The Set of Support Strategy in Temporal Resolution

Clare Dixon and Michael Fisher

Department of Computing and Mathematics
Manchester Metropolitan University
Manchester M1 5GD U.K.

EMAIL: {C,Dixon,M.Fisher}@doc.mmu.ac.uk

Abstract

A variety of proof methods have been developed to support the effective mechanisation of temporal logic. While *clausal* temporal resolution has been successfully employed for a range of problems, a number of improvements are still required. In particular, as there is no consistent control strategy underlying the method, a large amount of irrelevant information may sometimes be generated.

Following on from classical resolution, where the *Set of Support* strategy has been used very successfully, we here introduce, justify and apply a temporal version of this strategy, thus allowing the supporting set to be carried over between the different phases of the resolution method. This not only restricts the production of irrelevant information but, under certain conditions, retains the completeness of the refutation process.

1 Introduction

The effective mechanisation of temporal logic is vital to the application of temporal reasoning in many fields, for example the verification of reactive systems [13], the implementation of temporal query languages [4], and temporal logic programming [1]. Consequently, a range of proof methods have been developed, implemented and applied. In addition to well-known tableau [21] and automata-theoretic [19] methods, there has been a resurgence in interest in resolution-based methods [2, 3, 20, 9].

Although *clausal* temporal resolution [9] has been developed and implemented [5, 6] and has been used successfully in a variety of applications, experiments have shown that, in a number of cases, the basic method leads to the generation of an unnecessarily large set of clauses. As the majority of these clauses are irrelevant to the refutation, it is clear that refinements are needed. One approach, adopted in [11], is to refine the temporal resolution rule itself so that the original requirement to carry out all *step* resolution (effectively classical resolution) before attempting *temporal* resolution can be discarded. An additional refinement has concerned the step resolution phase itself. As in classical resolution, a *set of support* strategy [22] is used in attempting to derive a contradiction via step resolution [8]. However, in this work, the set of support is not utilised in successive temporal resolution phases.

While the above refinements of the basic temporal resolution approach have met with some success, we now consider utilising the set of support strategy *throughout* the resolution process. This involves passing the set of support between step and temporal phases and modifying the temporal resolution rule to incorporate this strategy. Thus, this paper introduces a new cycle of operations to be used in temporal resolution based on the set of support, proves the circumstances under which these restrictions are complete, and considers their implications for improvement in efficiency. The aim of this is to provide a strategy which will focus, more closely, on the core elements required for a refutation.

The structure of the paper is as follows. In §2 we define the form of temporal logic considered[1],

[1] For simplicity, we consider this future-time temporal logic rather than that defined in [9], which incorporated past-time operators.

namely Propositional Temporal Logic (PTL) [12], and review the original temporal resolution method. In §3, we introduce, justify and apply the new method based throughout on the set of support strategy. The implications of this approach, both for completeness of the resolution method and for its efficiency, are considered in §4. Finally, in §5, we provide conclusions and discuss future work in this area.

2 The Temporal Resolution Method

2.1 Propositional Temporal Logic

In this section, we present the syntax and semantics of the temporal logic we consider, namely PTL [12]. This generalises classical propositional logic, and thus it contains the standard propositional connectives $\neg$ (not) and $\vee$ (or); the remaining connectives are assumed to be introduced as abbreviations in the usual way. We use temporal connectives that can refer to the *future*, namely $\bigcirc$ (for 'next') and $\mathcal{U}$ (for 'until'). We explain these connectives in detail below. The temporal connectives are interpreted over a *flow of time* that is linear, discrete, bounded in the past, and infinite in the future. An obvious choice for such a flow of time is $(\mathbb{N}, <)$, i.e., the natural numbers ordered by the usual 'less than' relation.

2.1.1 Syntax

We now formally present the syntax of PTL.

Definition 1 *The language of PTL contains the following symbols:*

1. *A set* $\Phi = \{p, q, r, \ldots\}$ *of* primitive propositions*;*
2. **true** *and* **false***;*
3. *The binary propositional connective* $\vee$ *(or), and unary propositional connective* $\neg$ *(not);*
4. *The nullary temporal connective* **start***, the unary temporal connective* $\bigcirc$ *(next) and binary temporal connective* $\mathcal{U}$ *(until).*

Definition 2 *The set* WFF *of well-formed formulae of PTL is defined by the following rules:*

1. *(Primitive propositions are formulae): if* $p \in \Phi$ *then* $p \in$ WFF*;*
2. *(Nullary connectives):* $\mathbf{true}, \mathbf{false}, \mathbf{start} \in$ WFF*;*
3. *(Unary connectives): if* $\phi \in$ WFF *then* $\neg\phi \in$ WFF*,* $\bigcirc\phi \in$ WFF*, and* $(\phi) \in$ WFF*;*
4. *(Binary connectives): if* $\phi, \psi \in$ WFF*, then* $\phi \vee \psi \in$ WFF*, and* $\phi\,\mathcal{U}\,\psi \in$ WFF.

2.1.2 Semantics

We define a model, M, for PTL as a structure $\langle \mathcal{D}, \pi_p \rangle$ where

- $\mathcal{D}$ is the temporal domain, i.e. the Natural Numbers ($\mathbb{N}$), and
- $\pi_p : \mathcal{D} \times \Phi \to \{T, F\}$ is a function assigning T or F to each atomic proposition at each moment in time.

As usual, we define the semantics of the language via the satisfaction relation '$\models$'. For PTL, this relation holds between pairs of the form $\langle M, u \rangle$ (where M is a model and $u \in \mathbb{N}$), and well-formed PTL formulae. The rules defining the satisfaction relation are given below

$$
\begin{array}{lll}
\langle M, u \rangle \models \mathbf{true} & & \\
\langle M, u \rangle \not\models \mathbf{false} & & \\
\langle M, u \rangle \models \mathbf{start} & \text{iff} & u = 0 \\
\langle M, u \rangle \models q & \text{iff} & \pi_p(u, q) = T \text{ (where } q \in \Phi) \\
\langle M, u \rangle \models \neg\phi & \text{iff} & \langle M, u \rangle \not\models \phi \\
\langle M, u \rangle \models \phi \vee \psi & \text{iff} & \langle M, u \rangle \models \phi \text{ or } \langle M, u \rangle \models \psi \\
\langle M, u \rangle \models \bigcirc\phi & \text{iff} & \langle M, u+1 \rangle \models \phi \\
\langle M, u \rangle \models \phi\,\mathcal{U}\,\psi & \text{iff} & \exists v \in \mathbb{N} \text{ such that } (u \leq v) \\
 & & \quad \text{and } \langle M, v \rangle \models \psi, \text{and} \\
 & & \quad \forall w \in \mathbb{N}, \text{ if } (u \leq w < v) \\
 & & \quad \text{then } \langle M, w \rangle \models \phi
\end{array}
$$

Satisfiability and validity in PTL are defined in the usual way.

Other standard temporal connectives are introduced as abbreviations, in terms of $\mathcal{U}$, e.g.,

$$
\begin{array}{rcl}
\Diamond\phi & \stackrel{\text{def}}{=} & \mathbf{true}\,\mathcal{U}\,\phi \\
\Box\phi & \stackrel{\text{def}}{=} & \neg\Diamond\neg\phi \\
\phi\,\mathcal{W}\,\psi & \stackrel{\text{def}}{=} & \phi\,\mathcal{U}\,\psi \vee \Box\phi
\end{array}
$$

We now informally consider the meaning of the temporal connectives. First, consider the two basic connectives: $\bigcirc$ and $\mathcal{U}$. The $\bigcirc$ connective means 'at the next time'. Thus $\bigcirc\phi$ will be satisfied at some time if ϕ is satisfied at the *next* moment in time. The $\mathcal{U}$ connective means 'until'. Thus $\phi\mathcal{U}\psi$ will be satisfied at some time if ψ is satisfied at either the present time or some time in the future, and ϕ is satisfied at all times until the time that ψ is satisfied. As an example of the derived connectives, $\Diamond\phi$ will be satisfied at some time if ϕ is satisfied either at the present moment or at some future time. Finally, a temporal operator that takes no arguments is defined which is true only at the first moment in time: this operator is '**start**'. For more information about PTL and its application in specification and verification see for example [13, 14].

2.2 Basic Resolution Method for PTL

Before describing the resolution method in detail, we outline the motivation for the approach adopted. First, we recap the problems associated with clausal resolution in non-classical logics. The main problem with extending resolution to temporal logics, such as PTL, is that literals cannot generally be moved across temporal contexts. In particular, if T is a temporal operator, p and $\mathsf{T}\neg p$ cannot generally be resolved. Thus, the only inferences that can be made occur in particular temporal contexts. For example, both p and $\neg p$ can be resolved, as, for certain types of temporal operator, can $\mathsf{T}p$ and $\mathsf{T}\neg p$.

The clausal resolution method introduced in [9] addresses this problem by utilising a normal form, called Separated Normal Form (SNF), which separates out complex formulae from their contexts through the use of *renaming* [18], and a new temporal resolution rule introduced specifically for formulae in the normal form.

The resolution method consists of the following cycle of steps. To determine whether a formula, $\phi \in \textsc{wff}$, is unsatisfiable

1. Rewrite ϕ into SNF
2. Repeat
   (a) apply *step* resolution until either a contradiction is generated, or no further step resolution can be carried out
   (b) rewrite any new resolvents into SNF
   (c) apply simplification and subsumption rules
   (d) apply the temporal resolution rule
   (e) rewrite new resolvents into SNF

   until either **false** is derived or no more rules can be applied.

We briefly review each of these items, starting with the normal form itself.

2.2.1 Separated Normal Form

This resolution method [9], depends on formulae being rewritten into a normal form, called Separated Normal Form (SNF) [10]. In this section, we review SNF but do not consider the transformation procedure that takes an arbitrary formula of PTL and rewrites it into SNF (for further details, see [10]). We note that the transformation to SNF preserves satisfiability, i.e. given any PTL formula φ that is satisfiable, its translation into SNF is also satisfiable [10].

A formula in SNF is of the form:

$$\Box \bigwedge_{i=1}^{n} (\phi_i \Rightarrow \psi_i)$$

where each of the '$\phi_i \Rightarrow \psi_i$' (called *rules*) is one of the following.

$$\begin{array}{lcll}
\textbf{start} & \Rightarrow & \displaystyle\bigvee_{k=1}^{r} l_k & \text{(an \textit{initial} rule)} \\
\displaystyle\bigwedge_{j=1}^{q} m_j & \Rightarrow & \bigcirc \displaystyle\bigvee_{k=1}^{r} l_k & \text{(an \textit{always} rule)} \\
\displaystyle\bigwedge_{j=1}^{q} m_j & \Rightarrow & \Diamond l & \text{(a \textit{sometime} rule)}
\end{array}$$

where each m_j, l_k or l is a literal.

2.2.2 Step Resolution

The step resolution rules are simply versions of the classical resolution rule rewritten in two ways. First, the *initial* step resolution rule:

$$\begin{array}{rcl}
\textbf{start} & \Rightarrow & \psi_1 \vee l \\
\textbf{start} & \Rightarrow & \psi_2 \vee \neg l \\
\hline
\textbf{start} & \Rightarrow & \psi_1 \vee \psi_2
\end{array}$$

Then the *global* step resolution rule:

$$\begin{array}{rcl}
\phi_1 & \Rightarrow & \bigcirc(\psi_1 \vee l) \\
\phi_2 & \Rightarrow & \bigcirc(\psi_2 \vee \neg l) \\
\hline
(\phi_1 \wedge \phi_2) & \Rightarrow & \bigcirc(\psi_1 \vee \psi_2)
\end{array}$$

2.2.3 Simplification

The simplification rules used are similar to the classical case, consisting of both simplification and subsumption rules, and so will not be duplicated here. An additional rule is required when a contradiction in a state is produced, i.e.,

$$\frac{\phi \;\Rightarrow\; \bigcirc\mathbf{false}}{\begin{array}{rcl}\mathbf{start} & \Rightarrow & \neg\phi\\ \mathbf{true} & \Rightarrow & \bigcirc\neg\phi\end{array}}$$

This shows that, if a particular formula leads to a contradiction, then that formula should not be satisfied either in the initial state or in any subsequent state.

2.2.4 Temporal Resolution

Rather than describe the temporal resolution rule in detail, we refer the interested reader to [9]. The basic idea is to resolve one sometime rule with a *set* of always rules as follows.

$$\left.\begin{array}{rcl}\phi_1 & \Rightarrow & \bigcirc\psi_1\\ \phi_2 & \Rightarrow & \bigcirc\psi_2\\ \vdots & \vdots & \vdots\\ \phi_n & \Rightarrow & \bigcirc\psi_n\\ \chi & \Rightarrow & \Diamond\neg l\\ \hline \chi & \Rightarrow & (\neg\bigvee_{i=1}^{n}\phi_i)\,\mathcal{W}\,\neg l\end{array}\right\}\begin{array}{l}\text{where for all } 1\leqslant i\leqslant n\\ \phi_i \;\Rightarrow\; \bigcirc(l\wedge\bigvee_{j=1}^{n}\phi_j)\end{array}$$

The side condition ensures that the set of $\phi_i \Rightarrow \bigcirc\psi_i$ rules together imply

$$\bigvee_{i=1}^{n}\phi_i \;\Rightarrow\; \bigcirc\Box l.$$

Such a set of rules is known as a *loop* in l. The resolvent states that, once χ has occurred, none of the ϕ_i must occur while the eventuality (i.e. $\Diamond\neg l$) is outstanding. This resolvent must again be translated into SNF.

2.2.5 Termination

Finally, if **start** $\Rightarrow$ **false** is produced, the original formula is unsatisfiable and the resolution process terminates.

2.2.6 Correctness

The soundness and (refutation) completeness of the original temporal resolution method have been established in [9, 16, 5].

2.3 Example

We give an example showing the use of the step and temporal resolution rules. Rules 1–17 are an unsatisfiable set of rules given in SNF.

$$\begin{array}{rrcl}
1. & a & \Rightarrow & \bigcirc l\\
2. & a & \Rightarrow & \bigcirc(a\vee\neg c)\\
3. & a & \Rightarrow & \bigcirc c\\
4. & d & \Rightarrow & \bigcirc(\neg c\vee\neg p\vee\neg q)\\
5. & e & \Rightarrow & \bigcirc(\neg c\vee p\vee\neg q)\\
6. & (f\wedge p) & \Rightarrow & \bigcirc(\neg c\vee\neg p\vee q)\\
7. & (g\wedge q) & \Rightarrow & \bigcirc(\neg c\vee p\vee q)\\
8. & x & \Rightarrow & \bigcirc(y\vee z)\\
9. & y & \Rightarrow & \bigcirc x\\
10. & z & \Rightarrow & \bigcirc x\\
11. & x & \Rightarrow & \bigcirc l\\
12. & y & \Rightarrow & \bigcirc l\\
13. & z & \Rightarrow & \bigcirc l\\
14. & \mathbf{start} & \Rightarrow & d\\
15. & \mathbf{start} & \Rightarrow & a\\
16. & d & \Rightarrow & \Diamond\neg l\\
17. & \mathbf{start} & \Rightarrow & l
\end{array}$$

The rules generated from step resolution are given in Figure 2.3. Next temporal resolution is carried out. The following (combined) rules form a loop in l and can be resolved with rule 16.

$$\begin{array}{rcll}
a & \Rightarrow & \bigcirc(a\wedge l) & [1,18]\\
x & \Rightarrow & \bigcirc((y\wedge l)\vee(z\wedge l)) & [8,11]\\
y & \Rightarrow & \bigcirc(x\wedge l) & [9,12]\\
z & \Rightarrow & \bigcirc(x\wedge l) & [10,13]
\end{array}$$

This produces the resolvent

$$d\Rightarrow(\neg(a\vee x\vee y\vee z))\,\mathcal{W}\,\neg l$$

which can be translated into SNF as follows where t is a new proposition.

$$\begin{array}{rrcl}
43. & \mathbf{start} & \Rightarrow & \neg d\vee\neg y\vee\neg l\\
44. & \mathbf{start} & \Rightarrow & \neg d\vee\neg z\vee\neg l\\
45. & \mathbf{start} & \Rightarrow & \neg d\vee\neg a\vee\neg l\\
46. & \mathbf{start} & \Rightarrow & \neg d\vee\neg x\vee\neg l\\
47. & \mathbf{start} & \Rightarrow & \neg d\vee t\vee\neg l\\
48. & \mathbf{true} & \Rightarrow & \bigcirc(\neg d\vee\neg y\vee\neg l)\\
49. & \mathbf{true} & \Rightarrow & \bigcirc(\neg d\vee\neg z\vee\neg l)\\
50. & \mathbf{true} & \Rightarrow & \bigcirc(\neg d\vee\neg a\vee\neg l)\\
51. & \mathbf{true} & \Rightarrow & \bigcirc(\neg d\vee\neg x\vee\neg l)\\
52. & \mathbf{true} & \Rightarrow & \bigcirc(\neg d\vee t\vee\neg l)\\
53. & t & \Rightarrow & \bigcirc(\neg y\vee\neg l)\\
54. & t & \Rightarrow & \bigcirc(\neg z\vee\neg l)\\
55. & t & \Rightarrow & \bigcirc(\neg a\vee\neg l)\\
56. & t & \Rightarrow & \bigcirc(\neg x\vee\neg l)\\
57. & t & \Rightarrow & \bigcirc(t\vee\neg l)
\end{array}$$

18.	a	$\Rightarrow$	$\bigcirc a$	$[2,3]$
19.	$(d \wedge a)$	$\Rightarrow$	$\bigcirc(\neg p \vee \neg q)$	$[3,4]$
20.	$(e \wedge a)$	$\Rightarrow$	$\bigcirc(p \vee \neg q)$	$[3,5]$
21.	$(f \wedge p \wedge a)$	$\Rightarrow$	$\bigcirc(\neg p \vee q)$	$[3,6]$
22.	$(g \wedge q \wedge a)$	$\Rightarrow$	$\bigcirc(p \vee q)$	$[3,7]$
23.	$(e \wedge d)$	$\Rightarrow$	$\bigcirc(\neg c \vee \neg q)$	$[4,5]$
24.	$(f \wedge p \wedge d)$	$\Rightarrow$	$\bigcirc(\neg c \vee \neg p)$	$[4,6]$
25.	$(g \wedge q \wedge e)$	$\Rightarrow$	$\bigcirc(\neg c \vee p)$	$[5,7]$
26.	$(g \wedge q \wedge f \wedge p)$	$\Rightarrow$	$\bigcirc(\neg c \vee q)$	$[6,7]$
27.	$(e \wedge a \wedge d)$	$\Rightarrow$	$\bigcirc \neg q$	$[19,20]$
28.	$(f \wedge p \wedge a \wedge d)$	$\Rightarrow$	$\bigcirc \neg p$	$[19,21]$
29.	$(g \wedge q \wedge a \wedge e)$	$\Rightarrow$	$\bigcirc p$	$[20,22]$
30.	$(g \wedge q \wedge a \wedge f \wedge p)$	$\Rightarrow$	$\bigcirc q$	$[21,22]$
31.	$(g \wedge q \wedge e \wedge f \wedge p \wedge d)$	$\Rightarrow$	$\bigcirc \neg c$	$[23,26]$
32.	**start**	$\Rightarrow$	$(\neg g \vee \neg q \vee \neg a \vee \neg e \vee \neg f \vee \neg p \vee \neg d)$	$[3,31]$
33.	**true**	$\Rightarrow$	$\bigcirc(\neg g \vee \neg q \vee \neg a \vee \neg e \vee \neg f \vee \neg p \vee \neg d)$	$[3,31]$
34.	**start**	$\Rightarrow$	$(\neg g \vee \neg q \vee \neg a \vee \neg e \vee \neg f \vee \neg p)$	$[14,32]$
35.	**start**	$\Rightarrow$	$(\neg g \vee \neg q \vee \neg e \vee \neg f \vee \neg p)$	$[15,34]$
36.	a	$\Rightarrow$	$\bigcirc(\neg g \vee \neg q \vee \neg e \vee \neg f \vee \neg p \vee \neg d)$	$[18,33]$
37.	$(a \wedge e)$	$\Rightarrow$	$\bigcirc(\neg g \vee \neg q \vee \neg e \vee \neg f \vee \neg d)$	$[20,36]$
38.	e	$\Rightarrow$	$\bigcirc(\neg g \vee \neg q \vee \neg a \vee \neg e \vee \neg f \vee \neg d \vee \neg c)$	$[5,33]$
39.	$(a \wedge f \wedge p)$	$\Rightarrow$	$\bigcirc(\neg g \vee \neg e \vee \neg f \vee \neg p \vee \neg d)$	$[21,36]$
40.	$(f \wedge p)$	$\Rightarrow$	$\bigcirc(\neg g \vee \neg a \vee \neg e \vee \neg f \vee \neg p \vee \neg d \vee \neg c)$	$[6,33]$
41.	$(a \wedge e \wedge g \wedge q \wedge f \wedge p)$	$\Rightarrow$	$\bigcirc(\neg g \vee \neg e \vee \neg f \vee \neg d)$	$[29,39]$
42.	$(e \wedge g \wedge q \wedge f \wedge p)$	$\Rightarrow$	$\bigcirc(\neg g \vee \neg a \vee \neg e \vee \neg f \vee \neg d \vee \neg c)$	$[25,40]$

Figure 1 Example

More step resolution can be carried out and a contradiction produced by resolving rule 45 with rules 14,15 and 17.

58.	**start**	$\Rightarrow$	$\neg a \vee \neg l$	$[14,45]$
59.	**start**	$\Rightarrow$	$\neg l$	$[15,58]$
60.	**start**	$\Rightarrow$	**false**	$[17,59]$

3 Set of Support (SoS)

In this section the set of support for classical resolution is described and then extended to deal with the temporal case.

3.1 SoS in Classical Resolution

The Set of Support strategy [22] is a now well-established mechanism for restricting resolution operations whilst searching for a proof. It has been very successful, being central to a number of theorem-proving systems, in particular Otter [15], the predominant system for classical first-order logic.

The basic idea behind the set of support is to initially identify a set of clauses, and then restrict resolution operations so that they occur between two clauses, at least one of which appears in the set of support. The resolvent is then also added to the set of support.

Definition 3 *Supported Resolution. A resolution inference between two clauses is said to be supported if and only if at least one of the clauses is a member of the set of support.*

If the initial SoS is chosen appropriately, then this approach can lead to a refutation requiring less resolution steps than without using SoS by avoiding the production of excess (irrelevant) clauses. While the use of the set of support is obviously sound, there is a possibility of incompleteness.

Theorem 1 *Completeness of Set of Support. Given an unsatisfiable set of clauses, if the set of support is chosen such that the set of clauses outside the set of support is satisfiable then a* *refutation can be produced using only supported resolution steps.*

Proof See for example [22].

3.2 SoS in Temporal Resolution

The notion of the set of support being used in step resolution follows closely the classical version.

Definition 4 *Supported Step Resolution. A step resolution inference between two rules is said to be supported if and only if at least one of the rules is a member of the set of support.*

Theorem 2 *Completeness of Supported Step Resolution. Given an unsatisfiable set of initial and always rules, if the set of support is chosen such that for the set of rules outside the set of support the set of clauses on the right hand side of the initial rules is satisfiable and the set of clauses on the right hand side of the always rules is satisfiable then a refutation can be produced using only supported step resolution steps.*

Proof Follows from the classical case.

However, in temporal resolution, it is a little more complex.

Definition 5 *Supported Temporal Resolution. The resolvent produced by the temporal resolution rule below is* supported *if* all *the clauses required to form a loop (e.g. $\phi_n \Rightarrow \bigcirc\psi_n$) are in the set of support.*

$$\left.\begin{array}{ccc} \phi_1 & \Rightarrow & \bigcirc\psi_1 \\ \phi_2 & \Rightarrow & \bigcirc\psi_2 \\ \vdots & \vdots & \vdots \\ \phi_n & \Rightarrow & \bigcirc\psi_n \\ \chi & \Rightarrow & \Diamond\neg l \\ \hline \chi & \Rightarrow & (\neg \bigvee_{i=1}^{n} \phi_i)\,\mathcal{W}\,\neg l \end{array}\right\} \quad \begin{array}{l} \textit{where for all } 1 \leqslant i \leqslant n \\ \phi_i \;\Rightarrow\; \bigcirc(l \wedge \bigvee_{j=1}^{n} \phi_j) \end{array}$$

This seems like a strict criterion. Indeed, the cases where this rule is complete is relatively limited.

Theorem 3 *Completeness of Supported Temporal Resolution. The supported temporal resolution rule is complete if, prior to loop search, all clauses that must be combined to produce a loop are members of the set of support.*

Note, although this guarantees completeness it is not necessary for it. In some situations the detection of a non-maximal loop, is enough to lead to a contradiction. Considering the example given in § 2.3 we will see (in § 4.1) that the detection of the (non-maximal) loop

$$a \Rightarrow \bigcirc a \wedge l$$

is sufficient to generate a contradiction. Thus we actually only require the rules that are combined to make this loop to be in the set of support prior temporal resolution to retain completeness.

3.3 Cycle of Operations

The resolution method consists of the following cycle of steps. To determine whether a formula, $\phi \in$ WFF, is unsatisfiable

1. Rewrite ϕ into SNF, identifying set of support S
2. Apply S-*supported* step resolution until either a contradiction is generated, or no further step resolution can be carried out
3. Rewrite any new resolvents into SNF and add to S
4. Apply simplification and subsumption rules to S
5. Choose a sometime rule, e.g. $\ldots \Rightarrow \Diamond l$, such that $\neg l$ occurs on the right hand side of a rule in S, and apply the S-*supported* temporal resolution rule.
6. Rewrite any new resolvents into SNF and add to S
7. If either **false** has been derived or no new resolvents have been been derived, then terminate, otherwise go to 2.

4 Practical Usage

In this section we examine how useful supported temporal resolution is. First we reconsider the example given in § 2.3 using the set of support.

4.1 Example

We place rules 1 and 2 in the set of support. Only one rule is generated in the first cycle of step resolution namely

$$18.\ a \Rightarrow \bigcirc a\ [2,3,\mathrm{SRES}]$$

and added to the set of support. Next as no more step resolution can be carried out we try temporal resolution with rule 16 and search for a loop in l in the set of support. The loop detected is

$$a \Rightarrow \bigcirc(a \wedge l)$$

from rules 1 and 18 producing the resolvent

$$d \Rightarrow \neg a \mathcal{W} \neg l$$

which is written into SNF and added to the set of support as follows.

19.	**true**	$\Rightarrow$	$\bigcirc(\neg d \vee \neg a \vee \neg l)$	$[1,16,18,\mathrm{TRES}]$
20.	**start**	$\Rightarrow$	$\neg d \vee \neg a \vee \neg l$	$[1,16,18,\mathrm{TRES}]$
21.	**true**	$\Rightarrow$	$\bigcirc(\neg d \vee t \vee \neg l)$	$[1,16,18,\mathrm{TRES}]$
22.	**start**	$\Rightarrow$	$\neg d \vee t \vee \neg l$	$[1,16,18,\mathrm{TRES}]$
23.	t	$\Rightarrow$	$\bigcirc(\neg a \vee \neg l)$	$[1,16,18,\mathrm{TRES}]$
24.	t	$\Rightarrow$	$\bigcirc(t \vee \neg l)$	$[1,16,18,\mathrm{TRES}]$

Now resolution is completed by resolving rule 20 with rules 14,15 and 17 to obtain a contradiction as follows.

25.	**start**	$\Rightarrow$	$\neg a \vee \neg l$	$[14,20,\mathrm{SRES}]$
26.	**start**	$\Rightarrow$	$\neg l$	$[15,25,\mathrm{SRES}]$
27.	**start**	$\Rightarrow$	**false**	$[17,26,\mathrm{SRES}]$

4.2 When the SoS is Useful

The supported step and temporal resolution rules are useful in examples when a large amount of irrelevant rules are produced. The example given in § 2.3 illustrates this. Regarding step resolution many of the rules produced from step resolution in Figure 1 are irrelevant to the proof. During loop detection not only do we have to search through a large amount of rules but we also detect a loop involving several literals when in fact the loop $a \Rightarrow a \wedge l$ was sufficient to generate a contradiction.

4.3 Results

Initial experiments show that examples such as that illustrated, where much irrelevant information is generated, show the greatest reduction in the number of rules produced. Work carried out on Peterson's Algorithm [17] shows that the loop search algorithm can still detect a loop from a subset of the complete ruleset reducing the number of combinations required to find the loop.

4.4 Limitations

This approach effectively restricts the search required for a loop [6] to a small set of clauses. In order to retain this efficiency, but extend the completeness of the approach, we utilise the resolution rule developed in [11]. This means that the criterion that all the rules required for the loop must be in the set of support can be relaxed.

We have given no guidance how to choose the set of support in the first place. Identifying a suitable set of support affects both the completeness and the efficiency of the method. In classical resolution work has been carried out on using both semantic and syntactic methods to choose the set of support [22]. We anticipate that extensions of these methods can be adopted.

5 Conclusions and Further Work

We have extended the notion of set of support to the temporal resolution case in order to minimise the amount of irrelevant information produced. The approach works well on problems where the contradiction can be produced by limiting resolution steps to a particular subset of clauses. More work needs to be done to consider how to choose the set of support in the first place and when these sets retain completeness.

The use of the temporal set of support is a restriction strategy utilised in the temporal resolution theorem prover (CLATTER) being developed currently at Manchester. We will use CLATTER to test temporal set of support further.

Acknowledgements

This work was partially supported by EPSRC under Research Grant GR/K57282. Thanks for the comments from Adam Kellett and several anonymous referees on an earlier draft of this paper.

References

[1] M. Abadi and Z. Manna. Temporal Logic Programming. *Journal of Symbolic Computation*, 8: 277–295, 1989.

[2] M. Abadi and Z. Manna. Nonclausal Deduction in First-Order Temporal Logic. *ACM Journal*, 37(2):279–317, April 1990.

[3] A. Cavali and L. Fariñas del Cerro. A Decision Method for Linear Temporal Logic. In R. E. Shostak, editor, *Proceedings of the 7th International Conference on Automated Deduction*, pages 113–127. LNCS 170, 1984.

[4] J. Chomicki and D. Niwinski. On the Feasibility of Checking Temporal Integrity Constraints. *Journal of Computer and System Sciences*, 51(3):523–535, December 1995.

[5] C. Dixon. *Strategies for Temporal Resolution*. PhD thesis, Department of Computer Science, University of Manchester, Manchester M13 9PL, U.K., December 1995.

[6] C. Dixon. Search Strategies for Resolution in Temporal Logics. In *Proceedings of the Thirteenth International Conference on Automated Deduction (CADE)*, Lecture Notes in Computer Science. Springer-Verlag, August 1996.

[7] C. Dixon. Temporal Resolution: Removing Irrelevant Information. In *Proceedings of International Workshop on Temporal Reasoning (TIME)*, Daytona Beach, Florida, May 1997.

[8] C. Dixon. Using Otter for Temporal Resolution. In *Proceedings the Second International Conference on Temporal Logic (ICTL)*, Applied Logic Series, Manchester, U.K., July 1997. Kluwer. To appear.

[9] M. Fisher. A Resolution Method for Temporal Logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*, Sydney, Australia, August 1991. Morgan Kaufman.

[10] M. Fisher. A Normal Form for Temporal Logic and its Application in Theorem-Proving and Execution. *Journal of Logic and Computation*, 7(4), August 1997.

[11] M. Fisher and C. Dixon. Guiding Clausal Temporal Resolution. In *Proceedings the Second International Conference on Temporal Logic (ICTL)*, Applied Logic Series, Manchester, U.K., July 1997. Kluwer. To appear.

[12] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. The Temporal Analysis of Fairness. In *Proceedings of the Seventh ACM Symposium on the Principles of Programming Languages*, pages 163–173, Las Vegas, Nevada, January 1980.

[13] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, New York, 1992.

[14] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.

[15] W. W. McCune. *OTTER 2.0 Users Guide*. Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439-4801, March 1990. ANL-90/9.

[16] M. Peim. Propositional Temporal Resolution Over Labelled Transition Systems. Unpublished Technical Note, Department of Computer Science, University of Manchester, 1994.

[17] G. L. Peterson. Myths about the Mutual Exclusion Problem. *Information Processing Letters*, 12(3):115–116, 1981.

[18] D. A. Plaisted and S. A. Greenbaum. A Structure-Preserving Clause Form Translation. *Journal of Symbolic Computation*, 2(3):293–304, September 1986.

[19] A. P. Sistla, M. Vardi, and P. Wolper. The Complementation Problem for Büchi Automata with Applications to Temporal Logic. *Theoretical Computer Science*, 49:217–237, 1987.

[20] G. Venkatesh. A Decision Method for Temporal Logic based on Resolution. *Lecture Notes in Computer Science*, 206:272–289, 1986.

[21] P. Wolper. The Tableau Method for Temporal Logic: An overview. *Logique et Analyse*, 110–111:119–136, June-Sept 1985.

[22] L. Wos, G. Robinson, and D. Carson. Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. *ACM Journal*, 12:536–541, October 1965.