arXiv preprint arXiv:2510.10701* (2025)
- [12] Xu, Y., He, X., Chen, S., Liu, J., Zhong, X.: Dynamic automated deduction by contradiction separation: the standard extension algorithm. *arXiv preprint arXiv:2510.08468* (2025)

- [13] Liu, L., Wang, Z., Tong, H.: Neural-symbolic reasoning over knowledge graphs: A survey from a query perspective. *ACM SIGKDD Explorations Newsletter* **27**(1), 124–136 (2025)
- [14] Ferrag, M.A., Tihanyi, N., Debbah, M.: From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678* (2025)
- [15] Abdelaziz, I., Crouse, M., Makni, B., Austel, V., Cornelio, C., Ikbali, S., Kapapathi, P., Makondo, N., Srinivas, K., Witbrock, M., *et al.*: Learning to guide a saturation-based theorem prover. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **45**(1), 738–751 (2022)
- [16] Liu, P., Chen, S., Liu, J., Xu, Y., Cao, F., Wu, G.: An efficient contradiction separation based automated deduction algorithm for enhancing reasoning capability. *Knowledge-Based Systems* **261**, 110217 (2023)
- [17] Gleiss, B., Kovács, L., Rath, J.: Subsumption demodulation in first-order theorem proving. In: *International Joint Conference on Automated Reasoning*, pp. 297–315 (2020). Springer