Resolution in Solving Graph Problems

Kailiang Ji

Resolution in Solving Graph Problems

Kailiang Ji ⋆

LRI, Université Paris Sud
kailiang.ji@lri.fr

Abstract. Resolution is a proof-search method on proving satisfiability problems. Various refinements have been proposed to improve the efficiency of this method. However, when we try to prove some graph properties, none of the refinements have an efficiency comparable with traditional graph traversal algorithms. In this paper we propose a way of solving some graph traversal problems with resolution. And we design some simplification rules to make the proof-search algorithm work more efficiently on such problems.

1 Introduction

Since the introduction of Resolution [11], many refinements have been proposed to increase the efficiency of this method, by avoiding redundancies. A first refinement, *hyper-resolution*, has been introduced by Robinson himself the same year as Resolution [10]. More recently *ordered resolution* [9, 12], (*polarized*) *resolution modulo* (PRM) [5, 6], and finally *ordered polarized resolution modulo* (OPRM) [1] introduced more restrictions. However, as we shall see, these kind of refinements are still redundant.

In order to address the question of the redundancy of proof search methods, we encode graph problems, e.g. accessibility or cycle detection, as Resolution problem, and we compare two ways to solve these problems: by using a proof-search method and by a direct graph traversal algorithm. If the proof-search method simulates graph traversal step by step, and in particular never visits twice the same part of the graph, we can say that it avoids redundancies. Otherwise, this helps us analyze and eliminate the redundancies of the method, by analyzing why the method visits twice the same part of the graph.

The two graph problems can be expressed by predicate formulae with class variables (monadic second-order logic) [4, 7]. For instance, the cycle detection problem can be expressed as $\exists Y(s_1 \in Y \wedge \forall x(x \in Y \Rightarrow \exists x'(\mathsf{edge}(x, x') \wedge x' \in Y)))$. The satisfiability of this formula can be proved by reducing it to *effectively propositional* case [8], where the sub-formula $\forall x A$ is replaced by $A(s_1/x) \wedge \cdots \wedge A(s_n/x)$, and $\exists x A$ by $A(s_1/x) \vee \cdots \vee A(s_n/x)$, in which $s_1, ..., s_n$ are the constants for all the vertices of a graph. By representing the theory of the graph as a set of rewrite rules [7], these problems can be proved by some off-the-shelf automated

⋆ This work is supported by the ANR-NSFC project LOCALI (NSFC 61161130530 and ANR 11 IS02 002 01)

theorem provers, such as iProver Modulo [2]. As these problems can be expressed with temporal formulae [3], they can also be solved by model checking tools. In this paper, a propositional encoding of these two problems is given. To reduce the search space and avoid redundant resolution steps, we add a selection function and a new subsumption rule. This method works for encoding of several graph problems. Its generality remains to be investigated.

The paper is organized as follows. Section 2 describes the theorem proving system PRM. In Section 3, some basic definitions for the expressing of graph problems are presented. Section 4 and 5 presents the encoding of cycle detection and accessibility respectively. In Section 6, some simplification rules are defined. Finally, an implementation result is presented.

2 Polarized Resolution Modulo

In Polarized Resolution Modulo (see Figure 1), clauses are divided into two sets: one-way clauses (or theory clauses) and ordinary clauses. Each one-way clause has a selected literal and resolution is only permitted between two ordinary clauses, or a one-way clause and an ordinary clause, provided the resolved literal is the selected one (the one underlined later) in the one-way clause. In the rules of Figure 1, P and Q are literals, C and D denote a set of literals. σ is a substitution function, which is equal to the maximal general unifier (mgu) of P and Q. $\mathcal{R}$ is a set of one-way clauses that are under consideration.

$$\textbf{Resolution}\ \frac{P \vee C \qquad Q^{\perp} \vee D}{\sigma(C \vee D)} \qquad \textbf{Factoring}\ \frac{P \vee Q \vee C}{\sigma(P, C)}$$

$$\textbf{Ext.Narr.}\ \frac{P \vee C}{\sigma(D \vee C)}\ \text{if } \underline{Q^{\perp}} \vee D \text{ is a one-way clause of } \mathcal{R}$$

$$\textbf{Ext.Narr.}\ \frac{P^{\perp} \vee C}{\sigma(D \vee C)}\ \text{if } \underline{Q} \vee D \text{ is a one-way clause of } \mathcal{R}$$

Fig. 1. Polarized Resolution Modulo

Proving the completeness of the rules in Figure 1 requires to prove a cut elimination lemma [6, 5] for Polarized Deduction Modulo, the deduction system with a set of rewrite rules, containing for each one-way clause $\underline{P^{\perp}} \vee C$ the rule $P \rightarrow^{-} C$ and for each one-way clause $\underline{P} \vee C$ the rule $P \rightarrow^{+} C^{\perp}$.

Like in OPRM, in this paper we define a selection function to select literals in an ordinary clause which have the priority to be resolved and add the selection function to PRM.

Note that when applying a **Resolution** rule between an ordinary clause and a one-way clause, we are in fact using an **Extended Narrowing** rule on this ordinary clause. We write $\Gamma \mapsto_{\mathcal{R}} C$ if C can be derived from the set of clauses Γ by applying finitely many inference rules of PRM.

3 Basic Definitions

In this paper, we consider a propositional language which contains two atomic propositions B_i and W_i for each natural number. We denote a graph as $G = \langle V, E\rangle$, where V is a set of vertices enumerated by natural numbers, E is the set of directed edges in the graph. The sequence of vertices $l = s_0, ..., s_k$ is a *walk* if and only if $\forall 0 \leq i < k$, $(s_i, s_{i+1}) \in E$. The walk l is *closed* if and only if $\exists 0 \leq j \leq k$ such that $s_k = s_j$. The walk l is *blocked* if and only if s_k has no successors. The method we proposed is inspired by graph traversal algorithms.

Definition 1 (Black literal, white literal). *Let G be a graph and $\{s_1, ..., s_n\}$ be the set of all the vertices in G. For any $1 \leq i \leq n$, the literal B_i is called a* black literal *and the literal W_i is called a* white literal*.*

Intuitively, the black literals denote the vertices that have already been visited, while the white literals denote the non-visited ones.

Definition 2 (Original clause, traversal clause, success clause). *Let G be a graph and $\{s_1, ..., s_n\}$ the set of vertices in G. For each graph traversal problem starting from s_i $(1 \leq i \leq n)$, the clause of the form $B_i \vee W_1 \vee \cdots \vee W_n$ is called an* original clause $(\mathsf{OC}(s_i, G))$*. A clause with only white and black literals is called a* traversal clause*. Let C be a traversal clause, if there is no i, such that both B_i and W_i are in C, then C is called a* success clause*.*

Among the three kinds of clauses, the original clause is related to the starting point of the graph traversal algorithm, the traversal clause is the current state of the traveling, and the success clause denotes that a solution is derived. Trivially, the original clauses and success clauses are also traversal clauses.

4 Closed-walk Detection

In this section, we present a strategy of checking whether there exists a closed walk starting from a given vertex. For a graph, each edge is represented as a rewrite rule, and the initial situation is denoted by the original clause.

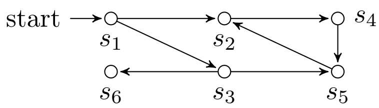
E-coloring rule Let G be a graph and $V = \{s_1, ..., s_n\}$ be the set of vertices in G. For each pair of vertices $\langle s_i, s_j\rangle$ in V, if there exists an edge from s_i to s_j, then we formalize this edge as an *E-coloring rewrite rule*

$$W_i \hookrightarrow B_j.$$

The corresponding one-way clause of this rewrite rule is $\underline{W_i^\perp} \vee B_j$ (called *E-coloring clause*). The set of all the E-coloring clauses for G is denoted as $\mathsf{EC}(G)$.

Resolution for closed-walk detection Let G be a graph and s be a vertex of G, then the the problem of checking *whether, starting from s, there exists a closed walk* can be encoded as the set of clauses $\{\mathsf{OC}(s, G)\} \cup \mathsf{EC}(G)$. By applying resolution rules among these clauses, a success clause can be derived if and only if there exists a closed walk starting from s.

Example 1. Consider the following graph



We prove that there exists a closed walk starting from s_1. For this problem, the original clause is $B_1 \vee W_1 \vee W_2 \vee W_3 \vee W_4 \vee W_5 \vee W_6$ and the set of E-coloring clauses for this graph are

$$\underline{W_1^{\perp}} \vee B_2,\ \ \underline{W_1^{\perp}} \vee B_3,\ \ \underline{W_2^{\perp}} \vee B_4,\ \ \underline{W_3^{\perp}} \vee B_5,\ \ \underline{W_3^{\perp}} \vee B_6,\ \ \underline{W_4^{\perp}} \vee B_5,\ \ \underline{W_5^{\perp}} \vee B_2.$$

The resolution steps are presented in the following tree from top to bottom

$$\begin{array}{c}
B_1 \vee W_1 \vee W_2 \vee W_3 \vee W_4 \vee W_5 \vee W_6 \quad \underline{W_1^{\perp}} \vee B_2 \\
\hline
B_1 \vee B_2 \vee W_2 \vee W_3 \vee W_4 \vee W_5 \vee W_6 \quad \underline{W_2^{\perp}} \vee B_4 \\
\hline
B_1 \vee B_2 \vee B_4 \vee W_3 \vee W_4 \vee W_5 \vee W_6 \quad \underline{W_4^{\perp}} \vee B_5 \\
\hline
B_1 \vee B_2 \vee B_4 \vee W_3 \vee B_5 \vee W_5 \vee W_6 \quad \underline{W_5^{\perp}} \vee B_2 \\
\hline
B_1 \vee B_2 \vee B_4 \vee W_3 \vee B_5 \vee W_6
\end{array}$$

The clause $B_1 \vee B_2 \vee B_4 \vee W_3 \vee B_5 \vee W_6$ is a success clause. Thus, there exists a closed walk starting from s_1.

Theorem 1. *Let G be a graph and s be a vertex of G. Starting from s, there exists a closed walk if and only if starting from $\{\mathsf{OC}(s, G)\} \cup \mathsf{EC}(G)$, a success clause can be derived.*

5 Blocked-walk Detection

In this section, a method on testing whether, starting from a vertex, there exists a blocked walk or not is given. In this method, the set of edges starting from the same vertex are represented as a rewrite rule.

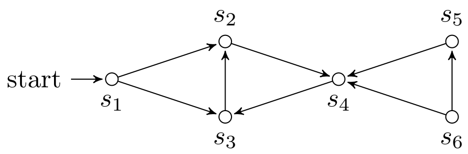
A-coloring rule Let G be a graph and $V = \{s_1, ..., s_n\}$ the set of vertices in G. For each s_i in V, assume that starting from s_i, there are edges to $s_{i_1}, ..., s_{i_j}$, then we formalize such set of edges as an *A-coloring rewrite rule*

$$W_i \hookrightarrow B_{i_1} \vee \cdots \vee B_{i_j}.$$

The one-way clause of this rewrite rule is $\underline{W_i^{\perp}} \vee B_{i_1} \vee \cdots \vee B_{i_j}$ (called *A-coloring clause*). The set of all the A-coloring clauses of G is denoted as $\mathsf{AC}(G)$.

Resolution for blocked-walk detection Let G be a graph and s be a vertex of G, then the problem of checking that *starting from s, whether there exists a blocked walk* can be encoded as the set of clauses $\{\mathsf{OC}(s,G)\} \cup \mathsf{AC}(G)$. By applying resolution rules among these clauses, a success clause can be derived if and only if *there is no blocked walk* starting from s.

Example 2. Consider the graph



and check whether there exists a blocked walk starting from s_1. For this problem, the original clause is $B_1 \lor W_1 \lor W_2 \lor W_3 \lor W_4 \lor W_5 \lor W_6$ and the set of A-coloring clauses for this graph are

$$\underline{W_1^{\perp}} \lor B_2 \lor B_3, \quad \underline{W_2^{\perp}} \lor B_4, \quad \underline{W_3^{\perp}} \lor B_2, \quad \underline{W_4^{\perp}} \lor B_3, \quad \underline{W_5^{\perp}} \lor B_4, \quad \underline{W_6^{\perp}} \lor B_4.$$

The resolution steps are presented in the following tree top-down

$$\dfrac{\dfrac{\dfrac{\dfrac{B_1 \lor W_1 \lor W_2 \lor W_3 \lor W_4 \lor W_5 \lor W_6 \quad \underline{W_1^{\perp}} \lor B_2 \lor B_3}{B_1 \lor B_2 \lor B_3 \lor W_2 \lor W_3 \lor W_4 \lor W_5 \lor W_6 \quad \underline{W_2^{\perp}} \lor B_4}}{B_1 \lor B_2 \lor B_3 \lor B_4 \lor W_3 \lor W_4 \lor W_5 \lor W_6 \quad \underline{W_3^{\perp}} \lor B_2}}{B_1 \lor B_2 \lor B_3 \lor B_4 \lor W_4 \lor W_5 \lor W_6 \quad \underline{W_4^{\perp}} \lor B_3}}{B_1 \lor B_2 \lor B_3 \lor B_4 \lor W_5 \lor W_6}$$

The clause $B_1 \lor B_2 \lor B_3 \lor B_4 \lor W_5 \lor W_6$ is a success clause. Thus, there is no blocked walk starting from s_1.

Theorem 2. *Let G be a graph and s be a vertex of G. Starting from s, there is no blocked walk if and only if, starting from $\{\mathsf{OC}(s,G)\} \cup \mathsf{AC}(G)$, a success clause can be derived.*

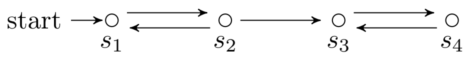
6 Simplification Rules

Traditional automatic theorem proving methods are only practical for graphs of relatively small size. In this section, the reason why the resolution method is not as efficient as graph traversal algorithms is analyzed. Moreover, some strategies are designed to address such problems in proof-search algorithms.

6.1 Selection Function

First we show that the number of resolution steps strongly depend on the literals that are selected. More precisely, the number of literals that are selected will also affect the number of resolution steps. Then a selection function is given.

Example 3. For the graph



we prove the property: *starting from s_1, there exists a closed walk.* The original clause is $B_1 \vee W_1 \vee W_2 \vee W_3 \vee W_4$ and the E-coloring clauses of the graph are

$$\underline{W_1^\perp} \vee B_2, \quad \underline{W_2^\perp} \vee B_1, \quad \underline{W_2^\perp} \vee B_3, \quad \underline{W_3^\perp} \vee B_4, \quad \underline{W_4^\perp} \vee B_3.$$

Starting from the original clause, we can apply resolution as follows: First, apply resolution with E-coloring clause $\underline{W_1^\perp} \vee B_2$, which yields

$$B_1 \vee B_2 \vee W_2 \vee W_3 \vee W_4. \tag{1}$$

Then for (1), apply resolution with E-coloring clause $\underline{W_2^\perp} \vee B_1$, which yields

$$B_1 \vee B_2 \vee W_3 \vee W_4. \tag{2}$$

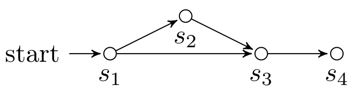
Clause (2) is a success clause. However, from (1), if we apply resolution with another E-coloring clause, more steps are needed to get a success clause.

The instinctive idea from Example 3 is similar to graph traversal algorithm. In a traversal clause, if there exists a pair of literals B_i and W_i, then the strategy of selecting W_i to have priority in applying resolution rules may have less resoution steps to get a success clause.

Definition 3 (Grey literal). *Let C be a traversal clause. For the pair of white literals and black literals $\langle W_i, B_i \rangle$, if both W_i and B_i are in C, then W_i is called a* grey literal *of C. The set of grey literals of C is defined as follows:*

$$\mathsf{grey}(C) = \{W_i \mid B_i \in C \ \&\ W_i \in C\}$$

Example 4. For the graph



we prove the property: *starting from s_1, there is no blocked walk.* The original clause is $B_1 \vee W_1 \vee W_2 \vee W_3 \vee W_4$ and the A-coloring clauses of the graph are

$$\underline{W_1^\perp} \vee B_2 \vee B_3, \quad \underline{W_2^\perp} \vee B_3, \quad \underline{W_3^\perp} \vee B_4$$

For the original clause, apply resolution with A-coloring clause $\underline{W_1^\perp} \vee B_2 \vee B_3$, which yields

$$B_1 \vee B_2 \vee B_3 \vee W_2 \vee W_3 \vee W_4. \tag{3}$$

Then for (3), we can apply resolution rules with A-coloring clauses $\underline{W_2^\perp} \vee B_3$ and $\underline{W_3^\perp} \vee B_4$, and two new traversal clauses are generated:

$$B_1 \vee B_2 \vee B_3 \vee W_3 \vee W_4, \tag{4}$$

$$B_1 \vee B_2 \vee B_3 \vee B_4 \vee W_2 \vee W_4. \tag{5}$$

Then for (4), apply resolution rule with A-coloring clause $\underline{W_3^\perp} \vee B_4$, which yields

$$B_1 \vee B_2 \vee B_3 \vee B_4 \vee W_4, \tag{6}$$

and for this clause, we cannot apply resolution rules any more. For (5), we can apply resolution rule with A-coloring clause $\underline{W_2^\perp} \vee B_3$, and the clause generated is the same as (6). Thus, the clause (5) is redundant.

To avoid generating redundant clauses similar to Example 4, the following selection function is defined.

Definition 4 (Selection function). *For any traversal clause C, the selection function δ is defined as:*

$$\delta(C) = \begin{cases} \mathsf{single}(\mathsf{grey}(C)), & \mathsf{grey}(C) \neq \emptyset \\ C, & \textit{Otherwise} \end{cases}$$

in which single *is a random process to select only one literal from a set of literals.*

Notations The *Polarized Resolution Modulo with δ* is written as PRM_δ. We write $\Gamma \to_{\mathcal{R}}^{\delta} C$ if the clause C can be derived from Γ in the system PRM_δ.

6.2 Elimination Rule

As we will see, selecting literals, which is at the base of Ordered Resolution, PRM, OPRM and this method are not sufficient enough, as we also have to restrict the method at the level of clauses.

Example 5. Reconsider the graph in Example 4, we prove the property: *starting from s_1, there exists a closed walk.* The original clause is $B_1 \vee W_1 \vee W_2 \vee W_3 \vee W_4$ and the E-coloring clauses of the graph are:

$$\underline{W_1^\perp} \vee B_2, \quad \underline{W_1^\perp} \vee B_3, \quad \underline{W_2^\perp} \vee B_3, \quad \underline{W_3^\perp} \vee B_4$$

For the original clause, apply resolution rules with $\underline{W_1^\perp} \vee B_2$ and $\underline{W_1^\perp} \vee B_3$, two new traversal clauses

$$B_1 \vee B_2 \vee W_2 \vee W_3 \vee W_4, \tag{7}$$

$$B_1 \vee B_3 \vee W_2 \vee W_3 \vee W_4 \tag{8}$$

are generated. For (7), apply resolution rule with $\underline{W_2^\perp} \vee B_3$, which yields

$$B_1 \vee B_2 \vee B_3 \vee W_3 \vee W_4. \tag{9}$$

Then for (9), apply resolution rule with $\underline{W_3^\perp} \vee B_4$, which yields

$$B_1 \vee B_2 \vee B_3 \vee B_4 \vee W_4. \tag{10}$$

Resolution rules cannot be applied on (10) any more. Then we can apply resolution rule between (8) and $\underline{W_3^\perp} \vee B_4$, with

$$B_1 \vee B_3 \vee W_2 \vee B_4 \vee W_4 \tag{11}$$

generated, on which the resolution rules cannot be applied neither.

In Example 5, The clause (8) has the same grey literal as (9). Note that no success clause can be derived start from either (8) or (9).

Definition 5 (Path subsumption rule (PSR)). *Let M be a set of A(E)-coloring clauses and C be a traversal clause. If we have $C, M \to_{\mathcal{R}}^{\delta} C_1$ and $C, M \to_{\mathcal{R}}^{\delta} C_2$, in which $\mathsf{grey}(C_1) = \mathsf{grey}(C_2)$, the following rule*

$$\frac{C_1 \quad C_2}{C_i} \ \mathsf{grey}(C_1) = \mathsf{grey}(C_2), i = 1 \ \mathit{or} \ 2$$

can be applied to delete either C_1 or C_2, without breaking the final result.

After each step of applying resolution rules, if we apply PSR on the set of traversal clauses, the clause (8) in Example 5 will be deleted.

Theorem 3 (Completeness). PRM$_\delta$ *with* PSR *is complete.*

7 Implementation

In this section, we talk about the issues during the implementation, and then present the data of experiments.

7.1 Issues in Implementation

Success Clauses In normal resolution based algorithms, the deduction will terminate if (i) an empty clause is generated, meaning the set of original clauses is **Unsat**isfiable or (ii) the resolution rule cannot be applied to derive any more new clauses, in this case the set of original clauses is **Sat**isfiable. However, for the problems in this paper, the derivation should stop when a success clause is derived, which is neither **Sat** nor **Unsat**. To implement our method in automated theorem provers, there may be two ways to deal with the success clauses. *The first way* is to give a set of rewrite rules, and make sure that every success clause can be rewritten into empty clause. For example, we can introduce class variables and treat the atomic propositions B_i and W_i as binary predicates, i.e, replace B_i with $B(s_i, Y)$ and W_i with $W(s_i, Y)$. Thus the success clause $B_1 \vee \cdots \vee B_i \vee W_{i+1} \vee \cdots \vee W_k$ is replaced by $B(s_1, Y) \vee \cdots \vee B(s_i, Y) \vee W(s_{i+1}, Y) \vee \cdots \vee W(s_k, Y)$. To deal with this kind of clause, the following rewrite rules are taken into account.

$$\begin{array}{ll} B(x, add(y, Z)) \hookrightarrow x = y^{\perp} \wedge B(x, Z) & W(x, nil) \hookrightarrow \perp \\ W(x, add(y, Z)) \hookrightarrow x = y \vee W(x, Z) & x = x \hookrightarrow \top \\ \text{for each pair of different vertices } s_i \text{ and } s_j,\ s_i = s_j \hookrightarrow \perp & \end{array}$$

This idea is a variation of the theory in [7]. *The second way* is to add a function to check whether a clause is a success clause or not to the proof-search procedure.

Path Subsumption Rule To make it simple, an empty set G is given in the initial part of the proof-search algorithm, and for the selected traversal clause in U, if the selected grey literal of the traversal clause is in G, then the traversal clause is dead, otherwise, add the selected grey literal to G.

```
Init   : original clause in U, coloring clauses in P
         G = ∅ // G is a set of sets of grey literals
Output: Sat or Unsat
while U ≠ ∅ do
    c = select(U);
    U = U \ c; // remove c from U
    if c is an empty clause or a success clause then
        return Unsat
    end
    g := δ(c); // δ is the literal selection function
    if g ∉ G then
        P = P ∪ {c}; // add c to P
        G = G ∪ {g};
        U = U + generate(c, P);
    end
end
return Sat;
```

Algorithm 1: Proof Search Algorithm

Algorithm The proof-search algorithm with literal selection function and path subsumption rule is in Algorithm 1. In this algorithm, $\mathsf{select}(\mathsf{U})$ selects a clause from U, g is the selected grey literal in c and $\mathsf{generate}(\mathsf{c}, \mathsf{P})$ produces all the clauses by applying an inference rule on c or between c and a clause in P.

7.2 Experimental Evaluation

In the following experiment, the procedure of identifying success clauses, the selection function, and the PSR are embedded into iProver Modulo. The data of the experiments on some randomly generated graphs are illustrated in Table 1.

Table 1. Closed-walk and Blocked-walk Detection

Graph				Result and Time			
Prop	**N(v)**	**N(e)**	**Num**	**Sat**	**Succ**	**PRM$_\delta$**	**PRM$_\delta$ + PSR**
	1.0×10^3	1.0×10^3	100	95	5	25m40s	25m0s
Closed Walk	1.0×10^3	1.5×10^3	100	50	50	1h06m40s	1h02m46s
	1.0×10^3	2.0×10^3	100	23	77	1h09m44s	1h09m46s
	1.0×10^3	2.0×10^3	100	100	0	17m48s	
Blocked Walk	1.0×10^3	3.0×10^3	100	100	0	1h06m28s	
	1.0×10^3	1.0×10^4	100	0	100	24h50m43s	

For the test cases of closed-walk detection, the total time on testing all the 100 graphs did not change much when we take PSR into account. By checking the running time of each graph, we found that *in most cases, PSR was inactive, as most of the vertices did not have the chance to be visited again.* However, *on some special graphs, the running time do reduces much.* On blocked walk detection, the running time grows while there are more edges on graphs, as the number of visited vertices increased.

8 Conclusion and Future Work

In this paper, two graph problems, closed-walk and blocked-walk detection, are considered. The problems are encoded with propositional formulae, and the edges are treated as rewrite rules. Moreover, a selection function and a subsumption rule are designed to address efficiency problems.

Safety and *liveness* are two basic model checking problems [3]. In a program, safety properties specify that "something bad never happens", while liveness assert that "something good will happen eventually". To prove the safety of a system, all the accessible states starting from the initial one should be traversed, which is a kind of blocked-walk detection problem. For liveness, we need to prove that on each infinite path starting from the initial state, there exists a "good" one. This problem can be treated as closed-walk detection. In the future, we will try to address some model checking problems by improving our strategy.

Acknowledgments. I am grateful to Gilles Dowek, for his careful reading and comments.

References

1. Burel, G.: Embedding Deduction Modulo into a Prover. In: Dawar, A., Veith, H. (eds.) CSL 2010. LNCS, vol. 6247, pp. 155–169. Springer Berlin Heidelberg (2010)
2. Burel, G.: Experimenting with Deduction Modulo. In: Sofronie-Stokkermans, V., Bjørner, N. (eds.) CADE 2011. Lecture Notes in Artificial Intelligence, vol. 6803, pp. 162–176. Springer (2011)
3. Clarke, Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge, MA, USA (1999)
4. Courcelle, B.: The Monadic Second-order Logic of Graphs. I. Recognizable Sets of Finite Graphs. Information and computation 85(1), 12–75 (1990)
5. Dowek, G.: Polarized Resolution Modulo. In: Calude, C.S., Sassone, V. (eds.) TCS 2010. IFIP AICT, vol. 323, pp. 182–196. Springer Berlin Heidelberg (2010)
6. Dowek, G., Hardin, T., Kirchner, C.: Theorem Proving Modulo. Journal of Automated Reasoning 31, 33–72 (2003)
7. Dowek, G., Jiang, Y.: Axiomatizing Truth in a Finite Model (2013), `https://who.rocq.inria.fr/Gilles.Dowek/Publi/classes.pdf`, manuscript
8. Navarro-Pérez, J.A.: Encoding and Solving Problems in Effectively Propositional Logic. Ph.D. thesis, The University of Manchester (2007)
9. Reiter, R.: Two Results on Ordering for Resolution with Merging and Linear Format. Journal of the ACM (JACM) 18(4), 630–646 (1971)
10. Robinson, J.A.: Automatic Deduction with Hyper-Resolution. Journal of Symbolic Logic 39(1), 189–190 (1974)
11. Robinson, J.A.: A Machine-oriented Logic Based on the Resolution Principle. Journal of the ACM (JACM) 12(1), 23–41 (1965)
12. Slagle, J.R., Norton, L.M.: Experiment with an Automatic Theorem-prover Having Partial Ordering Inference Rules. Communications of the ACM 16(11), 682–688 (1973)

A Appendix

A.1 Correctness of the Encoding of Closed-Walk Detection Problem

To prove that this kind of encoding suit for all closed walk detection problems, a proof of the theorem below is given.

Theorem 4. *Let G be a graph and s be a vertex in G. Starting from s, there exists a closed walk if and only if starting from $\{\mathsf{OC}(s, G)\} \cup \mathsf{EC}(G)$, a success clause can be derived.*

Before proving this theorem, several notations and lemmas are needed, which will also be used in the following sections.

Notations Let C_1, C_2, C_3 be clauses, Γ be a set of clauses:

- if C_3 is generated by applying resolution between C_1 and C_2, then write the resolution step as $C_1 \xrightarrow{C_2} C_3$; if the resolution is based on a selection function δ, then the resolution step is written as $C_1 \xrightarrow{C_2}_\delta C_3$.
- if C_2 is generated by applying resolution between C_1 and a clause in Γ, then write the resolution step as $C_1 \xrightarrow{\Gamma} C_2$; if the resolution is based on a selection function δ, then the resolution step is written as $C_1 \xrightarrow{\Gamma}_\delta C_2$.
- if C_1 is generated by one step of resolution on some clauses in Γ, then write the resolution step as $\Gamma \longrightarrow \Gamma, C_1$; if the resolution is based on a selection function δ, then the resolution step is written as $\Gamma \longrightarrow_\delta \Gamma, C_1$.

Lemma 1. *For any two traversal clauses, we cannot apply resolution rules between them.*

Proof. All the literals in traversal clauses are positive. □

Lemma 2. *If resolution rules can be applied between a traversal clause and a coloring clause, then one and only one traversal clause can be derived.*

Proof. As all the literals in the traversal clause are positive and there is only one negative literal in the coloring clause, straightforwardly, only one traversal clause can be derived. □

Proposition 1. *Let M be a set of coloring clauses, $C_1, \ldots, C_n$ be traversal clauses and S be a success clause. If $M, C_1, \ldots, C_n \to S$, then there exists $1 \leq i \leq n$, such that $M, C_i \to S$, and the length of the later derivation is at most equal to the former one.*

Proof. By induction on the size of the derivation $M, C_1, \ldots, C_n \to S$.

- If S is a member of $C_1, \ldots, C_n$, then there exists the derivation $M, S \to S$ without applying any resolution rules.

- If S is not a member of $C_1, \ldots, C_n$, then in each step of the derivation, by Lemma 1, the resolution rules can only be applied between a traversal clause and a coloring clause. Assume the derivation is $M, C_1, \ldots, C_n \longrightarrow M, C_1, \ldots, C_n, C' \rightarrow S$, in which, by Lemma 2, C' is a traversal clause. Then for the derivation $M, C_1, \ldots, C_n, C' \rightarrow S$, by induction hypothesis, $M, C' \rightarrow S$ or there exists $1 \leq i \leq n$ such that $M, C_i \rightarrow S$, with the steps of the derivation at most equal to $M, C_1, \ldots, C_n, C' \rightarrow S$. If $M, C_i \rightarrow S$, then the steps of the derivation are less than $M, C_1, \ldots, C_n \rightarrow S$, thus this derivation is as needed. If $M, C' \rightarrow S$, then by Lemma 1, there exists C_j in $C_1, \ldots, C_n$, such that $C_j \xrightarrow{M} C'$, thus the derivation $M, C_j \rightarrow S$, with the derivation steps at most equal to $M, C_1, \ldots, C_n \rightarrow S$, is as needed.

□

Proposition 2. *Let M be a set of coloring clauses, C be a traversal clause, and S be a success clause. If $M, C \rightarrow S(\pi_1)$[1], then there exists a derivation path $C(C_0) \xrightarrow{M} C_1 \xrightarrow{M} C_2 \cdots \xrightarrow{M} C_n(S)$.*

Proof. By induction on the size of the derivation π_1.

- If C is a success clause, then the derivation path can be built directly.
- Otherwise, by Lemma 1, in each step of the derivation, the resolution rules can only be applied between a traversal clause and a coloring clause. Assume the derivation is $M, C \longrightarrow M, C, C' \rightarrow S$, then for the derivation $M, C, C' \rightarrow S$, by Proposition 1, there exists a derivation $M, C \rightarrow S(\pi_2)$[2] or $M, C' \rightarrow S$, with the length less than π_1. For π_2, by induction hypothesis, there exists a derivation path $C(C_0) \xrightarrow{M} C_1 \cdots \xrightarrow{M} C_n(S)$, and this is just the derivation as needed. For $M, C' \rightarrow S$, by induction hypothesis, there exists a derivation path $C' \xrightarrow{M} C'_1 \cdots \xrightarrow{M} C'_m(S)$. As $C \xrightarrow{M} C'$, the derivation path $C \xrightarrow{M} C' \xrightarrow{M} C'_1 \cdots \xrightarrow{M} C'_m(S)$ is as needed.

□

Now it is ready to prove Theorem 4. The proof is as follows.
Proof of Theorem 4

- For the right direction, we assume that the path is

$$\begin{tikzcd} s_1(s_{k_1}) \arrow[r] & s_{k_2} \arrow[r] & \cdots \arrow[r] & s_{k_i} \arrow[r] & s_{k_{i+1}} \arrow[r] & \cdots \arrow[r] & s_{k_j} \arrow[llll, bend left] \end{tikzcd}$$

  By the method of generating E-coloring clauses of a graph, there exist E-coloring clauses:

$$\underline{W_{k_1}^{\perp}} \vee B_{k_2}, \ \ \underline{W_{k_2}^{\perp}} \vee B_{k_3}, \ \ \ldots, \ \ \underline{W_{k_{i-1}}^{\perp}} \vee B_{k_i}, \ \ \underline{W_{k_i}^{\perp}} \vee B_{k_{i+1}}, \ \ \ldots, \ \ \underline{W_{k_j}^{\perp}} \vee B_{k_i}.$$

[1] we denote the derivation as π_1.
[2] we denote the derivation as π_2.

Then starting from the original clause $C_1 = B_1 \vee W_1 \vee \cdots \vee W_n$, the derivation

$$C_1 \xrightarrow{D_1} C_2 \xrightarrow{D_2} \cdots C_{i-1} \xrightarrow{D_{i-1}} C_i \xrightarrow{D_i} \cdots C_j \xrightarrow{D_j} C_{j+1}$$

can be built, in which C_{j+1} is a success clause and for each $1 \le m \le j$, D_m is the E-coloring clause $\underline{W_{k_m}^{\perp}} \vee B_{k_{m+1}}$.

– For the left direction, by Proposition 2, starting from the original clause $C_1 = B_1 \vee W_1 \vee \cdots \vee W_n$, there exists a derivation path

$$C_1 \xrightarrow{D_1} C_2 \xrightarrow{D_2} \cdots C_{i-1} \xrightarrow{D_{i-1}} C_i \xrightarrow{D_i} \cdots C_j \xrightarrow{D_j} C_{j+1},$$

in which C_{j+1} is a success clause and for each $1 \le m \le j$, D_m is an E-coloring clause. As C_{j+1} is a success clause, for each black literal B_i in the clause C_{j+1}, there exists an E-coloring clause $\underline{W_i^{\perp}} \vee B_{k_i}$ in $D_1, \dots, D_j$. Thus for each black literal B_i in the clause C_{j+1}, there exists a vertex s_{k_i} such that there is an edge from s_i to s_{k_i}. As the number of black literals in C_{j+1} is finite, for each vertex s_i, if B_i is a member of C_{j+1}, then starting from s_i, there exists a path which contains a cycle. As the literal B_1 is in C_{j+1}, starting from s_1, there exists a path to a cycle.

□

A.2 Correctness of the Encoding of Block-Walk Detection Problem

Theorem 5. *Let G be a graph and s_1 be a vertex of G. Starting from s_1, there is no blocked walk if and only if, starting from $\{\mathsf{OC}(s_1, G)\} \cup \mathsf{AC}(G)$, a success clause can be derived.*

Before proving this theorem, a lemma is needed.

Lemma 3. *Let G be a graph and s_1 be a vertex of G. Starting from s_1, if all the reachable vertices are traversed in the order $s_1, s_2, \dots, s_k$ and each reachable vertex has at least one successor, then starting from $\{\mathsf{OC}(s_1, G)\} \cup \mathsf{AC}(G)$, there exists a derivation path $C_1(\mathsf{OC}(s_1, G)) \xrightarrow{D_1} C_2 \xrightarrow{D_2} \cdots C_k \xrightarrow{D_k} C_{k+1}$, in which C_{k+1} is a success clause and $\forall 1 \le i \le k$, D_i is an A-coloring clause of the form $\underline{W_i^{\perp}} \vee B_{i_1} \vee \cdots \vee B_{i_j}$.*

Proof. As $s_1, s_2 \dots, s_k$ are all the reachable vertices starting from s_1, for a vertex s, if there exists an edge from one of the vertices in $s_1, s_2, \dots, s_k$ to s, then s is a member of $s_1, s_2, \dots, s_k$. Thus, after the derivation $C_1 \xrightarrow{D_1} C_2 \xrightarrow{D_2} \cdots C_j \xrightarrow{D_j} C_{j+1}$, for each black literal B_i, the white literal W_i is not in C_{j+1}, thus C_{j+1} is a success clause. □

Now it is ready to prove Theorem 5. The proof is as follows.
Proof of Theorem 5

– For the right direction, assume that all the reachable vertices starting from s_1 are traversed in the order $s_1, s_2, \dots, s_k$. For the resolution part, by Lemma 3, starting from the original clause, a success clause can be derived.

– For the left direction, by Proposition 2, starting from the original clause $C_1 = \mathsf{OC}(s_1, G)$, there exists a derivation path

$$C_1 \xrightarrow{D_1} C_2 \xrightarrow{D_2} \cdots C_j \xrightarrow{D_j} C_{j+1},$$

in which C_{j+1} is a success clause and $\forall 1 \le i \le j$, D_i is an A-coloring clause with $W_{k_i}^{\perp}$ underlined. As there is no i such that both B_i and W_i are in C_{j+1}, for the vertices in $s_{k_1}, s_{k_2}, \ldots, s_{k_j}$, the successors of each vertex is a subset of $s_{k_1}, s_{k_2}, \ldots, s_{k_j}$. As the black literal B_1 is in the clause C_{j+1}, by the definition of success clause, the white literal W_1 is not in C_{j+1}, thus s_1 is a member of $s_{k_1}, s_{k_2}, \ldots, s_{k_j}$. Then recursively, for each vertex s, if s is reachable from s_1, then s is in $s_{k_1}, s_{k_2}, \ldots, s_{k_j}$. Thus starting from s_1, all the vertices reachable have successors.

□

A.3 Completeness of PRM$_\delta$ + PSR

For the completeness of our method, we first prove that PRM$_\delta$ is complete, then we prove that PRM$_\delta$ remains complete when we apply PSR eagerly.

Proposition 3 (Completeness of PRM$_\delta$). *Let M be a set of coloring clauses and $C_1, \ldots, C_n$ be traversal clauses. If $M, C_1, \ldots, C_n \to S$, in which the clause S is a success clause, then starting from $M, C_1, \ldots, C_n$, we can build a derivation by selecting the resolved literals with selection function δ in Definition 4 and get a success clause.*

Proof. By Proposition 1 and Proposition 2, there exists $1 \le i \le n$, such that $C_i(C_{i_0}) \xrightarrow{D_1} C_{i_1} \cdots \xrightarrow{D_n} C_{i_n}(S)$. As there are no white literals in any clauses of $D_1, \ldots, D_n$ and in each step of the resolution, the resolved literal in the traversal clause is a white literal, the order of white literals to be resolved in the derivation by applying Resolution rule with coloring clauses in $D_1, \ldots, D_n$ will not affect the result. Thus use selection function δ to select white literals to be resolved, until we get a traversal clause S' such that there are no grey literals in it. By the definition of success clause, S' is a success clause. □

Lemma 4. *Let M be a set of coloring clauses and C be a traversal clause. Assume $C(H_0) \xrightarrow{D_1}_\delta H_1 \xrightarrow{D_2}_\delta \cdots H(H_i) \xrightarrow{D_i}_\delta \cdots \xrightarrow{D_n}_\delta H_n$ in which H_n is a success clause and for each $1 \le j \le n$, the coloring clause D_j is in M, and $M, C \to^\delta K$ such that $\mathsf{grey}(H) = \mathsf{grey}(K)$. If $K, D_1, \ldots, D_n \to^\delta K'$, and K' is not a success clause, then there exists a coloring clause D_k in $D_1, \ldots, D_n$, such that $K' \xrightarrow{D_k}_\delta K''$.*

Proof. As K' is not a success clause, assume that the literals B_i and W_i are in K'. As W_i cannot be introduced in each step of resolution between a traversal clause and a coloring clause, W_i is in C and K. As the literal B_i is in clause K', during the derivation of K', there must be some clauses which contains B_i:

– if the literal B_i is in K, as W_i is also in K, W_i is a grey literal of K. As $\mathsf{grey}(H) = \mathsf{grey}(K)$, the literal B_i is also in H, and as B_i cannot be selected during the derivation, it remains in the traversal clauses $H_{i+1}, \ldots, H_n$.
– if the literal B_i is introduced by applying Resolution rule with coloring clause D_j in $D_1, \ldots, D_n$, which is used in the derivation of H_n as well, so the literal B_i is also a member of H_n.

In both cases, the literal B_i is in H_n. As H_n is a success clause, the literal W_i is not a member of H_n. As W_i is in C, there exists a coloring clause D_k in $D_1, \ldots, D_n$ with the literal $W_i^{\perp}$ selected. Thus, $K' \xrightarrow{D_k}_{\delta} K''$. □

Lemma 5. *Let M be a set of A(E)-coloring clauses and C be a traversal clause. If we have $M, C \to^{\delta} H$ and $M, C \to^{\delta} K$, such that $\mathsf{grey}(H) = \mathsf{grey}(K)$, then starting from M, H a success clause can be derived if and only if starting from M, K a success clause can be derived.*

Proof. Without loss of generality, prove that if starting from M, H we can get to a success clause, then starting from M, K, we can also get to a success clause. By Proposition 2, starting from C, there exists $H_0(C) \xrightarrow{M}_{\delta} H_1 \xrightarrow{M}_{\delta} \cdots H_i(H) \xrightarrow{M}_{\delta} \cdots \xrightarrow{M}_{\delta} H_n$, in which H_n is a success clause. More precisely, $H_0(C) \xrightarrow{D_1}_{\delta} H_1 \xrightarrow{D_2}_{\delta} \cdots H_i(H) \xrightarrow{D_{i+1}}_{\delta} \cdots \xrightarrow{D_n}_{\delta} H_n$, where for each $1 \leq j \leq n$, the coloring clause D_j is in M. Then by Lemma 4, starting from M, K, we can always find a coloring clause in $D_1, \ldots, D_n$ to apply resolution with the new generated traversal clause, until we get a success clause. As the white literals in the generated traversal clauses decrease by each step of resolution, we will get a success clause at last. □

Theorem 6 (Completeness). *PRM_{δ} with PSR is complete.*

Proof. By Lemma 5, each time after we apply PSR, the satisfiability is preserved. □