A Normal Form for First-Order Temporal Formulae

Michael Fisher*

Department of Computer Science,
University of Manchester
Manchester, U.K.

EMAIL: michael@cs.man.ac.uk

Abstract

In this paper a normal form for formulae of a first-order temporal logic is described. This normal form, called First-Order Separated Normal Form (SNF_f), forms the basis of both a temporal resolution method [5] and a family of executable temporal logics [2]. A first-order temporal logic, based on a linear discrete model structure, is introduced and the procedure for transforming an arbitrary formula of this logic to SNF_f is described. The transformation process not only preserves satisfiability but also ensures that any model of the transformed formula is a model of the original one. These properties ensure that the transformation into SNF_f has applications in both theorem proving and execution.

1 Introduction

One of the barriers to the development of clausal proof methods for modal and temporal logics has been the difficulties encountered in producing a normal form for such formulae. The main problem is that quantifiers and modal or temporal operators cannot be arbitrarily interchanged. For example, $\Diamond \forall x.\, p(x)$ is not, in general, equivalent to $\forall x.\, \Diamond p(x)$.

The technique of *renaming* has been used to preserve the structure of classical formulae and to improve the efficiency of deduction processes which apply to formulae in clausal form [7, 3]. It can be seen that, by renaming selected subformulae, modal and temporal formulae can be transformed whilst retaining their structure. For example, $\Diamond \forall x.\, p(x)$ can be transformed into the 'equivalent' formula, $\Diamond q \wedge \Box(q \Leftrightarrow \forall x.\, p(x))$, where x is a new predicate symbol. In this paper, this observation is used and extended to provide a transformation process, based on renaming, for transforming temporal formulae into a normal form. By using an appropriate normal form, new (renaming) subformulae, such as $\Box(q \Leftrightarrow \forall x.\, p(x))$, can also be transformed into the normal form.

The normal form used in this paper, called First-Order Separated Normal Form (SNF_f), is derived from the Propositional Separated Normal Form (SNF) used in [5]. Formulae in SNF_f are represented using a minimal set of temporal operators, in a form to which both resolution and execution techniques can be applied.

In §2 the temporal logic that will be used is described, with the syntax and semantics of the first-order temporal logic (called FML) being given in §2.1 and §2.2 respectively. In §3,

*This work was partially supported both by ESPRIT under Basic Research Action 3096 (SPEC), and by SERC under Research Grant GR/H/18449.

the normal form used for formulae of FML is outlined, reviewing propositional SNF in §3.1, and defining first-order SNF (SNF_f) in §3.2.

In §4, the procedure for transforming an arbitrary formula of FML into SNF_f is described. This procedure includes renaming transformations to remove quantifiers from inside temporal contexts and renaming transformations to eliminate certain temporal operators. The justifications for this second type of transformation are given elsewhere [5, 4] as they are analogous to the transformations used in the propositional case.

The applications of this transformation into SNF_f, particularly the temporal resolution procedure described in [5] and the direct execution of temporal formulae, are outlined in §5. Finally, in §6, the main results are summarised and further work in this area is outlined.

2 A First-Order Linear Temporal Logic

The first-order temporal logic, based on discrete, linear models with finite past and infinite future, is called FML, and its syntax and semantics are outlined below. Note that, FML incorporates the *constant domain* assumption with regard to quantification over temporal formulae.

2.1 Syntax

Well-formed formulae of FML (WFF_f) are generated from the following symbols.

- A set, $\mathcal{L}_p$, of *predicate symbols* represented by strings of lowercase alphabetic characters.

  Associated with each predicate symbol, p, is a non-negative integer, *arity(p)*.
  (Note that predicates with arity 0 are termed *propositions*.)

- Classical connectives, $\neg$, $\vee$, $\wedge$, $\Rightarrow$, and $\Leftrightarrow$.

- Future-time temporal operators, including unary operators $\bigcirc$, $\Diamond$ and $\Box$, and binary operators $\mathcal{U}$ and $\mathcal{W}$.

- Past-time temporal operators, including unary operators ◎, ●, ♦ and ■, and binary operators $\mathcal{S}$ and $\mathcal{Z}$.

- A set, $\mathcal{L}_v$, of *variable symbols*, x, y, z, etc.

- A set, $\mathcal{L}_c$, of *constant symbols*, a, b, c, etc.

- A set, $\mathcal{L}_f$, of *function symbols*, f, g, h, etc.

  Associated with each function symbol, f, is an arity, given by *arity(f)*.

- Quantifiers, $\forall$ and $\exists$.

- '(' and ')' which are, as usual, used to avoid ambiguity.

The set of *terms*, $\mathcal{L}_t$, contains those strings that are either constants or variables, or are generated by application of function symbols to other elements of $\mathcal{L}_t$. $\mathcal{L}_t$ is defined as follows.

1. Both $\mathcal{L}_v$ and $\mathcal{L}_c$ are subsets of $\mathcal{L}_t$.
2. If $t_1, \ldots, t_n$ are in $\mathcal{L}_t$, and f is a function symbol of arity n, then $f(t_1, \ldots, t_n)$ is in $\mathcal{L}_t$.

The set of well-formed formulae of FML (WFF_f) is defined as follows.

1. If $t_1, \ldots, t_n$ are in $\mathcal{L}_t$, and p is a predicate symbol of arity n, then $p(t_1, \ldots, t_n)$ is in WFF_f.
2. if A and B are in WFF_f, then the following are in WFF_f

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

3. If A is in WFF_f and v is in $\mathcal{L}_v$, then $\exists v.\ A$ and $\forall v.\ A$ are both in WFF_f.

Sub-classifications of WFF_f are defined as follows. A *literal* is defined as either a predicate applied to an appropriate term, or the negation of such a predicate.
A *State-formula* is either a literal or a non-temporal 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 $\neg A$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $\exists v.\ A$, $\forall v.\ A$, $A \mathcal{U} B$, $A \mathcal{W} B$, $\bigcirc A$, $\Diamond A$, 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 $\neg A$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $\exists v.\ A$ and $\forall v.\ A$ are all strict past-time formulae.

2.2 Semantics

The basic models of FML are discrete, linear structures with finite past and infinite future. To this structure a domain, $\mathcal{D}$, and mappings from elements of the language to denotations are added. Thus the full model structure for FML is

$$\mathcal{M} = \langle \sigma, \mathcal{D}, \pi_c, \pi_f, \pi_p \rangle$$

where

- σ, a sequence of states $s_0, s_1, s_2, s_3, \ldots$,
- $\mathcal{D}$ is the object-level domain,
- π_c is a map from $\mathcal{L}_c$ to $\mathcal{D}$,
- π_f is a map from $\mathcal{L}_f$ to $\mathcal{D}^n \rightarrow \mathcal{D}$, where n is the arity of f, and,
- π_p is a map from $\mathbf{N} \times \mathcal{L}_p$ to $\mathcal{D}^n \rightarrow \{T, F\}$.

Thus, for a particular state s, and a particular predicate p of arity n, $\pi_p(s, p)$ represents a map from n-tuples of elements of $\mathcal{D}$ to T or F. Note that the *constant domain* assumption is used, i.e. that $\mathcal{D}$ is constant for every state.

Next, *variable assignments* and *term assignments* are defined. A variable assignment is a mapping from $\mathcal{L}_v$ to elements of $\mathcal{D}$. Given a variable assignment, V, and the valuation functions, π_c and π_f, associated with a particular model structure, a term assignment $\tau_{v\pi}$ is a mapping from $\mathcal{L}_t$ to $\mathcal{D}$, defined inductively as follows.

- **if** $c \in \mathcal{L}_c$ **then** $\tau_{v\pi}(c) = \pi_c(c)$
- **if** $f \in \mathcal{L}_f$ **then** $\tau_{v\pi}(f(t_1, \ldots, t_n)) = \pi_f(f)(\tau_{v\pi}(t_1), \ldots, \tau_{v\pi}(t_n))$ [*where arity*$(f) = n$]
- **if** $v \in \mathcal{L}_v$ **then** $\tau_{v\pi}(c) = V(c)$
- **if** $p \in \mathcal{L}_p$ **then** $\tau_{v\pi}(p(t_1, \ldots, t_n)) = p(\tau_{v\pi}(t_1), \ldots, \tau_{v\pi}(t_n))$ [*where arity*$(p) = n$]
- **if** $t = \mathrm{OP}\, t'$ **then** $\tau_{v\pi}(t) = \mathrm{OP}\, \tau_{v\pi}(t')$ [*where* $\mathrm{OP} \in \{\neg, \bigcirc, \bullet, \circledcirc, \Diamond, \Box, \blacklozenge, \blacksquare\}$]
- **if** $t = t' \,\mathrm{OP}\, t''$ **then** $\tau_{v\pi}(t) = \tau_{v\pi}(t') \,\mathrm{OP}\, \tau_{v\pi}(t'')$ [*where* $\mathrm{OP} \in \{\wedge, \vee, \Rightarrow, \mathcal{S}, \mathcal{Z}, \mathcal{U}, \mathcal{W}\}$]

The semantics of a well-formed formula is given with respect to a model structure, a state at which the temporal formula is to be interpreted, and a variable assignment. The satisfaction relation, $\models$, relates such tuples to well-formed formulae.

The semantics of a predicate is given by the truth value of the predicate application as defined in the model:

$$\langle \mathcal{M}, s_i, V \rangle \models \; p(x_1, \ldots, x_n) \text{ iff } \pi_p(i, p)(\tau_{v\pi}(x_1), \ldots, \tau_{v\pi}(x_n)) = T.$$

Two special predicate symbols are identified. In every model, and at every state, s_i, $\pi_p(i, \mathit{true})$ is a function that returns T no matter what its arguments are. Similarly, $\pi_p(i, \mathit{false})$ returns F. When *true* and *false* have arity 0, **true** and **false**, respectively, will be used.

The semantics of the standard propositional connectives is as in classical logic, e.g.,

$$\langle \mathcal{M}, i, V \rangle \models \varphi \vee \psi \text{ iff } \langle \mathcal{M}, i, V \rangle \models \varphi \text{ or } \langle \mathcal{M}, i, V \rangle \models \psi$$

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

$$\begin{aligned}
&\langle \mathcal{M}, s_i, V \rangle \models \bigcirc \varphi \quad \text{iff} \quad \langle \mathcal{M}, s_{i+1}, V \rangle \models \varphi \\
&\langle \mathcal{M}, s_i, V \rangle \models \Diamond \varphi \quad \text{iff} \quad \text{there exists a } j \geq i \text{ such that } \langle \mathcal{M}, s_j, V \rangle \models \varphi \\
&\langle \mathcal{M}, s_i, V \rangle \models \Box \varphi \quad \text{iff} \quad \text{for all } j \geq i \text{ then } \langle \mathcal{M}, s_j, V \rangle \models \varphi.
\end{aligned}$$

The informal semantics of these operators is as follows: $\bigcirc \varphi$ means that φ must be true in the *next* state; $\Diamond \varphi$ means that φ must be true at *some* state in the future; $\Box \varphi$ means that φ must be true at *all* states in the future.

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

$$\begin{aligned}
\langle \mathcal{M}, s_i, V \rangle \models \varphi \,\mathcal{U}\, \psi \text{ iff } &\text{ there exists a } k \geq i \text{ such that } \langle \mathcal{M}, s_k, V \rangle \models \psi \\
&\text{and for all } i \leq j < k \text{ then } \langle \mathcal{M}, s_j, V \rangle \models \varphi \\
\langle \mathcal{M}, s_i, V \rangle \models \varphi \,\mathcal{W}\, \psi \text{ iff } &\text{ for all } j \geq i \text{ then } \langle \mathcal{M}, s_j, V \rangle \models \varphi \\
&\text{or } \langle \mathcal{M}, s_i, V \rangle \models \varphi \,\mathcal{U}\, \psi
\end{aligned}$$

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 is given as follows:

$$\langle \mathcal{M}, s_i, V \rangle \models \bullet \varphi \text{ iff } i = 0 \text{ or } \langle \mathcal{M}, s_{i-1}, V \rangle \models \varphi$$
$$\langle \mathcal{M}, s_i, V \rangle \models \circledcirc \varphi \text{ iff } i > 0 \text{ and } \langle \mathcal{M}, s_{i-1}, V \rangle \models \varphi$$
$$\langle \mathcal{M}, s_i, V \rangle \models \blacklozenge \varphi \text{ iff there exists } j \text{ such that } 0 \leq j < i \text{ and } \langle \mathcal{M}, s_j, V \rangle \models \varphi$$
$$\langle \mathcal{M}, s_i, V \rangle \models \blacksquare \varphi \text{ iff for all } j \text{ such that } 0 \leq j < i \text{ then } \langle \mathcal{M}, s_i, V \rangle \models \varphi$$

Note that, in contrast to the future-time operators, the $\blacklozenge$ and $\blacksquare$ 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 φ, $\circledcirc \varphi$ is false at the beginning of time, while $\bullet \varphi$ is true at the beginning of time. In particular, $\bullet$ **false** is only true when interpreted at the beginning of time; otherwise it is false. Note that the equivalence, $\circledcirc \neg \varphi \equiv \neg \bullet \varphi$, relates these operators.

The semantics for the binary past-time operators $\mathcal{S}$ and $\mathcal{Z}$ relates to that for $\mathcal{U}$ and $\mathcal{W}$ just as the unary past-time operators related to the unary future-time operators. Finally, the semantics of quantifiers is defined as follows.

$$\langle \mathcal{M}, s, V \rangle \models \forall x.\ \varphi \text{ iff } \text{ for all } d \in \mathcal{D}.\ \langle \mathcal{M}, s, V \dagger [x \mapsto d] \rangle \models \varphi$$
$$\langle \mathcal{M}, s, V \rangle \models \exists x.\ \varphi \text{ iff } \text{ there exists } d \in \mathcal{D}. \text{ such that } \langle \mathcal{M}, s, V \dagger [x \mapsto d] \rangle \models \varphi$$

As the interpretation consists of a triple, comprising model, state, and assignment components, a well-formed formula, A, is *satisfied* in a particular model, $\mathcal{M}$, at the beginning of time, s_0, under a particular variable assignment, V, if $\langle \mathcal{M}, s_0, V \rangle \models A$. A formula is *satisfiable* if it is satisfied in some model, and for some variable assignment, at the beginning of time. Note that the more standard definition of satisfiability, where the formula is interpreted in *some* state, can be translated to the definition given above by using the following equivalence.

$$\exists \mathcal{M}.\ \exists V.\ \exists s.\ \langle \mathcal{M}, s, V \rangle \models A \qquad \text{if, and only if,} \qquad \exists \mathcal{M}.\ \exists V.\ \langle \mathcal{M}, s_0, V \rangle \models \Diamond A$$

In this paper, closed formulae, i.e. formulae containing no free variables, will mainly be used. In this case, the empty mapping, [ ], is used as the initial variable assignment.

3 A Normal Form for Temporal Formulae

The approach outlined in this paper is to rewrite arbitrary temporal formulae into *First-Order Separated Normal Form* (SNF_f). Before defining SNF_f, the similar, but simpler, propositional normal form will be reviewed.

3.1 Propositional Separated Normal Form

A temporal formula in the basic (propositional) Separated Normal Form (SNF) described in [5] 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

$$\bullet\,\mathbf{false} \;\Rightarrow\; \bigvee_{k=1}^{r} l_k \qquad \text{(an } \textit{initial} \,\Box\text{-rule)}$$

$$\circledcirc \bigwedge_{j=1}^{m} l_j \;\Rightarrow\; \bigvee_{k=1}^{r} l_k \qquad \text{(a } \textit{global} \,\Box\text{-rule)}$$

$$\bullet\,\mathbf{false} \;\Rightarrow\; \Diamond l \qquad \text{(an } \textit{initial} \,\Diamond\text{-rule)}$$

$$\circledcirc \bigwedge_{j=1}^{m} l_j \;\Rightarrow\; \Diamond l \qquad \text{(a } \textit{global} \,\Diamond\text{-rule)}$$

where each l_j, l_k, or l is a literal. In previous papers, we have shown how propositional temporal formulae can be rewritten into this form [4, 5].

3.2 First-Order Separated Normal Form

In defining first-order SNF (SNF_f), the basic rule form is again required to be '$P_i \Rightarrow F_i$'. However, now the restrictions on P_i and F_i are slightly different:

- each P_i is now a *non-strict* past-time temporal formula;
- each F_i is again a (non-strict) future-time formula.

This restriction would not make any difference in propositional SNF as, in the propositional case, state-formulae on the left-hand side of the implication can always be moved across to the right. However, as will be seen later, the ability to be able to refer to state-formulae on the left-hand side of this implication is required for SNF_f.

The general form of a formula in SNF_f is

$$\Box \bigwedge_{i=1}^{n} \forall \bar{x}_i.\ (\forall \bar{y}_i.\ P_i(\bar{x}_i, \bar{y}_i)) \;\Rightarrow\; (\exists \bar{z}_i.\ F_i(\bar{x}_i, \bar{z}_i))$$

with the above restrictions of P_i and F_i. Here, '$\bar{x}_i$' represents a vector of variables, $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$, so $\forall \bar{x}_i$ represents

$$\forall x_{i_1}.\ \forall x_{i_2}.\ \ldots\ \forall x_{i_m}$$

and $P_i(\bar{x}_i, \bar{y}_i)$ represents

$$P_i(x_{i_1}, x_{i_2}, \ldots, x_{i_m}, y_{i_1}, y_{i_2}, \ldots, y_{i_k}).$$

Now, in SNF_f, each rule is further restricted to be one of the following.

$$\forall \bar{x}. \quad [(\forall \bar{y}. \bullet \mathbf{false} \wedge \bigwedge_{b=1}^{h} l_b(\bar{x}, \bar{y})) \quad \Rightarrow \quad \exists \bar{z}. \bigvee_{j=1}^{r} m_j(\bar{x}, \bar{z})] \qquad \text{(an } \textit{initial} \ \Box\text{-rule)}$$

$$\forall \bar{x}. \quad [(\forall \bar{y}. (\circledcirc \bigwedge_{a=1}^{g} k_a(\bar{x}, \bar{y})) \wedge \bigwedge_{b=1}^{h} l_b(\bar{x}, \bar{y})) \quad \Rightarrow \quad \exists \bar{z}. \bigvee_{j=1}^{r} m_j(\bar{x}, \bar{z})] \qquad \text{(a } \textit{global} \ \Box\text{-rule)}$$

$$\forall \bar{x}. \quad [(\forall \bar{y}. \bullet \mathbf{false} \wedge \bigwedge_{b=1}^{h} l_b(\bar{x}, \bar{y})) \quad \Rightarrow \quad \exists \bar{z}. \Diamond l(\bar{x}, \bar{z})] \qquad \text{(an } \textit{initial} \ \Diamond\text{-rule)}$$

$$\forall \bar{x}. \quad [(\forall \bar{y}. (\circledcirc \bigwedge_{a=1}^{g} k_a(\bar{x}, \bar{y})) \wedge \bigwedge_{b=1}^{h} l_b(\bar{x}, \bar{y})) \quad \Rightarrow \quad \exists \bar{z}. \Diamond l(\bar{x}, \bar{z})] \qquad \text{(a } \textit{global} \ \Diamond\text{-rule)}$$

where each k_a, l_b, m_j or l is a literal.

4 Rewriting Formulae into SNF_f

In this section, it will be shown how arbitrary well-formed formulae of FML can be transformed, using satisfiability preserving transformations, into SNF_f. For simplicity, this translation process will be split into two phases. The first transforms an arbitrary formula from WFF_f into rule form, i.e.,

$$\Box \bigwedge_{i=1}^{n} \forall \bar{x}_i.\ Q_i(\bar{x}_i) \ \Rightarrow \ G_i(\bar{x}_i)$$

where $Q_i(\bar{x}_i)$ is a past-time formula and $G_i(\bar{x}_i)$ is a future-time formula. The second phase translates a formula in this rule form into one in SNF_f.

Rather than giving the full set of transformation rules, only the transformations that are most fundamental to the production of SNF_f will be presented individually, together with an overview of the translation process. The proofs of the satisfiability preserving properties of the transformations used to remove temporal operators can be found in [4], while the correctness of the new transformations introduced in this paper follows from the theorem given below.

Theorem 1 *Given a formula, $\mathcal{A}(B(\bar{x})$, where $\mathcal{A}$ is a context[1] and B is a formula with free variables, $\bar{x}$, then let N be the formula*

$$\mathcal{A}(b(\bar{x})) \ \wedge \ \Box \forall \bar{y}.\ (b(\bar{y}) \ \Leftrightarrow \ B(\bar{y}))$$

where 'b' is a new predicate symbol (i.e. b has not occurred in any previous formula). Given the above constraints, both the following statements hold.

1. *If $\mathcal{A}(B(\bar{x})$ is satisfiable, then N is satisfiable, i.e.*
   if $\exists \mathcal{M}.\ \exists V.\ \langle \mathcal{M}, s_0, V \rangle \models \mathcal{A}(B(\bar{x})$ *then* $\exists \mathcal{M}'.\ \exists V'.\ \langle \mathcal{M}', s_0, V' \rangle \models N$.

[1] A context is taken to be a function over WFF_f that embeds its argument at a given position in an enclosing formula. Various contexts will be used to represent subclasses of WFF_f.

2. *Any model for N is a model for $\mathcal{A}(B(\bar{x}))$, i.e.*

   if $\exists \mathcal{M}''.\ \exists V''.\ \langle \mathcal{M}'', s_0, V'' \rangle \models N$ *then* $\langle \mathcal{M}'', s_0, V'' \rangle \models \mathcal{A}(B(\bar{x}))$

The transformations required for rewriting a formula into SNF_f will now be introduced.

4.1 Generating Rule Form

An arbitrary temporal formula, A, is first transformed as follows.

$$A \longrightarrow \Box\, \forall \bar{x}.\ (\bullet\, \mathbf{false} \Rightarrow A)$$

This is a sound transformation as long as none of the variables in $\bar{x}$ occur in A.

Example The formula $\exists w.\ \Diamond\, \blacksquare p(w)$, can be replaced by

$$\Box\, \forall \bar{x}.\ (\bullet\, \mathbf{false} \Rightarrow \exists w.\ \Diamond\, \blacksquare p(w)) \tag{1}$$

Since satisfiability is 'anchored' at s_0, and since $\bullet\, \mathbf{false}$ can only be satisfied at s_0 then, to satisfy $\langle \mathcal{M}, s_0, V \rangle \models \Box\, \forall \bar{x}.\ (\bullet\, \mathbf{false} \Rightarrow A)$, A must be satisfied at s_0.

Next, all the past-time operators are removed from the right-hand side of the formula. This is achieved by renaming the past-time formulae with new predicates and adding the definition of these new predicates to the original formula. Using $\mathcal{F}$ to represent any arbitrary non-strict future-time context, and $B(\bar{v})$ to represent a past-time formula with free variables $\bar{v}$, the following transformation is applied.

$$\Box\, \forall \bar{x}.\ P(\bar{x}) \Rightarrow \mathcal{F}(B(\bar{v})) \longrightarrow \left\{ \begin{array}{llllll} & \Box\, \forall \bar{x}. & P(\bar{x}) & \Rightarrow & \mathcal{F}(b(\bar{v})) \\ \wedge & \Box\, \forall \bar{y}. & B(\bar{y}) & \Rightarrow & b(\bar{y}) \\ \wedge & \Box\, \forall \bar{y}. & \neg B(\bar{y}) & \Rightarrow & \neg b(\bar{y}) \end{array} \right\}$$

Example Given the formula (1) above, $\blacksquare p(w)$ can be replaced by a new predicate $q(w)$, defined in such a way that $q(y)$ is satisfied if, and only if, $\blacksquare p(y)$ is satisfied. Thus the new set of rules generated by this transformation is

$$\begin{array}{llllll} & \Box\, \forall \bar{x}. & \bullet\, \mathbf{false} & \Rightarrow & (\exists w.\ \Diamond q(w)) \\ \wedge & \Box\, \forall y. & \blacksquare p(y) & \Rightarrow & q(y) \\ \wedge & \Box\, \forall y. & \neg \blacksquare p(y) & \Rightarrow & \neg q(y) \end{array}$$

Thus, this transformation preserves satisfiability as each subformula (possibly containing free variables) is replaced by a new predicate (applied to those free variables) whose satisfiability is directly linked to that of the replaced subformula. For example, the defining formulae for the new predicate q above combine to give

$$\Box\, \forall y.\ (\blacksquare p(y) \Leftrightarrow q(y))$$

Thus, the value of $q(y)$ is always the same as that of $\blacksquare p(y)$. In particular, at the moment when $\blacksquare p(w)$ would have been evaluated, $q(w)$ has exactly the same value.

Similarly, any future-time temporal sub-formulae can be removed from the left-hand side of the formula, by using the renaming transformation:

$$\Box\,\forall\bar{x}.\ \mathcal{P}(A(\bar{v}))\ \Rightarrow\ F(\bar{x}) \longrightarrow \left\{ \begin{array}{lllll} & \Box\,\forall\bar{x}. & \mathcal{P}(a(\bar{v})) & \Rightarrow & F(\bar{x}) \\ \wedge & \Box\,\forall\bar{y}. & a(\bar{y}) & \Rightarrow & A(\bar{y}) \\ \wedge & \Box\,\forall\bar{y}. & \neg a(\bar{y}) & \Rightarrow & \neg A(\bar{y}) \end{array} \right\}$$

4.2 From Rule Form to SNF$_f$

After applying the above transformations, the remaining formulae are in rule form, i.e.

$$\Box\forall\bar{x}.\ (P(\bar{x})\ \Rightarrow\ F(\bar{x}))\,,$$

where $P(\bar{x})$ is a (non-strict) past-time formula and $F(\bar{x})$ is a (non-strict) future-time formula. The translation of this form into SNF$_f$ involves removing all past-time temporal operators (apart from one level of last-time operators) from $P(\bar{x})$, and removing all future-time operators (apart from, at most, one '$\Diamond$' operator) from the $F(\bar{x})$, re-ordering the quantifiers to be in the form required for SNF$_f$, and rewriting the remaining formulae into the correct conjunctive or disjunctive form (as described in §3.2).

A procedure for carrying out this translation from rule form to SNF$_f$ will now be described. Note that several of the transformations used, namely those involved in the removal of temporal operators, were introduced previously [5, 4]. The procedure described is certainly not the most efficient method of translating to SNF$_f$, however it has the advantage of being reasonably simple to explain.

Remove all embedded quantifiers from the right-hand side. Here, any embedded quantifiers are removed from the right-hand side of any rule (i.e., from $F(\bar{x})$). Thus, the following two transformations are used to ensure that either no quantifiers appear on the right-hand side of a rule, or that they only occur at the top level, e.g. $\Box\,\forall\bar{x}.\ P(\bar{x})\ \Rightarrow\ (\forall w.\ F(\bar{x}, w))$.

$$\Box\,\forall\bar{x}.\ P(\bar{x})\ \Rightarrow\ \mathcal{F}(\forall w.\ B(\bar{v}, w)) \longrightarrow \left\{ \begin{array}{lllll} & \Box\,\forall\bar{x}. & P(\bar{x}) & \Rightarrow & \mathcal{F}(b(\bar{v})) \\ \wedge & \Box\,\forall\bar{y}. & b(\bar{y}) & \Rightarrow & \forall w.\ B(\bar{y}, w) \\ \wedge & \Box\,\forall\bar{y}. & \neg b(\bar{y}) & \Rightarrow & \exists w.\ \neg B(\bar{y}, w) \end{array} \right\}$$

$$\Box\,\forall\bar{x}.\ P(\bar{x})\ \Rightarrow\ \mathcal{F}(\exists w.\ B(\bar{v}, w)) \longrightarrow \left\{ \begin{array}{lllll} & \Box\,\forall\bar{x}. & P(\bar{x}) & \Rightarrow & \mathcal{F}(b(\bar{v})) \\ \wedge & \Box\,\forall\bar{y}. & b(\bar{y}) & \Rightarrow & \exists w.\ B(\bar{y}, w) \\ \wedge & \Box\,\forall\bar{y}. & \neg b(\bar{y}) & \Rightarrow & \forall w.\ \neg B(\bar{y}, w) \end{array} \right\}$$

Example Given the formula $\Box\,\forall\bar{x}.\ P(\bar{x})\ \Rightarrow\ \Diamond(\forall z.\ \Box q(\bar{x}, z))$, the above transformations can be applied to give

$$\begin{array}{lllll} & \Box\,\forall\bar{x}. & P(\bar{x}) & \Rightarrow & \Diamond c(\bar{x}) \\ \wedge & \Box\,\forall\bar{y}. & c(\bar{y}) & \Rightarrow & \forall z.\ \Box q(\bar{y}, z) \\ \wedge & \Box\,\forall\bar{y}. & \neg c(\bar{y}) & \Rightarrow & \exists z.\ \neg\,\Box q(\bar{y}, z) \end{array}$$

Again, a subformula is replaced by a new predicate whose value mirrors that of the subformula. Thus, the predicate $c(\bar{y})$ above is satisfied if, and only if, $\forall z.\ \Box q(\bar{y}, z))$ is satisfied.

Remove all embedded quantifiers from the left-hand side. Here, a similar set of transformations is applied in order to ensure that there are no embedded quantifiers on the left-hand side of a rule, i.e., if a quantifier such as $\forall y$ occurs it will occur as

$$\Box\, \forall \bar{x}.\ (\forall y.\ P(\bar{x}, y)) \ \Rightarrow\ F(\bar{x})$$

The transformation rules used in this process are the duals of those given above. For example, embedded existential quantifiers can be removed using the following rule (where $\mathcal{P}$ is a general past-time context).

$$\Box\, \forall \bar{x}.\ \mathcal{P}(\exists w.\ B(\bar{v}, w)) \ \Rightarrow\ F(\bar{x}) \longrightarrow \left\{ \begin{array}{lllll} & \Box\, \forall \bar{x}. & \mathcal{P}(b(\bar{v})) & \Rightarrow & F(\bar{x}) \\ \wedge & \Box\, \forall \bar{y}. & \exists w.\ B(\bar{y}, w) & \Rightarrow & b(\bar{y}) \\ \wedge & \Box\, \forall \bar{y}. & \forall w.\ \neg B(\bar{y}, w) & \Rightarrow & \neg b(\bar{y}) \end{array} \right\}$$

Now that quantifiers only appear at the 'outer' levels of each rule, transformations aimed at removing various temporal operators can be applied. Initially, implications in either side of the rule are expanded into their definitions in terms of negation and disjunction, and all negations are pushed through the formula so that they apply only to predicates. Next transformations which remove certain temporal operators (those not required for SNF_f) and replace them by new predicates are applied. As these transformations are the obvious first order extensions of those described in earlier papers [5, 4], only a brief outline of their use will be given, and any justification for their correctness will be omitted.

Removing Past-time Temporal Operators. The operators ■ and ♦ can be rewritten in terms of the $\mathcal{Z}$ and $\mathcal{S}$ operators respectively. Next, the $\mathcal{Z}$ and $\mathcal{S}$ operators, and any multiple occurrences of last-time operators (represented below by $\mathcal{L}$), are removed. The following transformations are used for this purpose. (Note that, here, $\mathcal{P}$ represents a general (non-strict) past-time context, and that $A \Leftrightarrow B$ is used as an abbreviation for $(A \Rightarrow B) \wedge (\neg A \Rightarrow \neg B)$.)

$$\Box\, \forall \bar{x}.\ \mathcal{P}(A(\bar{v})\, \mathcal{S}\, B(\bar{v})) \ \Rightarrow\ F(\bar{x}) \longrightarrow \left\{ \begin{array}{lllll} & \Box\, \forall \bar{x}. & \mathcal{P}(s(\bar{v})) & \Rightarrow & F(\bar{x}) \\ \wedge & \Box\, \forall \bar{y}. & ⊚\,(B(\bar{y}) \vee (A(\bar{y}) \wedge s(\bar{y}))) & \Leftrightarrow & s(\bar{y}) \end{array} \right\}$$

$$\Box\, \forall \bar{x}.\ \mathcal{P}(A(\bar{v})\, \mathcal{Z}\, B(\bar{v})) \ \Rightarrow\ F(\bar{x}) \longrightarrow \left\{ \begin{array}{lllll} & \Box\, \forall \bar{x}. & \mathcal{P}(z(\bar{v})) & \Rightarrow & F(\bar{x}) \\ \wedge & \Box\, \forall \bar{y}. & ●\,(B(\bar{y}) \vee (A(\bar{y}) \wedge z(\bar{y}))) & \Leftrightarrow & z(\bar{y}) \end{array} \right\}$$

$$\Box\, \forall \bar{x}.\ \mathcal{P}(\mathcal{L}(A(\bar{v}))) \ \Rightarrow\ F(\bar{x}) \longrightarrow \left\{ \begin{array}{lllll} & \Box\, \forall \bar{x}. & \mathcal{P}(l(\bar{v})) & \Rightarrow & F(\bar{x}) \\ \wedge & \Box\, \forall \bar{y}. & \mathcal{L}(A(\bar{y})) & \Leftrightarrow & l(\bar{y}) \end{array} \right\}$$

Removing Future-time Temporal Operators With the future-time operators, all but the '$\Diamond$' temporal operator are removed and the transformations must ensure that each $\Diamond$ only applies to a literal. The $\Box$ operator can be reduced to $\mathcal{W}$, so the following transformations must be

used to remove the unwanted temporal operators.

$$\Box\,\forall\bar{x}.\ P(\bar{x}) \Rightarrow \mathcal{F}(A(\bar{v})\,\mathcal{W}\,B(\bar{v})) \longrightarrow \left\{ \begin{array}{l} \Box\,\forall\bar{x}.\ P(\bar{x}) \Rightarrow \mathcal{F}(B(\bar{v}) \vee (A(\bar{v}) \wedge w(\bar{v}))) \\ \wedge \\ \Box\,\forall\bar{y}.\ \circledcirc\, w(\bar{y}) \Leftrightarrow B(\bar{y}) \vee (A(\bar{y}) \wedge w(\bar{y})) \end{array} \right\}$$

$$\Box\,\forall\bar{x}.\ P(\bar{x}) \Rightarrow \mathcal{F}(A(\bar{v})\,\mathcal{U}\,B(\bar{v})) \longrightarrow \left\{ \begin{array}{l} \Box\,\forall\bar{x}.\ P(\bar{x}) \Rightarrow \\ \qquad \mathcal{F}((B(\bar{v}) \vee (A(\bar{v}) \wedge u(\bar{v}))) \wedge \Diamond B(\bar{v})) \\ \wedge \\ \Box\,\forall\bar{y}.\ \circledcirc\, u(\bar{y}) \Leftrightarrow B(\bar{y}) \vee (A(\bar{y}) \wedge u(\bar{y})) \end{array} \right\}$$

$$\Box\,\forall\bar{x}.\ P(\bar{x}) \Rightarrow \mathcal{F}(\bigcirc A(\bar{v})) \longrightarrow \left\{ \begin{array}{ll} & \Box\,\forall\bar{x}.\ P(\bar{x}) \Rightarrow \mathcal{F}(n(\bar{v})) \\ \wedge & \Box\,\forall\bar{y}.\ \circledcirc\, n(\bar{y}) \Leftrightarrow A(\bar{y}) \end{array} \right\}$$

$$\Box\,\forall\bar{x}.\ P(\bar{x}) \Rightarrow \mathcal{F}(\Diamond A(\bar{v})) \longrightarrow \left\{ \begin{array}{ll} & \Box\,\forall\bar{x}.\ P(\bar{x}) \Rightarrow \mathcal{F}(f(\bar{v})) \\ \wedge & \Box\,\forall\bar{y}.\ f(\bar{y}) \Leftrightarrow A(\bar{y}) \end{array} \right\}$$

This last transformation is used only when a '$\Diamond$' operator is not applied to a literal. Note that, the $\mathcal{S}$, $\mathcal{Z}$, $\mathcal{W}$, and $\mathcal{U}$ operators are effectively replaced by their fixpoint definitions [8].

Having removed all the unnecessary temporal operators, the last major transformation that needs to be carried out is to remove all existential quantifiers from the past-time component of each rule and all universal quantifiers from the future-time component of each rule. This is because, though the quantifiers now only appear at the outer level of the formula, each rule is required to be of the form

$$\Box\,\forall\bar{x}.\ (\forall\bar{y}.\ P(\bar{x}, \bar{y})) \Rightarrow (\exists\bar{z}.\ F(\bar{x}, \bar{z}))$$

Removing Existential Quantifiers from Past-time Components. To remove existential quantifiers from the past-time component of a rule, several renamings are applied. Given a rule of the form

$$\Box\,\forall\bar{x}.\ (\exists y.\ P(\bar{x}, y)) \Rightarrow F(\bar{x})$$

a new predicate is introduced to replace the past-time formula $P(\bar{x}, y)$. Thus, $P(\bar{x}, y)$ is replaced by $q(\bar{x}, y)$, and the following new rules are added, ensuring that $q(\bar{x}, y)$ is satisfied if, and only if, $P(\bar{x}, y)$ is.

$$\Box\,\forall\bar{v}.\ \forall w.\quad [P(\bar{v}, w) \Rightarrow q(\bar{v}, w)]$$
$$\Box\,\forall\bar{v}.\ \forall w.\quad [\neg P(\bar{v}, w) \Rightarrow \neg q(\bar{v}, w)]$$

Thus, the original rule becomes

$$\Box\,\forall\bar{x}.\ (\exists y.\ q(\bar{x}, y)) \Rightarrow F(\bar{x})$$

Now, another new predicate '$e(\bar{x})$' is introduced in order to replace '$\exists y.\ q(\bar{x}, y)$'. Thus, the original formula becomes $\Box\,\forall\bar{x}.\ e(\bar{x}) \Rightarrow F(\bar{x})$ and the following new rules, defining the value of the predicate e, are added.

$$\Box\,\forall\bar{v}.\quad [e(\bar{v}) \Rightarrow (\exists y.\ q(\bar{v}, y))]$$
$$\Box\,\forall\bar{v}.\quad [\neg e(\bar{v}) \Rightarrow (\forall y.\ \neg q(\bar{v}, y))]$$

Note that though a new universal quantifier is introduced, it appears at the outer level of the future-time component (such universal quantifiers are removed below). Each renaming operation has introduced a new predicate whose value is linked directly to the subformula it replaced.

Removing Universal Quantifiers from Future-time Components. Given a rule of the form

$$\Box\, \forall \bar{x}.\ P(\bar{x}) \ \Rightarrow\ (\forall y.\ F(\bar{x}, y))$$

the fact that the predicate *true* is satisfied for any arguments is used and the above rule is rewritten to

$$\Box\, \forall \bar{x}.\ \forall y.\ (P(\bar{x}) \wedge \mathit{true}(y)) \ \Rightarrow\ F(\bar{x}, y)\,.$$

This is a sound non-temporal transformation and ensures that the quantifiers are in the positions specified by SNF_f.

Rewriting Non-Temporal Formulae The rules can now be rewritten using classical transformation rules into the appropriate SNF_f form. This simply involves rewriting the formulae into the appropriate conjunctive or disjunctive forms.

4.3 Example

To show how the transformation proceeds, an example of a formula in rule form will be given and its transformation into SNF_f will be followed.

- The original formula is as follows and is in the 'past implies future' form.

$$\Box\, \forall x.\ [\, \bigcirc \exists z.\ c(z, x)] \ \Rightarrow\ [a(x)\, \mathcal{W}\, (\forall y.\ b(x, y))]$$

- First, the embedded quantifier is removed from the future-time component, generating the following rules.

$$\begin{array}{lllrcl}
1. & \Box\, \forall x. & [\, \bigcirc \exists z.\ c(z, x)] & \Rightarrow & a(x)\, \mathcal{W}\, q(x) \\
2. & \Box\, \forall w. & q(w) & \Rightarrow & \forall y.\ b(w, y) \\
3. & \Box\, \forall w. & \neg q(w) & \Rightarrow & \exists y.\ \neg b(w, y)
\end{array}$$

- Similarly, the embedded quantifier is removed from the past-time component, giving the following.

$$\begin{array}{lllcl}
1. & \Box\, \forall x. & \bigcirc e(x) & \Rightarrow & a(x)\, \mathcal{W}\, q(x) \\
4. & \Box\, \forall w. & e(w) & \Rightarrow & \exists z.\ c(z, w) \\
5. & \Box\, \forall w. & \neg e(w) & \Rightarrow & \forall z.\ \neg c(z, w)
\end{array}$$

- All the quantifiers occur at the outer level of each component, so now the '$\mathcal{W}$' operator is replaced by its definition, giving the following.

$$\begin{array}{lllcl}
1. & \Box\, \forall x. & \bigcirc e(x) & \Rightarrow & q(x) \vee (a(x) \wedge u(x)) \\
6. & \Box\, \forall y. & \bigcirc u(y) & \Rightarrow & q(y) \vee (a(y) \wedge u(y))
\end{array}$$

- Next, the universal quantifiers that appear in the future-time components of rules 2 and 5 are removed, giving the following replacements for these rules.

$$\begin{array}{lllll} 2. & \Box\,\forall w.\,\forall y. & q(w) \wedge \mathit{true}(y) & \Rightarrow & b(w,y) \\ 5. & \Box\,\forall w.\,\forall z. & \neg e(w) \wedge \mathit{true}(z) & \Rightarrow & \neg c(z,w) \end{array}$$

- Finally, rules 1 and 6 are split to ensure that the future-time component of each rule is a disjunction of literals. This splits rule 1 into new rules 1a and 1b, and splits rule 6 into the new rules 6a and 6b. Thus, the final set of SNF_f rules produced from the original formula is

$$\begin{array}{lllll} 1a. & \Box\,\forall x. & \bigcirc e(x) & \Rightarrow & q(x) \vee a(x) \\ 1b. & \Box\,\forall x. & \bigcirc e(x) & \Rightarrow & q(x) \vee u(x) \\ 2. & \Box\,\forall w.\,\forall y. & q(w) \wedge \mathit{true}(y) & \Rightarrow & b(w,y) \\ 3. & \Box\,\forall w. & \neg q(w) & \Rightarrow & \exists y.\,\neg b(w,y) \\ 4. & \Box\,\forall w. & e(w) & \Rightarrow & \exists z.\,c(z,w) \\ 5. & \Box\,\forall w.\,\forall z. & \neg e(w) \wedge \mathit{true}(z) & \Rightarrow & \neg c(z,w) \\ 6a. & \Box\,\forall y. & \bigcirc u(y) & \Rightarrow & q(y) \vee a(y) \\ 6b. & \Box\,\forall y. & \bigcirc u(y) & \Rightarrow & q(y) \vee u(y) \end{array}$$

4.4 Correctness of Transformations

Assuming that the transformation procedure is characterised by $\tau\colon \mathrm{WFF}_f \mapsto \mathrm{WFF}_f$, where for any formula A, $\tau(A)$ is in SNF_f, the following results can be given.

Lemma 1 *Given a formula A in WFF_f, if A is satisfiable, so is $\tau(A)$.*

Lemma 2 *Given a formula A in WFF_f, if $\tau(A)$ is satisfied in a model $\mathcal{M}$, then A is also satisfied in $\mathcal{M}$.*

These results follow immediately from Theorem 1 and the corresponding properties of the transformations not based upon renaming [4].

4.5 Complexity

Although a thorough investigation into the complexity of the transformation to SNF_f is part of our future work (see §6), coarse complexity bounds can be given by recognising that the renaming transformations used to remove both quantifiers and temporal operators introduce at most five new rules.

5 Applications

Two applications immediately follow from Lemmas 1 and 2. The first is a temporal resolution method that can be applied to formulae in SNF_f, which relies on Lemma 1. The second is the direct execution of temporal formulae in SNF_f, which relies on both Lemmas.

5.1 Temporal Resolution

Given a formula in SNF_f, i.e., $\Box\, \forall \bar{x}.\ (\forall \bar{y}.\ P(\bar{x}, \bar{y})) \ \Rightarrow\ (\exists \bar{z}.\ F(\bar{x}, \bar{z}))$ skolemization can be applied to give a universally quantified formula of the form

$$\Box\, \forall \bar{x}.\ P'(\bar{x}) \ \Rightarrow\ F'(\bar{x})$$

where P' and F' contain no quantifiers. To this type of formula, a variety of resolution methods can be applied, ranging from non-clausal resolution [1] to translation methods [6]. However, the particular motivation for developing this normal form was to provide the basis of a first-order extension of the resolution method described in [5].

This resolution process applies to sets of formulae in SNF and is split two components: the application of non-temporal (classical) resolution to $\Box$-rules; and the application of a temporal resolution rule. Temporal resolution is applied to a $\Diamond$-rule containing a future-time component such as $\Diamond q$. In this case, the rule can be resolved with a set of $\Box$-rules, 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 be satisfied while $\Diamond q$ is outstanding.

This temporal resolution rule encodes a form of induction over temporal sequences as hidden $\Box$-formulae must be found from the set of rules in order that they may be resolved with a complementary $\Diamond$-formula. The actual search for the formulae on which to apply the temporal resolution rule can be implemented using graph-theoretic techniques [5].

Further work on the first-order version of this method is under way, as are investigations into the first-order analogue of the graph-theoretic methods that are used to implement temporal resolution in the propositional case.

5.2 Execution of Temporal Formulae

Having produced a formula in SNF_f which is satisfiable, if and only if, the original formula is satisfiable, this formula can be executed, producing a model consistent with either formula. In particular, METATEM [2], a system for executing temporal formulae, accepts formulae in SNF_f. The formulae executed within METATEM are represented using *implicit* quantification. As the general form of a formula in SNF_f is

$$\Box\, \forall \bar{x}.\ (\forall \bar{y}.\ P(\bar{x}, \bar{y})) \ \Rightarrow\ (\exists \bar{z}.\ F(\bar{x}, \bar{z}))$$

then this can be represented in the form expected by METATEM by eliminating the quantifiers. The assumptions made by the METATEM system are that

- variables that appear only on the left of the '$\Rightarrow$' are universally quantified
- variables that appear only on the right of the '$\Rightarrow$' are existentially quantified
- variables that appear on both sides '$\Rightarrow$' are universally quantified at the outer level

For example,

$$\bigcirc\, test(x, y) \ \wedge\ run(x, z) \ \Rightarrow\ \Diamond terminate(a, w, y)$$

where w, x, y, and z are variables, and a is a ground term, represents the formula

$$\Box\, \forall y.\ [\forall x.\ \forall z.\ (\ \bigcirc\, test(x, y) \ \wedge\ run(x, z))] \ \Rightarrow\ [\exists w.\ \Diamond terminate(a, w, y)]$$

Thus, given a formula in the appropriate form, METATEM will attempt to generate a model for this formula. (For more details of this approach to executable temporal logics, see [2].)

6 Further Work

In this paper a normal form for a first-order temporal logic has been described. The procedure by which arbitrary formulae are transformed into this normal form has been outlined and justified. There are several obvious areas yet to be investigated. For example, more work is required on refining the complexity bounds for the transformation process. Related to this is the question of whether, by using more specific renaming techniques (such as using '$\Rightarrow$' rather than '$\Leftrightarrow$' in certain cases [7]), these bounds can be reduced.

The transformations described in this paper have been applied to a specific temporal logic. One question that remains is whether similar transformations can be applied to related logics, for example 'standard' modal logics, and logics that do not have constant domains.

Work on extending the resolution method described in [5] to this first-order framework is in progress, as is the development of an implementation of this procedure.

7 Acknowledgements

The author wishes to thank Philippe Noël, Rajeev Goré, and the anonymous referees for their helpful comments on an earlier version of this paper.

References

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

[2] 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).

[3] T. Boy de-la Tour and G. Chaminade. The Use of Renaming to Improve the Efficiency of Clausal Theorem Proving. In Ph. Jorrand and V. Sgurev, editors, *Artificial Intelligence IV: Methodology, Systems, Applications*, pages 3–12. Elsevier Science Publishers B.V. (North-Holland), 1990.

[4] 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.

[5] Michael 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.

[6] Andreas Nonnengart. Resolution for First-Order Linear Temporal Logics. (Draft), 1991.

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

[8] Pierre Wolper. Temporal Logic Can Be More Expressive. *Information and Control*, 56, 1983.