# Automated Theorem Finding by Forward Reasoning Based on Strong Relevant Logic: A Case Study in Graph Theory

Hongbiao Gao, Yuichi Goto, and Jingde Cheng

Department of Information and Computer Sciences,
Saitama University, Saitama 338-8570, Japan
{gaohongbiao,gotoh,cheng}@aise.ics.saitama-u.ac.jp

**Abstract.** The problem of automated theorem finding is one of 33 basic research problems in automated reasoning which was originally proposed by Wos. The problem is still an open problem until now. To solve the problem, a systematic methodology with forward reasoning based on strong relevant logic has been proposed. This paper presents a case study of automated theorem finding in graph theory to show the generality of the methodology, and presents a future direction for automated theorem finding based on the methodology.

**Keywords:** Automated theorem finding, forward reasoning, strong relevant logic, graph theory.

## 1 Introduction

The problem of automated theorem finding (ATF for short) is one of 33 basic research problems in automated reasoning which was originally proposed by Wos in 1988 [14,15], and it is still an open problem [6, 9, 10, 12-15]. The ATF problem [14, 15]: "What properties can be identified to permit an automated reasoning program to find new and interesting theorems, as opposed to proving conjectured theorems?"

The most important and difficult requirement of the problem is that, in contrast to prove conjectured theorems supplied by the user, it asks for criteria that an automated reasoning program can use to find some theorems in a field that must be evaluated by theorists of the field as new and interesting theorems. The significance of solving the problem is obvious because an automated reasoning program satisfying the requirement can provide great assistance for scientists in various fields [2].

To solve the ATF problem, a forward reasoning approach based on strong relevant logic [1, 2] and its systematic methodology [8] have been proposed. To verify the effectiveness of the approach, we used the methodology to perform a case study of ATF in axiomatic set theory, and the result of that case study shows that the forward reasoning approach based on strong relevant logic is hopeful to solve the ATF problem [8].

This paper presents a case study of automated theorem finding in graph theory [7] to show the generality of the proposed methodology, and presents a future direction for ATF based on the methodology.



James J. (Jong Hyuk) Park et al. (eds.), *Advanced Multimedia and Ubiquitous Engineering*,
Lecture Notes in Electrical Engineering 352, DOI: 10.1007/978-3-662-47487-7_4

## 2 Basic Notions and Notations

A formal logic system $L$ is an ordered pair $(F(L), \vdash_L)$ where $F(L)$ is the set of well formed formulas of $L$, and $\vdash_L$ is the consequence relation of $L$ such that for a set $P$ of formulas and a formula $C$, $P \vdash_L C$ means that within the framework of $L$ taking $P$ as premises we can obtain $C$ as a valid conclusion. $Th(L)$ is the set of logical theorems of $L$ such that $\phi \vdash_L T$ holds for any $T \in Th(L)$. According to the representation of the consequence relation of a logic, the logic can be represented as a Hilbert style system, Gentzen sequent calculus system, Gentzen natural deduction system, and so on [3].

Let $(F(L), \vdash_L)$ be a formal logic system and $P \subseteq F(L)$ be a non-empty set of sentences. *A* formal theory with premises $P$ based on $L$, called an $L$-theory with premises $P$ and denoted by $T_L(P)$, is defined as $T_L(P) =_{df} Th(L) \cup Th^e{}_L(P)$ where $Th^e{}_L(P) =_{df} \{A | P \vdash_L A \text{ and } A \notin Th(L)\}$, $Th(L)$ and $Th^e{}_L(P)$ are called the logical part and the empirical part of the formal theory, respectively, and any element of $Th^e{}_L(P)$ is called an empirical theorem of the formal theory [3].

Based on the definition above, the problem of ATF can be said as "for any given premises $P$, how to construct a meaningful formal theory $T_L(P)$ and then find new and interesting theorems in $Th^e{}_L(P)$ automatically" [3].

The notion of predicate abstract level [8] is defined as follows: (1) Let $pal(X) = k$ denote that an abstract level of a predicate $X$ is $k$ where $k$ is a natural number, (2) $pal(X) = 1$ if $X$ is the most primitive predicate in a target field, (3) $pal(X) = max(pal(Y_1), pal(Y_2), ..., pal(Y_n))+1$ if a predicate $X$ is defined by other predicates $Y_1$, $Y_2$, ..., $Y_n$ in the target field where $n$ is a natural number. A predicate $X$ is called $k$-level predicate, if $pal(X) = k$. If $pal(X) < pal(Y)$, then the abstract level of predicate $X$ is lower than $Y$, and $Y$ is higher than $X$.

The notion of function abstract level [8] is defined as follows: (1) Let $fal(f) = k$ denote that an abstract level of a function $f$ is $k$ where $k$ is a natural number, (2) $fal(f) = 1$ if $f$ is the most primitive function in the target field, (3) $fal(f) = \max(fal(g_1), fal(g_2), ..., fal(g_n)) + 1$ if a function $f$ is defined by other functions $g_1$, $g_2$, ..., $g_n$ in the target field where n is a natural number. A function $f$ is k-level function, if $fal(f) = k$. If $fal(f) < fal(g)$, we call the abstract level of function $f$ is lower than $g$, and $g$ is higher than $f$.

The notion of abstract level of a formula [8] is defined as follows: (1) $lfal(f) = (k, m)$ denotes that an abstract level of a formula $A$ where $k = pal(A)$ and $m = fal(A)$, (2) $pal(A) = \max(pal(Q_1), pal(Q_2), ..., pal(Q_n))$ where $Q_i$ is a predicate and occurs in $A$ ($1 \le i \le n$), or $pal(A) = 0$, if there is not any predicate in $A$, (3) $fal(A) = \max(fal(g_1), fal(g_2), ..., fal(g_n))$ where $g_i$ is a function and occurs in $A$ ($1 \le i \le n$), or $fal(A) = 0$, if there is not any function in $A$. A formula $A$ is $(k, m)$-level formula, if $lfal(A) = (k, m)$.

$(k, m)$-fragment of premises $P$, denoted by $P(k, m)$, is a set of all formulas in $P$ that consists of only $(j, n)$-level formulas where $m$, $n$, $j$ and $k$ are natural number ($0 \le j \le k$ and $0 \le n \le m$) [8].

## 3 A Systematic Methodology for ATF with Forward Reasoning

The systematic methodology for ATF consists of five phases [8]. Phase 1 is to prepare logical fragments [5] of strong relevant logic for various empirical theories. The prepared logic fragments are independent from any target field, therefore they can be reused for ATF in different fields. Phase 2 is to prepare empirical premises of the target theory and draw up a plan to use collected empirical theorems. In detail, we prepare $(k, m)$-fragment of collected empirical premises in the target field to define a semi-lattice. A set of the prepared fragments and inclusion relation on the set is a partial order set, and is a finite semi-lattice. Moreover, a set of formal theories with the fragments and inclusion relation on the set is also a partial order set, and is also a finite semi-lattice. Partial order of the set of the prepared fragments can be used for a plan to reason out fragments of formal theories with collected empirical premises. According to the partial order, we can systematically do ATF from simple theorems to complex theorems.

Phase 3 to phase 5 are performed repeatedly until deducing all fragments of formal theory that have been planned in phase 2. In this methodology, *loop* means doing phase 3 to phase 5 at once. We use one of $(k, m)$-fragment of collected empirical premises to perform phase 3 to phase 5 in one loop. In detail, we firstly use lowest level fragment of premises to reason out empirical theorems. Then, we enter into phase 4 to abstract deduced empirical theorems, and then enter into phase 5 to find new and interesting theorems from empirical theorems. After that, we go back to the phase 3, and use the next level fragment of premises and theorems obtained in last loop as premises of this loop. Then, we enter into phase 4 and phase 5 to abstract theorems and find interesting theorems again. We repeat the loops until all of the $(k, m)$-fragment of collected empirical premises have been used.

## 4 Case Study of ATF in Graph Theory

The purpose of the case study is to show the generality of our methodology. We chose graph theory as the field of ATF in the case study, because graph theory can be established above axiomatic set theory. We have performed a case study of ATF [8] in axiomatic set theory by using the proposed methodology. If we can also perform ATF in graph theory by using the methodology, it means that we can also do ATF in other mathematical fields by using our methodology, because almost all of mathematical fields can be established above axiomatic set theory.

We performed the case study according to our methodology. Phase 1 is to prepare logical fragments of strong relevant logic. We prepared logical fragments in the case study of axiomatic set theory [8] and those logic fragments can be reused in the case study. Phase 2 is to prepare the empirical premises of graph theory. Diestel [7] recorded the definitions of graph theory in his book. In the case study, we chose 21 definitions in Diestel's book as empirical premises of the case study and formalized them based on the predicates and functions of NBG set theory [11].

The definitions of graph theory formalized by us are shown as follows.

1. Definition of *graph*
   $G\ (V, E) = \langle V, E \rangle$
2. Definition of *empty graph*
   $0 = G\ (0, 0)$
3. Definition of *incident edge*
   $\forall x \forall y \forall v\ (\ (\{x, y\} \in E) \wedge (v \in \{x, y\}) \Leftrightarrow (\{x, y\} = incidentedge\ (v)))$
4. Definition of *loop*
   $\forall x\ (loop\ (x) = \{x\})$
5. Definition of *isomorphism*
   $\forall w \forall v \forall v' \forall e \forall e'\ ((G\ (v, e) = G\ (v', e')) \Leftrightarrow (w \in e \Rightarrow isomorphism\ (w) \in e')$
6. Definition of $\cap_g$
   $\forall x \forall x' \forall y \forall y'\ (G\ (x, y) \cap_g G\ (x', y') = G\ (x \cap x', y \cap y'))$
7. Definition of $\cup_g$
   $\forall x \forall x' \forall y \forall y'\ (G\ (x, y) \cup_g G\ (x', y') = G\ (x \cup x', y \cup y'))$
8. Definition of *subgraph*
   $\forall x \forall x' \forall y \forall y'\ (Sub\ (G\ (x, y), G'\ (x', y')) \Leftrightarrow ((x \subseteq x') \wedge (y \subseteq y') \wedge (x' \subseteq V) \wedge (y' \subseteq E)))$
9. Definition of *induced subgraph*
   $\forall v \forall x \forall x' \forall y \forall y'\ (InducedSub\ (G\ (x, y), G'\ (x', y')) \Leftrightarrow Sub\ (G\ (x, y), G'\ (x', y')) \wedge ((v \in (x \cap x')) \Rightarrow (incidentedge\ (v) \in y)))$
10. Definition of *super graph*
    $\forall x \forall x' \forall y \forall y'\ (Sup\ (G'\ (x', y'), G\ (x, y)) \Leftrightarrow Sub\ (G\ (x, y), G'\ (x', y'))$
11. Definition of *simple graph*
    $\forall x \forall m \forall n \forall v \forall e\ (SimpleGraph\ (G\ (v, e)) \Leftrightarrow (v \subseteq V) \wedge (e \subseteq E) \wedge ((x \in v) \Rightarrow \neg(loop\ (x) \in e)) \wedge ((m \in e) \wedge (n \in e) \Rightarrow \neg(m = n)))$
12. Definition of *adjacent*
    $\forall x \forall y\ (Adjacent\ (x, y) \Leftrightarrow (\{x, y\} \in E))$
13. Definition of *complete graph*
    $\forall x \forall y \forall v \forall e\ (CompleteGraph\ (G\ (v, e)) \Leftrightarrow SimpleGraph\ (G\ (v, e)) \wedge ((x \in v) \wedge (y \in v) \Rightarrow Adjacent\ (x, y)))$
14. Definition of *disjoint*
    $\forall x \forall x' \forall y \forall y'\ (Disjoint\ (G\ (x, y), G\ (x', y')) \Leftrightarrow (G\ (x, y) \cap_g G\ (x', y') = 0))$
15. Definition of –
    $\forall x \forall y \forall u\ (G\ (x, y) - u = G\ (\sim (x \cap u), \sim (y \cap incidentedge\ (u)))$
16. Definition of *connected graph*
    $\forall u \forall x \forall y \forall z \forall v \forall e\ (ConnectedGraph\ (G\ (v, e)) \Leftrightarrow ((G\ (u, x) \cup_g G\ (y, z) = G\ (v, e)) \Rightarrow \neg Disjoint\ (G\ (u, x), G\ (y, z)))$
17. Definition of *path*
    $\forall e \forall v \forall x \forall m \forall n \forall p((path\ (v, e) = G\ (v, e)) \Leftrightarrow (ConnectedGraph\ (G\ (v, e)) \wedge ((m \in e) \wedge (n \in e) \wedge (p \in e) \wedge \neg(m = n) \wedge \neg(n = p) \wedge \neg(m = p) \wedge (x \in m) \wedge (x \in n)) \Rightarrow \neg(x \in p)))$

18. Definition of *cycle*
$\forall v \forall e \forall x \forall m \exists n\ ((cycle(v, e) = G\ (v, e)) <=> ((path\ (v, e) = G\ (v, e)) \wedge ((m \in e) \wedge (x \in m) => (x \in n) \wedge (n \in e) \wedge \neg (m=n)))$
19. Definition of *connectivity*
$\forall x \forall y \forall e \forall v\ (Connect\ (x, y) <=> ((x \in v) \wedge (y \in v) => (\{v\} \in path(v, e))))$
20. Definition of *forest*
$\forall v \forall v' \forall e \forall e'\ (Forest\ (G(v, e)) <=> \neg Sub\ (cycle\ (v', e'), G\ (v, e)))$
21. Definition of *tree*
$\forall v \forall e\ (Tree\ (G\ (v, e)) <=> ConnectedGraph\ (G\ (v, e)) \wedge Forest\ (G\ (v, e)))$

Then, we defined a semi-lattice of abstract level fragments of formalized empirical premises according to the proposed methodology. In detail, first we summarized all of the predicate abstract levels in the formalized definitions: 2-level predicate: *Adjacent* (abstract from $\in$), *Connect* (from $\in$); 3-level predicate: *Sub* (from $\subseteq$); 4-level predicate: *Forest* (from *Sub*), *InducedSub* (from *Sub*, $\in$), *Sup* (from *Sub*), *SimpleGraph* (from $\in$, $\subseteq$, =), *Disjoint* (from =); 5-level predicate: *CompleteGraph* (from *SimpleGraph*, $\in$, *Adjacent*), *ConnectedGraph* (from =, *Disjoint*); 6-level predicate: *Tree* (from *ConnectedGraph*, *Forest*). Second, we summarized all of the function abstract levels in the formalized definition: 2-level function: *incidentedge* (abstract from *unordered pair*); 3-level function: *loop* (from *singleton*); 4-level function: *G* (from *ordered pair*); 5-level function: *path* (from *G*), $\cap_g$ (from $\cap$, *G*), $\cup_g$ (from $\cup$, *G*), *isomorphism* (from *G*), $-$ (from G, ~, $\cap$, incidentedge); 6-level function: *cycle* (from *G*, *path*). Third, we summarized all of the abstract levels of formalized definitions as shown in Table 1. Finally, we defined the semi-lattice of abstract level fragments of empirical premises in graph theory as shown in Fig. 1 by using the methodology. In this case study NBG set theory is seen as the minimum element of the semi-lattice, which is different from the last case study of axiomatic set theory [8].

**Table 1.** The abstract level of definitions in graph theory

| Abstract Level | Definition |
|---|---|
| (2, 1) | Definition of *adjacent* |
| (2, 5) | Definition of *connectivity* |
| (3, 2) | Definition of *incident edge* |
| (3, 3) | Definition of *loop* |
| (3, 4) | Definition of *graph*, *empty graph*, *subgraph* |
| (3, 5) | Definition of *isomorphism*, $\cap_g$, $\cup_g$, $-$ |
| (3, 6) | Definition of *cycle* |
| (4, 4) | Definition of *super graph*, *simple graph*, *induced subgraph* |
| (4, 5) | Definition of *disjoint* |
| (4, 6) | Definition of *forest* |
| (5, 4) | Definition of *complete graph* |
| (5, 5) | Definition of *path*, *connected graph* |
| (6, 4) | Definition of *tree* |

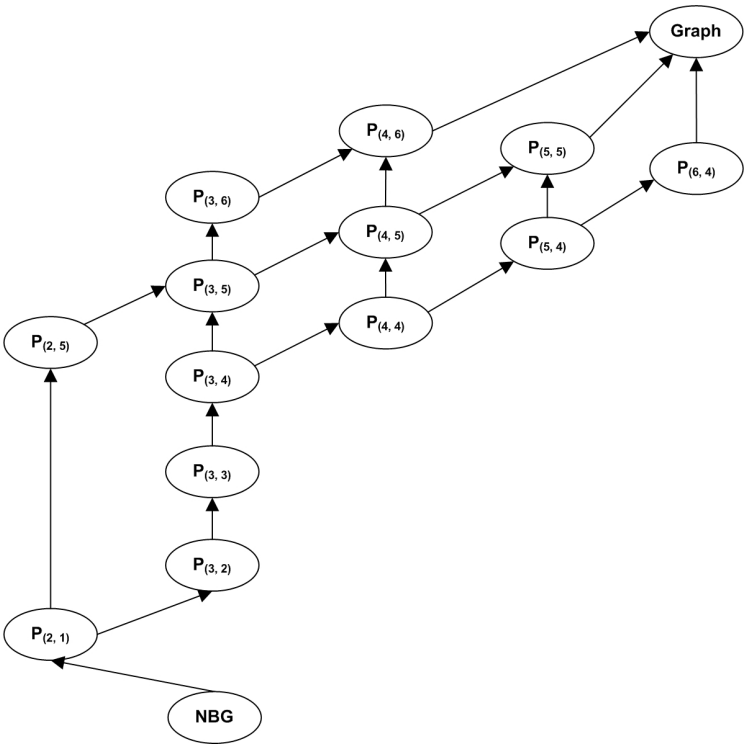


**Fig. 1.** The defined semi-lattice of graph theory based on NBG set theory

From Phase 3 to Phase 5, we performed deduction, abstraction, and finding of theorems. We used the general reasoning engine FreeEnCal [5] as the tool. In detail, we used the prepared logic fragments [8] from small to large according to the defined semi-lattice of strong relevant logic [8] as logic premises, and used ($k$, $m$)-fragments of premises of graph theory from lower abstract level to higher abstract level as empirical premises, we also used obtained empirical theorems of NBG set theory in last case study [8] as empirical premises to perform ATF. Then, we used filtering method to remove uninteresting theorems. We showed the results of the case study in Table 2.

The case study shows that our methodology holds generality. By using our methodology, we defined the ($k$, $m$)-fragments of empirical premises of axiomatic set theory and extended them to graph theory well. Besides, it is sure that the case study of ATF in graph theory were performed systematically in each phase. All of those empirical theorems are obtained by using forward reasoning method and our filtering method can filter most of uninteresting theorems based on syntax, more than 90% empirical theorems reasoned out by all of the five prepared logical fragments can be removed as uninteresting theorems automatically such that the scientists can find interesting theorems from the filtered results based on semantics by acceptable time.

**Table 2.** The results of the case study

| Used logic fragments | Obtained empirical theorems | Core empirical theorems | Filtered results |
|---|---|---|---|
| $Th^{(\Rightarrow, 2)}(EeQ)$ | 1138 | 102 | 83 |
| $Th^{(\Rightarrow, 3)}(EeQ)$ | 1662 | 133 | 93 |
| $Th^{(\Rightarrow, 2, \neg, 1)}(EenQ)$ | 1139 | 103 | 84 |
| $Th^{(\Rightarrow, 3, \neg, 1)}(EenQ)$ | 2885 | 288 | 216 |
| $Th^{(\Rightarrow, 2, \neg, 1, \wedge, 1)}(EcQ)$ | 1216 | 122 | 97 |

## 5 A Future Direction for ATF

Cheng has proposed a semi-lattice model of formal theories [4], which can support forward reasoning approach based on strong relevant logic to do ATF in multi-fields. The core of the model is strong relevant logic (SRL) and axiomatic set theory is seen as the minimum element in the semi-lattice, and other formal theories can be established above the axiomatic set theory as shown in Fig. 2.

To do ATF based on Cheng's semi-lattice model [4] of formal theories by using our methodology is a future direction for ATF, because it holds generality. Our methodology can support Cheng's semi-lattice model and provide a method to establish the semi-lattice of formal theories, that is we define the ($k$, $m$)-fragment of empirical premises of axiomatic set theory and we extend them to other mathematical fields. The case study of graph theory which we have presented in Section 4 shows our methodology is effective to support Cheng's semi-lattice model of formal theories. We consider finding process of theorems must include some concept/notion abstraction processes in other mathematical fields, so we can conclude that our methodology is suitable for ATF in other mathematical fields, such as group theory, lattice theory and number theory.

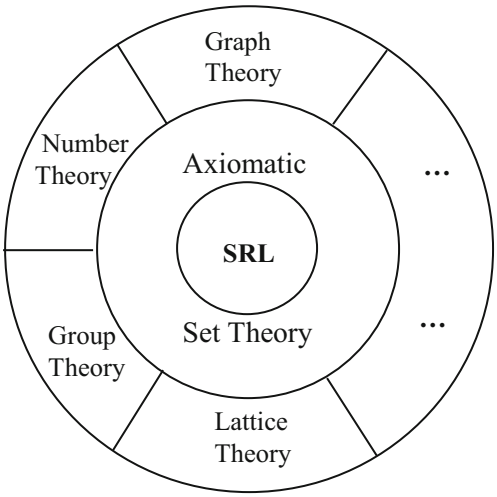


**Fig. 2.** The semi-lattice model of formal theories

## 6 Concluding Remarks

We have presented a case study of ATF in graph theory, and showed the generality of our proposed methodology through the case study. We have shown a future direction for ATF through the case study. We will do ATF in other fields like lattice theory, group theory, and number theory in future according to Cheng's semi-lattice model of formal theories.

## References

1. Cheng, J.: A Relevant Logic Approach to Automated Theorem Finding. In: The Workshop on Automated Theorem Proving attached to International Symposium on Fifth Generation Computer Systems, pp. 8–15 (1994)
2. Cheng, J.: Entailment Calculus as the Logical Basis of Automated Theorem Finding in Scientific Discovery. In: Systematic Methods of Scientific Discovery: Papers from the 1995 Spring Symposium, pp. 105–110. AAAI Press - American Association for Artificial Intelligence (1995)
3. Cheng, J.: A Strong Relevant Logic Model of Epistemic Processes in Scientific Discovery. In: Information Modelling and Knowledge Bases XI. Frontiers in Artificial Intelligence and Applications, pp. 136–159. IOS Press (2000)
4. Cheng, J.: A Semilattice Model for the Theory Grid. In: Proc. 3rd International Conference on Semantics, Knowledge and Grid, pp. 152–157. IEEE Computer Society (2007)
5. Cheng, J., Nara, S., Goto, Y.: FreeEnCal: A Forward Reasoning Engine with General-Purpose. In: Apolloni, B., Howlett, R.J., Jain, L. (eds.) KES 2007, Part II. LNCS (LNAI), vol. 4693, pp. 444–452. Springer, Heidelberg (2007)
6. Colton, S., Meier, A., Sorge, V., McCasland, R.: Automatic Generation of Classification Theorems for Finite Algebras. In: Basin, D., Rusinowitch, M. (eds.) IJCAR 2004. LNCS (LNAI), vol. 3097, pp. 400–414. Springer, Heidelberg (2004)
7. Diestel, R.: Graph Theory. Springer, Heidelberg (2000)
8. Gao, H., Goto, Y., Cheng, J.: A Systematic Methodology for Automated Theorem Finding. Theoretical Computer Science 554, 2–21 (2014)
9. Gao, H., Goto, Y., Cheng, J.: Research on Automated Theorem Finding: Current State and Future Directions. In: Park, J.J(J.H.), Pan, Y., Kim, C.-S., Yang, Y. (eds.) Future Information Technology. LNEE, vol. 309, pp. 105–110. Springer, Heidelberg (2014)
10. McCasland, R., Bundy, A., Autexier, S.: Automated Discovery of Inductive Theorems. Journal of Studies in Logic, Grammar and Rhetoric 10(23), 135–149 (2007)
11. Quaife, A.: Automated Development of Fundamental Mathematical Theories. Kluwer Academic (1992)
12. Recio, T., Velez, M.Z.: Automatic Discovery of Theorems in Elementary Geometry. Journal of Automated Reasoning 23(1), 63–82 (1999)
13. Tang, P., Lin, F.: Discovering Theorems in Game Theory: Two-Person Games with Unique Pure Nash Equilibrium Payoffs. Artificial Intelligence 175(14), 2010–2020 (2011)
14. Wos, L.: Automated Reasoning: 33 Basic Research Problem. Prentic-Hall (1988)
15. Wos, L.: The Problem of Automated Theorem Finding. Journal of Automated Reasoning 10(1), 137–138 (1993)