# A Resolution Method for Temporal Logic

**Michael Fisher***

Department of Computer Science
University of Manchester
Manchester, U.K.
(michael@cs.man.ac.uk)

## Abstract

In this paper, a resolution method for propositional temporal logic is presented. Temporal formulae, incorporating both past-time and future-time temporal operators, are converted to *Separated Normal Form* (SNF), then both non-temporal and temporal resolution rules are applied. The resolution method is based on classical resolution, but incorporates a temporal resolution rule that can be implemented efficiently using a graph-theoretic approach.

## 1 Introduction

This report describes a resolution procedure for discrete, linear, propositional temporal logic. This logic incorporates both past-time and future-time temporal operators and its models consist of sequences of states, each sequence having finite past and infinite future.

A naive application of the classical resolution rule to temporal logics fails as two complementary literals may not represent a contradictory formula, depending on their temporal context. Because of such problems with resolution, the majority of the decision methods for temporal logics have been based either on tableaux or automata-theoretic techniques [Wolper, 1985; Vardi and Wolper, 1986]. Recently, however, interest has been rekindled in the use of resolution in such logics.

The resolution method described in this report relies on a translation of temporal formulae into a normal form. This normal form is derived from that developed for use in METATEM [Barringer *et al.*, 1989], an executable temporal logic, and the rewrite rules used to produce the normal form are derived from the work on the transformation and determinisation of METATEM programs [Fisher and Noël, 1992]. Several of these transformations are similar to those developed in [Sakuragawa, 1986].

Before developing the resolution procedure in detail, an outline of the temporal logic used, will be given.

## 2 A Linear Temporal Logic

In this section, a propositional temporal logic based on a linear and discrete model of time, with finite past and infinite future, is introduced. Temporal logic can be seen as classical logic extended with various modalities. Commonly, these are $\Diamond$, $\Box$, and $\bigcirc$. The intuitive meaning of these connectives is as follows: $\Diamond A$ is true now if $A$ is true *sometime* in the future; $\Box A$ is true now if $A$ is true *always* in the future; and $\bigcirc A$ is true now if $A$ is true at the *next* moment in time. Similar connectives are introduced to enable reasoning about the *past*.

### 2.1 Syntax

*Well-formed formulae* (wff) are defined as follows.

- any element of PROP is a wff,
- if $A$ and $B$ are wff's, then the following are all wff's

$$\begin{array}{llll} \neg A & A \vee B & A \wedge B & A \Rightarrow B \\ \Diamond A & \Box A & A\,\mathcal{U}\,B & A\,\mathcal{W}\,B \\ \blacklozenge A & \blacksquare A & A\,\mathcal{S}\,B & A\,\mathcal{Z}\,B \\ \bigcirc A & \circledcirc A & \bullet A & (A) \end{array}$$

Further sub-classifications of temporal formulae are defined as follows. A *literal* is defined as either a proposition (i.e., an element of PROP), or the negation of a proposition. A *State-formula* is either a literal or a boolean combination of other state-formulae.

*Future-time formulae* (non-strict) are defined as follows

- if $A$ is a state-formula, then $A$ is a future-time formula,
- if $A$ and $B$ are future-time formulae, then $\neg A$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A\,\mathcal{U}\,B$, $A\,\mathcal{W}\,B$, $\bigcirc A$, $\Diamond A$, and $\Box B$ are all future-time formulae.

*Strict past-time formulae* are defined as follows

- if $A$ and $B$ are either state-formulae or strict past-time formulae, then $\circledcirc A$, $\bullet B$, $A\,\mathcal{S}\,B$, $A\,\mathcal{Z}\,B$, $\blacklozenge A$, and $\blacksquare B$ are all strict past-time formulae,
- if $A$ and $B$ are strict past-time formulae, then $\neg A$, $A \wedge B$, $A \vee B$, and $A \Rightarrow B$ are all strict past-time formulae.

### 2.2 Semantics

The temporal logic used is based on a discrete, linear model, $\sigma$, having finite past and infinite future, i.e.

$$\sigma \;=\; s_0,\ s_1,\ s_2,\ s_3,\ \ldots$$

Here, each $s_i$ is called a *state* and is a subset of PROP representing the propositions that are true at the $i^{th}$ moment in

*This work was supported by ESPRIT under Basic Research Action 3096 (SPEC).

time. An interpretation for this logic is defined as a pair $(\sigma, i)$, where $\sigma$ is the model and $i$ the index of the state at which the temporal statement is to be interpreted.

A semantics for well-formed temporal formulae is defined as a relation between interpretations and formulae, and is defined inductively as follows, with the (infix) semantic relation being represented by $\models$. The semantics of a proposition is defined by the valuation given to it in the model:

$$(\sigma, i) \models p \quad \text{iff} \quad p \in s_i \qquad \text{for } p \in \text{PROP}.$$

The semantics of the standard propositional connectives are as in classical logic, for example

$$(\sigma, i) \models A \lor B \quad \text{iff} \quad (\sigma, i) \models A \quad \text{or} \quad (\sigma, i) \models B.$$

The semantics of the unary future-time temporal operators is defined as follows

$$\begin{array}{lll} (\sigma, i) \models \bigcirc A & \text{iff} & (\sigma, i+1) \models A \\ (\sigma, i) \models \Diamond A & \text{iff} & \text{there exists a } j \geq i \text{ s.t. } (\sigma, j) \models A \\ (\sigma, i) \models \Box A & \text{iff} & \text{for all } j \geq i \text{ then } (\sigma, j) \models A. \end{array}$$

The two binary future-time temporal operators are interpreted as follows

$$\begin{array}{lll} (\sigma, i) \models A \,\mathcal{U}\, B & \text{iff} & \text{there exists a } k \geq i \text{ s.t. } (\sigma, k) \models B \\ & & \text{and for all } i \leq j < k \text{ then } (\sigma, j) \models A \\ (\sigma, i) \models A \,\mathcal{W}\, B & \text{iff} & (\sigma, i) \models A \,\mathcal{U}\, B \quad \text{or} \\ & & \text{for all } j \geq i \text{ then } (\sigma, j) \models A \end{array}$$

If past-time temporal formulae are interpreted at a particular state, $s_i$, then states with indices less than $i$ are *in the past* of the state $s_i$. The semantics of unary past-time operators are given as follows:

$$\begin{array}{l} (\sigma, i) \models \bullet A \ \text{iff} \ i = 0 \quad \text{or} \quad (\sigma, i-1) \models A \\ (\sigma, i) \models \circledcirc A \ \text{iff} \ i > 0 \quad \text{and} \quad (\sigma, i-1) \models A \\ (\sigma, i) \models \blacklozenge A \ \text{iff} \\ \qquad \text{there exists a } j \text{ s.t. } 0 \leq j < i \text{ and } (\sigma, j) \models A \\ (\sigma, i) \models \blacksquare A \ \text{iff for all } j \text{ s.t. } 0 \leq j < i \text{ then } (\sigma, j) \models A. \end{array}$$

Note that, in contrast to the future-time operators, the $\blacklozenge$ ('sometime in the past') and $\blacksquare$ ('always in the past') operators are interpreted as being *strict*, i.e., the current index is not included in the definition. Also, as there is a unique start state, termed the *beginning of time*, two different last-time operators are used. The difference between $\circledcirc$ and $\bullet$ is that for any formula $A$, $\circledcirc A$ is false, while $\bullet A$ is true, when interpreted at the beginning of time. In particular, $\bullet$ **false** is *only* true when interpreted at the beginning of time; otherwise it is false.

Apart from their strictness, the binary past-time operators are similar to their future-time counterparts; their semantics is defined as follows.

$$\begin{array}{lll} (\sigma, i) \models A \,\mathcal{S}\, B & \text{iff} & \text{there exists a } k \text{ s.t. } 0 \leq k < i \text{ and} \\ & & (\sigma, k) \models B \text{ and} \\ & & \text{for all } j \text{ s.t. } k < j < i \text{ then } (\sigma, j) \models A \\ (\sigma, i) \models A \,\mathcal{Z}\, B & \text{iff} & \text{for all } j \text{ such that } 0 \leq j < i \text{ then} \\ & & (\sigma, j) \models A \quad \text{or} \quad (\sigma, i) \models A \,\mathcal{S}\, B \end{array}$$

The $\Diamond$ and $\Box$ (and their past-time counterparts) can be derived from the $\mathcal{U}$ and $\mathcal{W}$ operators ($\mathcal{S}$ and $\mathcal{Z}$) respectively as follows:

$$\begin{array}{lll} \Diamond A & \equiv & \mathbf{true}\,\mathcal{U}\, A \\ \Box A & \equiv & A \,\mathcal{W}\, \mathbf{false} \end{array}$$

As an interpretation consists of a model/state-index pair, we say that a wff is satisfied in a particular model, at a particular state. The terminology is often extended to include a wff being *satisfied in a model*. A formula is satisfied in a model if it is satisfied in that model, at the beginning of time.

## 3 A New Approach to Temporal Resolution

When developing proof methods for non-classical logics, such as modal and temporal logics, it is natural to investigate extending classical methods into these domains. As the classical resolution rule is of the form

$$\frac{\begin{array}{c} \vdash A \lor p \\ \vdash B \lor \neg p \end{array}}{\vdash A \lor B}$$

one might ask whether such an inference rule can be used directly in temporal logics. Unfortunately, the use of such a rule in temporal logics has two main problems:

1. Can $\Box(A \lor p)$ and $\bigcirc(B \lor \neg p)$ be resolved? If so, what is the resolvent?

   This problem is characteristic of those found when resolution is required between complementary literals occurring in different temporal contexts.

2. An obvious temporal resolution rule, extended directly from the classical rule, would be

   $$\frac{\begin{array}{c} \vdash A \lor \Box p \\ \vdash B \lor \Diamond \neg p \end{array}}{\vdash A \lor B}$$

   However, sometimes the '$\Box$' is hidden, e.g.

   $$p \ \land \ \Box(x \Rightarrow \bigcirc(p \land y)) \ \land \ x \ \land \ \Box(y \Rightarrow \bigcirc(p \land x))$$

   This formula actually implies $\Box p$, but it is difficult to apply the resolution rule directly. Also, the resolution rule given above can not be applied to $\mathcal{U}$ and $\mathcal{W}$ formulae.

In (2), the $\Box p$ must be recognised if temporal resolution is to be applied between this formula and $\Diamond \neg p$. One solution to this is to provide a variety of resolution rules that 'recognise' various formulations of $\Box$. However, as there are a large number of formulae in which '$\Box$-like' formulae can occur, this approach is impractical.

The difficulties in recognising $\Box$ is one of the problems caused by the interaction between the $\bigcirc$ and $\Box$ operators. This interaction involves induction and accounts for many of the problems relating to the production of proof techniques for temporal logics.

The approach outlined in this report is to rewrite arbitrary temporal formulae into *Separated Normal Form* (SNF). The method relies on the fact that, in SNF, the only future-time temporal operator that occurs is the *sometime* operator, '$\Diamond$'. For those literals that appear outside the scope of a $\Diamond$ operator, non-temporal resolution is applied; for those inside the scope of a $\Diamond$ operator, temporal resolution is applied.

The non-temporal resolution rule is only applied between non-temporal formulae that refer to the same state. Thus, the problem of applying classical resolution rules across temporal contexts is avoided. By the definition and derivation of SNF, any two (non-temporal) literals appearing in a formula refer to the same state.

Temporal resolution is applied to formulae containing the '$\Diamond$' operator, such as $\Diamond q$. In this case, the formula can be resolved with a set of formulae, $S$, which, when satisfied, imply that $q$ will never be satisfied (i.e. $\Box\neg q$). As $\Diamond q$ guarantees that $q$ must be satisfied at some stage in the future, the resolution rule derives the constraint that the formulae in $S$ can never satisfied while $\Diamond q$ is outstanding. Thus, the second problem described above is partially solved in that once a hidden $\Box$-formulae is found it can be resolved with a complementary $\Diamond$-formula. The actual search for the formulae that represent the $\Box$-formula now becomes a search for formulae on which to apply the temporal resolution rule, and this search can be implemented using graph-theoretic techniques (see §8).

The process of applying temporal and non-temporal resolution rules to a set of formulae in SNF eventually terminates. On termination, either **false** has been derived, showing that the formula is unsatisfiable, or the formula is satisfiable. As in classical resolution, simplification and subsumption procedures are applied throughout the process.

## 4 Separated Normal Form

Gabbay has shown [Gabbay, 1987a] that for a linear temporal logic of the form described above, arbitrary formulae can be *separated* into their past-time, present and future-time components. We use this result to develop our normal form, called *Separated Normal Form* (SNF).

A temporal formula in SNF is of the form

$$\Box \bigwedge_{i=1}^{n} (P_i \Rightarrow F_i).$$

Here, each $P_i$ is a *strict* past-time temporal formula and each $F_i$ is a (non-strict) future-time formula. Each of the '$P_i \Rightarrow F_i$' (called *rules*) is further restricted to be of one the following

$$\begin{array}{llll}
\bullet\,\mathbf{false} & \Rightarrow & \bigvee_{k=1}^{r} l_k & \text{(an } \textit{initial}\ \Box\text{-rule)} \\
\circledcirc \bigwedge_{j=1}^{m} l_j & \Rightarrow & \bigvee_{k=1}^{r} l_k & \text{(a } \textit{global}\ \Box\text{-rule)} \\
\bullet\,\mathbf{false} & \Rightarrow & \Diamond l & \text{(an } \textit{initial}\ \Diamond\text{-rule)} \\
\circledcirc \bigwedge_{j=1}^{m} l_j & \Rightarrow & \Diamond l & \text{(a } \textit{global}\ \Diamond\text{-rule)}
\end{array}$$

where each $l_j$, $l_k$, or $l$ is a literal.

Sets of rules in SNF incorporate the following restriction. For each literal, $l$, there are at most two rules with a single occurrence of $l$ (i.e., not as part of a disjunction) as their future-time component. If there are two such rules, one must be initial, while the other must be global. This ensures that there is, at most, one rule defining the initial value of $l$ and, at most, one rule constraining $l$ after the beginning of time.

## 5 Rewriting Formulae into SNF

In this section, we show how any arbitrary formula of our temporal logic can be rewritten as an equivalent formula in SNF. Rather than give the full set of transformation rules used to translate a temporal formula in this way, a few selected transformations, together with an overview of the translation process, will be presented. (A more complete description of the transformation rules will be given in the full paper — see also [Fisher and Noël, 1992] for similar rules.)

The transformation rules are given as rewrite rules over sets of formulae. In this section, we only consider the translation of formulae such as

$$\Box \bigwedge_{i=1}^{n} (P_i \Rightarrow F_i),$$

where $P_i$ is a (strict) past-time formula and $F_i$ is a future-time formula, into SNF. The separation that was necessary to reach this form will not be considered here as it is described elsewhere [Gabbay, 1987a; Barringer *et al.*, 1989].

The translation of the above form into SNF initially involves removing all past-time temporal operators (apart from one level of last-time operators) from the $P_i$ and all future-time operators (apart from, at most, one '$\Diamond$' operator) from the $F_i$. After these operators have been removed, the remaining formulae must be rewritten to the correct conjunctive or disjunctive form (as described in §4).

The transformations are carried out in the context of a '$\Box$' operator, i.e., a rewrite rule such as

$$\{A \Rightarrow B\} \longrightarrow \left\{ \begin{array}{lll} C & \Rightarrow & D \\ E & \Rightarrow & F \end{array} \right\}$$

is to be read as a translation from a formula such as $\Box(A \Rightarrow B)$ to $\Box((C \Rightarrow D) \wedge (E \Rightarrow F))$.

During the transformation process, characterised by the rewrite rules, new propositions may be introduced. These are propositions that have not previously appeared in the formula and are represented by the symbols $x$, $y$, $z$, etc.

### 5.1 Pushing Negations

Initially, some basic (simplifying) rewrites are applied. The first such set of rewrites removes $\Box$, $\blacklozenge$, and $\blacksquare$ operators and replaces them with their definitions in terms of $\mathcal{W}$, $\mathcal{S}$, and $\mathcal{Z}$, respectively. Next, rewrite rules that 'push' negation operators to the propositions, are applied. This ensures that all negations only appear when applied to propositions, and this translation is analogous to the translation into *Negation Normal Form* used in classical logics.

### 5.2 Removing Past-time Operators

Next, the removal of past-time operators such as $\mathcal{S}$ and $\mathcal{Z}$, and the removal of multiple last-time operators is considered. Three (simplified) rewrite rules that are used for this purpose are given as follows[1].

$$\{A\,\mathcal{S}\,B \Rightarrow F\} \longrightarrow \left\{ \begin{array}{lll} \circledcirc(B \vee (A \wedge x)) & \Leftrightarrow & x \\ \bullet\,\mathbf{true} & \Rightarrow & \neg x \vee F \end{array} \right\}$$

$$\{A\,\mathcal{Z}\,B \Rightarrow F\} \longrightarrow \left\{ \begin{array}{lll} \bullet(B \vee (A \wedge y)) & \Leftrightarrow & y \\ \bullet\,\mathbf{true} & \Rightarrow & \neg y \vee F \end{array} \right\}$$

$$\{\mathcal{L}_1\mathcal{P}(\mathcal{L}_2 A) \Rightarrow F\} \longrightarrow \left\{ \begin{array}{lll} \mathcal{L}_2 A & \Rightarrow & z \\ \mathcal{L}_1\mathcal{P}(z) & \Rightarrow & F \end{array} \right\}$$

[1]Here, $A \Leftrightarrow B$ is used as a shorthand for the two rules, $A \Rightarrow B$ and $(\neg A) \Rightarrow (\neg B)$.

In the first two rules, a new proposition is introduced to represent the formula being replaced and its value is linked to the fixpoint representing the formula. This is, effectively, the same process used in QPTL [Wolper, 1982] to represent fixpoint operations using quantifiers [Banieqbal and Barringer, 1987], but, because of the finite nature of the past, only one type of fixpoint is used. In the last rule, $\mathcal{L}_1$ and $\mathcal{L}_2$ represent arbitrary last-time operators.

### 5.3 Removing Future-time Operators

To remove the future-time operators, $\mathcal{U}$, $\mathcal{W}$, and $\bigcirc$, and to simplify $\Diamond$-formulae so that each $\Diamond$ operator is only applied to a literal, rather than a general temporal formula, the following rewrite rules are applied. Note that the $\mathcal{F}$ used here represents a general future-time temporal context.

$$\{P \Rightarrow \mathcal{F}(A\,\mathcal{W}\,B)\} \longrightarrow \left\{ \begin{array}{lcl} P & \Rightarrow & \mathcal{F}(B \vee (A \wedge x)) \\ \circledcirc x & \Leftrightarrow & B \vee (A \wedge x) \end{array} \right\}$$

$$\{P \Rightarrow \mathcal{F}(A\,\mathcal{U}\,B)\} \longrightarrow \left\{ \begin{array}{lcl} P & \Rightarrow & \mathcal{F}(B \vee (A \wedge y)) \\ \circledcirc y & \Leftrightarrow & B \vee (A \wedge y) \\ P & \Rightarrow & \mathcal{F}(\Diamond B) \end{array} \right\}$$

$$\{P \Rightarrow \mathcal{F}(\bigcirc A)\} \longrightarrow \left\{ \begin{array}{lcl} \circledcirc z & \Leftrightarrow & A \\ P & \Rightarrow & \mathcal{F}(z) \end{array} \right\}$$

$$\{P \Rightarrow \mathcal{F}(\Diamond A)\} \longrightarrow \left\{ \begin{array}{lcl} \bullet\,\mathbf{true} & \Rightarrow & A \vee \neg z \\ \bullet\,\mathbf{true} & \Rightarrow & \neg A \vee z \\ P & \Rightarrow & \mathcal{F}(\Diamond z) \end{array} \right\}$$

Again, the technique of introducing a new proposition symbol to represent a particular formula is used. As the $\Box$ operator represents a maximal fixpoint, the $\mathcal{W}$ operator (which is also a maximal fixpoint) can be translated directly into its fixpoint definition. However, a combination of a fixpoint definition and '$\Diamond$' operator must be used to represent the $\mathcal{U}$ operator, which is a minimal fixpoint. The third rule removes all occurrences of $\bigcirc$ operators, and the final transformation rule is used if a $\Diamond$ operator is not applied to a literal.

### 5.4 Rewriting Non-Temporal Formulae

Having removed all the operators that are not required in SNF, further rewrite rules can be used to translate the remaining rules into the appropriate form. This involves ensuring that all past-time components are either '$\bullet$**false**' or '$\circledcirc$' applied to a conjunction of literals, and that all future-time components are either a disjunction of literals, or '$\Diamond l$', where $l$ is a literal.

As examples, we give some of the transformation rules used to manipulate last-time formulae into the appropriate form. Notice that *weak* last-time operators ('$\bullet$') are removed except when applied to **false**.

$$\{\bullet A \Rightarrow F\} \longrightarrow \left\{ \begin{array}{lcl} \circledcirc A & \Rightarrow & F \\ \bullet\,\mathbf{false} & \Rightarrow & F \end{array} \right\}$$

$$\left\{ \begin{array}{lcl} \circledcirc A & \Rightarrow & F \\ \circledcirc B & \Rightarrow & F \end{array} \right\} \longrightarrow \{\circledcirc (A \vee B) \Rightarrow F\}$$

$$\{\circledcirc A \wedge \bullet B \Rightarrow F\} \longrightarrow \{\circledcirc (A \wedge B) \Rightarrow F\}$$

$$\{\circledcirc A \vee \bullet B \Rightarrow F\} \longrightarrow \{\bullet (A \vee B) \Rightarrow F\}$$

Similar rules are used to transform future-time components into the appropriate disjunctive form.

This concludes our brief overview of the rewrite rules used for translating formulae into SNF. There are, however, some obvious simplifications that can be carried out on the rules constructed using the above rewrites. Discussion of such simplifications will be deferred until §6.2.

Finally, note that any well-formed formula in our logic can be translated to an equivalent set of rules in SNF. As an arbitrary formula in the logic can be separated into past, present, and future components [Gabbay, 1987a], the translations described above can each be shown to preserve satisfiability. Though the proof of this theorem is relatively straightforward, it does require the use of quantified propositional temporal logic [Wolper, 1982] as it must be shown that a formula in which a new proposition has been introduced is unsatisfiable if, and only if, the original formula is unsatisfiable.

## 6 The Resolution Procedure

Given a set of rules in SNF, both non-temporal and temporal resolution rules can be applied. Their application and effect is described in the following sections.

### 6.1 Non-Temporal Resolution

The non-temporal resolution rule is, essentially, the classical resolution rule and is only applied to $\Box$-rules; $\Diamond$-rules are processed using the temporal resolution rule described in §6.3.

The non-temporal resolution rule used here can again be expressed as a rewrite rule:

$$\boxed{\left\{ \begin{array}{lcl} P & \Rightarrow & B \vee a \\ Q & \Rightarrow & C \vee \neg a \end{array} \right\} \longrightarrow \left\{ \; P \wedge Q \;\Rightarrow\; B \vee C \; \right\}}$$

As with classical resolution, various strategies for the application of this rule can be employed, and simplification rules can be used during, and after, its application.

### 6.2 Simplification

The basic simplification rules are

$$\begin{array}{lcl} \{\circledcirc\,\mathbf{false} \Rightarrow F\} & \longrightarrow & \{\,\} \\ \{P \Rightarrow \mathbf{true}\} & \longrightarrow & \{\,\} \end{array}$$

The removal of such rules is obvious, since $\circledcirc$**false** can never be satisfied, and $P \Rightarrow$ **true** is always satisfiable.

Similarly, subsumption rules may be applied. Along with the standard (classical) subsumption, the following rule for subsumption between SNF rules can be applied.

$$\left\{ \begin{array}{lcl} P & \Rightarrow & F \\ Q & \Rightarrow & G \end{array} \right\} \xrightarrow{\vdash P \Rightarrow Q,\ \vdash G \Rightarrow F} \{Q \Rightarrow G\}$$

This rewrite rule can only be applied if the conditions, $\vdash P \Rightarrow Q$ and $\vdash G \Rightarrow F$ are satisfied.

If, after simplification, a formula of the form

$$\circledcirc R \Rightarrow \mathbf{false}$$

has been derived, then the following rewrite rule can be invoked.

$$\{\circledcirc R \Rightarrow \mathbf{false}\} \longrightarrow \{\bullet\,\mathbf{true} \Rightarrow \neg R\}$$

If this rule is used, the new constraint must be rewritten into SNF and non-temporal resolution re-applied. The process is repeated until either all $\Box$-rules containing complementary literals have been processed, or one of the following rules has been derived.

$$\begin{array}{lrcl} a) & \bullet\,\mathbf{false} & \Rightarrow & \mathbf{false} \\ b) & \bullet\,\mathbf{true} & \Rightarrow & \mathbf{false} \\ c) & \circledcirc\,\mathbf{true} & \Rightarrow & \mathbf{false} \end{array}$$

If any of these formulae occur, then the original formula is unsatisfiable and the resolution process terminates.

### 6.3 Temporal Resolution

If the non-temporal resolution procedure described in §6.1 terminates, without **false** having been derived, temporal resolution can be applied.

Before applying temporal resolution, the following rewrite rule is applied to all global $\Box$-rules. (Note that this transformation is not strictly necessary, but simplifies the presentation of the temporal resolution rule.)

$$\left\{ \begin{array}{lcl} \circledcirc P & \Rightarrow & F \\ \circledcirc Q & \Rightarrow & G \end{array} \right\} \longrightarrow \{\circledcirc(P \wedge Q) \Rightarrow (F \wedge G)\} \qquad (\dagger)$$

The temporal resolution rule applies to one $\Diamond$-rule, such as $P \Rightarrow \Diamond\neg l$ and a set of global $\Box$-rules. Here, if $P$ is satisfied, then the rule is satisfiable unless the set of $\Box$-rules force $l$ to always be true. Thus, our approach is to, for every such set of rules, ensure that these, and $P$, are never satisfied at the same time.

The temporal resolution rule is characterised as follows.

$$\left\{ \begin{array}{lcl} \circledcirc A_0 & \Rightarrow & B_0 \\ \dots & & \dots \\ \circledcirc A_n & \Rightarrow & B_n \\ \mathcal{L}P & \Rightarrow & \Diamond\neg l \end{array} \right\}$$
$$|$$
$$\begin{array}{lll} \text{for all } 0 \le i \le n. & \vdash B_i \Rightarrow l & \text{and} \\ & \vdash B_i \Rightarrow \bigvee_{j=0}^{n} A_j & \text{and} \\ & \vdash (A_i \wedge l) \Rightarrow \bigvee_{k=0}^{n} B_k & \end{array}$$
$$\downarrow$$
$$\left\{ \begin{array}{lcl} \bullet\,\mathbf{true} & \Rightarrow & \neg P \vee \bigwedge_{i=0}^{n} \neg A_i \\ \mathcal{L}P & \Rightarrow & (\bigwedge_{i=0}^{n} \neg A_i)\,\mathcal{W}\,\neg l \end{array} \right\}$$

A set of rules that, together, imply that $l$ is always true, is termed a *loop* in $l$. The side conditions on the resolution rule ensure that the the set of rules $\{\circledcirc A_i \Rightarrow B_i \mid 1 \le i \le n\}$ form a loop in $l$. Note that even this rule is a simplification — the full rule is quite complex and does not require the application of rewrite (†) beforehand.

If a loop in $l$ can be found, and there is a rule such as $\mathcal{L}P \Rightarrow \Diamond\neg l$, then the following new rules are added:

1. $\bullet\,\mathbf{true} \Rightarrow \neg P \vee \bigwedge_{i=0}^{n} \neg A_i$
   This rule expresses the constraint that none of the conditions for entering a loop in $l$ must be allowed to occur at the same time as $P$ occurs.
2. $\mathcal{L}P \Rightarrow (\bigwedge_{i=0}^{n} \neg A_i)\,\mathcal{W}\,\neg l$
   This rule expresses the constraint that once $P$ has occurred, none of the conditions for entering a loop must be allowed to occur until the eventuality initiated by $P$ has been satisfied.

#### 6.3.1 Example

As an example, consider the set of rules representing the conjunction of the formulae described earlier, i.e.

$$p \wedge \Box(x \Rightarrow \bigcirc(p \wedge y)) \wedge x \wedge \Box(y \Rightarrow \bigcirc(p \wedge x))$$

and $\Diamond\neg p$. Once the formulae have been rewritten into SNF and non-temporal resolution has been attempted (none occurs), the set of rules is as follows.

$$\begin{array}{lrcl} 1. & \bullet\,\mathbf{false} & \Rightarrow & x \\ 2. & \bullet\,\mathbf{false} & \Rightarrow & p \\ 3. & \bullet\,\mathbf{false} & \Rightarrow & \Diamond\neg p \\ 4. & \circledcirc x & \Rightarrow & p \wedge y \\ 5. & \circledcirc y & \Rightarrow & p \wedge x \end{array}$$

Temporal resolution can be applied to rules 3, 4, and 5, as rules 4 and 5 form a loop in $p$. This generates the new constraint

$$\bullet\,\mathbf{false} \Rightarrow (\neg x \wedge \neg y)\,\mathcal{W}\,\neg p$$

which is rewritten in SNF as:

$$\begin{array}{lrcl} 6. & \bullet\,\mathbf{false} & \Rightarrow & \neg p \vee \neg x \\ 7. & \bullet\,\mathbf{false} & \Rightarrow & \neg p \vee \neg y \\ 8. & \bullet\,\mathbf{false} & \Rightarrow & \neg p \vee z \\ 9. & \circledcirc z & \Rightarrow & \neg p \vee \neg x \\ 10. & \circledcirc z & \Rightarrow & \neg p \vee \neg y \\ 11. & \circledcirc z & \Rightarrow & \neg p \vee z \end{array}$$

Applying non-temporal resolution on rules 6, 1, and 2, generates a contradiction.

## 7 Correctness of the Resolution Procedure

The soundness, completeness and termination of the resolution procedure have been established. Proofs will be given in the full paper.

## 8 Implementation

Though the temporal resolution rule is complete, its use introduces various practical problems. A naive search of all the possible subsets of rules that match the preconditions of the resolution rule would be prohibitively expensive, consequently we have developed a procedure for finding loops in a particular literal within sets of rules. Lack of space prevents us from giving a full exposition of the approach, but we will give a brief outline below.

For every rule of the form $P \Rightarrow \Diamond\neg l$, all rules that imply $l$ are collected together. This set of rules is then represented directly as an AND/OR graph, with each rule representing a set of edges, or as a standard state-graph. In either case, loops in $l$ correspond to terminal strongly connected components (SCCs) of the graph structure. These can be found through the use of a suitable version of Tarjan's algorithm. Using either representation, the complexity of the operation is exponential in the number of rules in the original set.

Once all the terminal SCCs of the graph have been found, temporal resolution can be applied to all the rules in these SCCs, with any new rules generated are added to the original set of rules. If no terminal SCC is found, the procedure moves on to processing the next $\Diamond$-rule.

## 9 Related Work

A great variety of resolution-based proof methods for modal logics have been developed [del Cerro, 1984; Gabbay, 1987b; Ohlbach, 1988; Enjalbert and del Cerro, 1989], yet few of these can be used for temporal logics that incorporate the '*next*' operator. Several resolution methods specific to temporal logics hove been developed. Venkatesh, [Venkatesh, 1986], develops a resolution method for a future-time fragment of propositional temporal logic using customised resolution and unwinding rules. Sakuragawa, [Sakuragawa, 1986] uses transformations similar to those described in this report to generate a normal form for temporal formulae, but does not develop specific proof rules. In [Abadi and Manna, 1990], a resolution method, based on non-clausal resolution, is developed and applied to both propositional and first-order temporal logics.

## 10 Further Work

A prototype system based on the approach described in this report has been implemented and initial results are encouraging. Investigations into heuristics for the application of both temporal and non-temporal resolution rules are under way, as are investigations into alternate graph-theoretic techniques for implementing temporal resolution.

We can extend the method described in this report to first-order temporal logics that have unvarying domains and that satisfy the separation property. This is current work and will form part of a future report. One point to note is that not all temporal logics allow separation, though most linear temporal logics do [Gabbay, 1987a].

Finally, the resolution method described in this paper can be used as part of a *backward-chaining* execution mechanism that complements the forward-chaining that is standard in METATEM, a framework for executable temporal logics [Barringer *et al.*, 1989].

## References


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

[Banieqbal and Barringer, 1987] B. Banieqbal and H. Barringer. Temporal Logic with Fixed Points. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proceedings of Colloquium on Temporal Logic in Specification*, pages 62–74, Altrincham, U.K., 1987. (Published in *Lecture Notes in Computer Science*, volume 398, Springer Verlag).

[Barringer *et al.*, 1989] H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens. METATEM: A Framework for Programming in Temporal Logic. In *Proceedings of REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, Mook, Netherlands, June 1989. (Published in *Lecture Notes in Computer Science*, volume 430, Springer Verlag).

[del Cerro, 1984] Luis Fariñas del Cerro. Resolution Modal Logics. In *Proceedings of Advanced NATO Study Institute on Logics and Models for Verification and Specification of Concurrent Systems*, pages 46–78, La Colle-sur-Loup, France, October 1984.

[Enjalbert and del Cerro, 1989] P. Enjalbert and L. Fariñas del Cerro. Modal Resolution in Clausal Form. *Theoretical Computer Science*, 65:1–33, 1989.

[Fisher and Noël, 1992] M. Fisher and P. Noël. Transformation and Synthesis in METATEM – Part I: Propositional METATEM. Technical Report UMCS-92-2-1, Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, U.K., February 1992.

[Gabbay, 1987a] Dov Gabbay. Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proceedings of Colloquium on Temporal Logic in Specification*, pages 402–450, Altrincham, U.K., 1987. (Published in *Lecture Notes in Computer Science*, volume 398, Springer Verlag).

[Gabbay, 1987b] Dov Gabbay. Modal and Temporal Logic Programming. In A. Galton, editor, *Temporal Logics and their Applications*, pages 121–168. Academic Press, 1987.

[Ohlbach, 1988] Hans-Jürgen Ohlbach. A Resolution Calculus for Modal Logics. *Lecture Notes in Computer Science*, 310:500–516, May 1988.

[Sakuragawa, 1986] T. Sakuragawa. Temporal Prolog. Technical report, Research Institute for Mathematical Sciences, Kyoto University, 1986. to appear in Computer Software.

[Vardi and Wolper, 1986] M. Y. Vardi and P. Wolper. Automata-theoretic Techniques for Modal Logics of Programs. *Journal of Computer and System Sciences*, 32(2):183–219, April 1986.

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

[Wolper, 1982] Pierre Wolper. *Synthesis of Communicating Processes from Temporal Logic Specifications*. PhD thesis, Stanford University, 1982.

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