Mechanising First-Order Temporal Resolution

Boris Konev [a,1] Anatoli Degtyarev [b] Clare Dixon [a] Michael Fisher [a]
Ullrich Hustadt [a]

[a] *Department of Computer Science, University of Liverpool, Liverpool, U.K.*

[b] *Department of Computer Science, King's College London, Strand, London, U.K.*

Abstract

First-order temporal logic is a concise and powerful notation, with many potential applications in both Computer Science and Artificial Intelligence. While the full logic is highly complex, recent work on monodic first-order temporal logics has identified important enumerable and even decidable fragments. Although a complete and correct resolution-style calculus has already been suggested for this specific fragment, this calculus involves constructions too complex to be of a practical value. In this paper, we develop a machine-oriented clausal resolution method which features radically simplified proof search. We first define a normal form for monodic formulae and then introduce a novel resolution calculus that can be applied to formulae in this normal form. The calculus is based on classical first-order resolution and can, thus, be efficiently implemented. We prove correctness and completeness results for the calculus and illustrate it on a comprehensive example. An implementation of the method is briefly discussed.

Key words: clausal resolution, temporal logics, monodic fragment

1 Introduction

In its propositional form, linear, discrete *temporal logic* has been widely used in the formal specification and verification of reactive systems [1–3]. Although recognised as both a powerful and natural formalism, *first-order* temporal logic has generally been avoided due to completeness problems. In particular, there is no finite

Email addresses: `B.Konev@csc.liv.ac.uk` (Boris Konev), `Anatoli@dcs.kcl.ac.uk` (Anatoli Degtyarev), `C.Dixon@csc.liv.ac.uk` (Clare Dixon), `M.Fisher@csc.liv.ac.uk` (Michael Fisher), `U.Hustadt@csc.liv.ac.uk` (Ullrich Hustadt).

[1] On leave from Steklov Institute of Mathematics at St.Petersburg

axiom system for general first-order temporal logic. However, recent work by Hodkinson *et al.* [4] has shown that a specific fragment of first-order temporal logic, termed the *monodic* fragment, has the completeness (sometimes even decidability) property. This breakthrough has led to considerable research activity examining the monodic fragment, in terms of decidable classes, extensions, applications and mechanisation, and promises important advances for the future of formal methods for reactive systems.

In order to effectively utilise monodic temporal logics, we require tools mechanising their decision procedures. Concerning the mechanisation of monodic temporal logics, general tableau and resolution calculi have already been defined, in [5] and [6,7], respectively. However, neither of these is particularly practical: given a formula ϕ to be tested for satisfiability, the tableau method requires representation of *all* possible first-order models of first-order subformulae of ϕ, while the resolution method involves *all* possible combinations of temporal clauses in the clause normal form of ϕ. Thus, improved tools are required.

In this paper, we focus on an important subclass of temporal models, having a wide range of applications, for example in spatio-temporal logics [8,9] and temporal description logics [10], namely those models that have *expanding domains*. In such models, the domains over which first-order terms range can increase at each temporal step. The focus on this class of models allows us to produce a simplified clausal resolution calculus, termed the *fine-grained* resolution calculus, which is more amenable to efficient implementation. Thus, we here describe such an implementation, consider its properties and extend the consideration to constant-domain problems. This represents the first practically useful tool for handling monodic first-order temporal logics.

In what follows, we will define the expanding domain monodic fragment and the fine-grained resolution calculus, and provide completeness results for the fine-grained resolution calculus relative to the completeness of the general resolution calculus [11]. A number of examples will be given, showing how the fine-grained resolution calculus works in practice and describing the implementation being developed. We briefly describe how constant domain problems can be handled, through a translation of the formulae and, finally, consider conclusions and future work.

2 First-Order Temporal Logic

First-Order (discrete linear time) Temporal Logic, FOTL, is an extension of classical first-order logic with operators that deal with a discrete and linear model of time (isomorphic to $\mathbb{N}$, and the most commonly used model of time). The vocabulary of FOTL consists of:

- *Predicate symbols* $P_0, P_1, \ldots$ each of which is of some fixed arity (null-ary predicate symbols are called *propositions*);
- *Individual variables* $x_0, x_1, \ldots$;
- *Individual constants* $c_0, c_1, \ldots$;
- *Booleans* $\wedge$, $\neg$, $\vee$, $\Rightarrow$, $\equiv$, **true** ('true'), **false** ('false');
- *Quantifiers* $\forall$ and $\exists$;
- *Temporal operators* $\square$ ('always in the future'), $\Diamond$ ('sometime in the future'), $\bigcirc$ ('at the next moment'), U (until), and W (weak until).

There are no function symbols or equality in this FOTL language, but it does contain constants.

The set of FOTL-formulae is defined in a standard way [12,4]:

- Booleans **true** and **false** are FOTL-formulae;
- if P is an n-ary predicate symbol and t_i, $1 \leq i \leq n$, are variables or constants, then $P(t_1, \ldots, t_n)$ is an (atomic) FOTL-formula;
- if ϕ and ψ are FOTL-formulae, so are $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \Rightarrow \psi$, and $\phi \equiv \psi$;
- if ϕ is an FOTL-formula and x is a variable, then $\forall x \phi$ and $\exists x \phi$ are FOTL-formulae;
- if ϕ and ψ are FOTL-formulae, then so are $\square\phi$, $\Diamond\phi$, $\bigcirc\phi$, $\phi \mathsf{U} \psi$, and $\phi \mathsf{W} \psi$.

For a given formula, ϕ, $const(\phi)$ denotes the set of constants occurring in ϕ. We write $\phi(x)$ to indicate that $\phi(x)$ *has at most one free variable* x (if not explicitly stated otherwise). Formula having no free occurrences of variables is called *closed*.

Intuitively, FOTL formulae are interpreted in *first-order temporal structures* which are sequences $\mathfrak{M}$ of *worlds*, $\mathfrak{M} = \mathfrak{M}_0, \mathfrak{M}_1, \ldots$ with truth values in different worlds being connected via temporal operators.

More formally, for every moment of time $n \geq 0$, there is a corresponding *first-order* structure, $\mathfrak{M}_n = \langle D_n, I_n \rangle$, where every D_n is a non-empty set such that whenever $n < m$, $D_n \subseteq D_m$, and I_n is an interpretation of predicate and constant symbols over D_n. We require that the interpretation of constants is *rigid*. Thus, for every constant c and all moments of time $i, j \geq 0$, we have $I_i(c) = I_j(c)$.

A *(variable) assignment* $\mathfrak{a}$ is a function from the set of individual variables to $\cup_{n \in \mathbb{N}} D_n$. (This definition implies that variable assignments are also rigid.) We denote the set of all assignments by $\mathfrak{V}$. The set of variable assignments $\mathfrak{V}_n$ corresponding to $\mathfrak{M}_n$ is a subset of the set of all assignments, $\mathfrak{V}_n = \{\mathfrak{a} \in \mathfrak{V} \mid \mathfrak{a}(x) \in D_n$ for every variable $x\}$; clearly, $\mathfrak{V}_n \subseteq \mathfrak{V}_m$ if $n < m$.

The *truth* relation $\mathfrak{M}_n \models^{\mathfrak{a}} \phi$ in a structure $\mathfrak{M}$, is defined inductively on the construction of ϕ *only for those assignments* $\mathfrak{a}$ *that satisfy the condition* $\mathfrak{a} \in \mathfrak{V}_n$:

$\mathfrak{M}_n \models^{\mathfrak{a}}$ **true**, $\mathfrak{M}_n \not\models^{\mathfrak{a}}$ **false**

$\mathfrak{M}_n \models^{\mathfrak{a}} P(t_1,\ldots,t_m)$	iff $\langle I_n^{\mathfrak{a}}(t_1),\ldots I_n^{\mathfrak{a}}(t_m)\rangle \in I_n(P)$, where
	$I_n^{\mathfrak{a}}(t_i) = I_n(t_i)$, if t_i is a constant, and
	$I_n^{\mathfrak{a}}(t_i) = \mathfrak{a}(t_i)$, if t_i is a variable
$\mathfrak{M}_n \models^{\mathfrak{a}} \neg\phi$	iff $\mathfrak{M}_n \not\models^{\mathfrak{a}} \phi$
$\mathfrak{M}_n \models^{\mathfrak{a}} \phi \wedge \psi$	iff $\mathfrak{M}_n \models^{\mathfrak{a}} \phi$ and $\mathfrak{M}_n \models^{\mathfrak{a}} \psi$
$\mathfrak{M}_n \models^{\mathfrak{a}} \phi \vee \psi$	iff $\mathfrak{M}_n \models^{\mathfrak{a}} \phi$ or $\mathfrak{M}_n \models^{\mathfrak{a}} \psi$
$\mathfrak{M}_n \models^{\mathfrak{a}} \phi \Rightarrow \psi$	iff $\mathfrak{M}_n \models^{\mathfrak{a}} (\neg\phi \vee \psi)$
$\mathfrak{M}_n \models^{\mathfrak{a}} \phi \equiv \psi$	iff $\mathfrak{M}_n \models^{\mathfrak{a}} ((\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi))$
$\mathfrak{M}_n \models^{\mathfrak{a}} \bigcirc\phi$	iff $\mathfrak{M}_{n+1} \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} \Diamond\phi$	iff there exists $m \geq n$ such that $\mathfrak{M}_m \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} \Box\phi$	iff for all $m \geq n$, $\mathfrak{M}_m \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} (\phi \mathsf{U} \psi)$	iff there exists $m \geq n$, such that $\mathfrak{M}_m \models^{\mathfrak{a}} \psi$ and
	for all $i \in \mathbb{N}, n \leq i < m$ implies $\mathfrak{M}_i \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} (\phi \mathsf{W} \psi)$	iff $\mathfrak{M}_n \models^{\mathfrak{a}} (\phi \mathsf{U} \psi)$ or $\mathfrak{M}_n \models^{\mathfrak{a}} \Box\phi$.

$\mathfrak{M}$ is a *model* for a formula ϕ (or ϕ is *true* in $\mathfrak{M}$) if, and only if, there exists an assignment $\mathfrak{a}$ in D_0 such that $\mathfrak{M}_0 \models^{\mathfrak{a}} \phi$. A formula is *satisfiable* if, and only if, it has a model. A formula is *valid* if, and only if, it is true in any temporal structure $\mathfrak{M}$ under any assignment $\mathfrak{a}$ in D_0.

The models introduced above are known as *models with expanding domains* since $D_n \subseteq D_{n+1}$. Another important class of models consists of *models with constant domains* in which the class of first-order temporal structures, where FOTL formulae are interpreted, is restricted to structures $\mathfrak{M} = \langle D_n, I_n\rangle$, $n \in \mathbb{N}$, such that $D_i = D_j$ for all $i, j \in \mathbb{N}$. The notions of truth and validity are defined similarly to the expanding domain case.

Example 1 The formula

$$\forall x P(x) \wedge \Box(\forall x P(x) \Rightarrow \bigcirc\forall x P(x)) \wedge \Diamond\exists y \neg P(y)$$

is unsatisfiable over both expanding and constant domains; the formula

$$\forall x P(x) \wedge \Box(\forall x(P(x) \Rightarrow \bigcirc P(x))) \wedge \Diamond\exists y \neg P(y)$$

is unsatisfiable over constant domains but has a model with an expanding domain.

It is known [13] that satisfiability over expanding domains can be reduced to satisfiability over constant domains with a polynomial increase in size of formulae.

First-order temporal logic is complex: providing complete methods for solving the satisfiability or validity problem for this logic in its full generality is impossible. Furthermore, it is known that even "small" fragments of FOTL, such as the *two-variable monadic* fragment (where all predicates are unary), are not recursively enumerable [14,4]. However, the set of valid *monodic* formulae (see Definition 1 below) is known to be finitely axiomatisable [15].

Definition 1 An FOTL-formula ϕ is called *monodic* if, and only if, any subformulae of the form $\mathcal{T}\psi$, where $\mathcal{T}$ is one of $\bigcirc$, $\square$, $\Diamond$ (or $\psi_1 \mathcal{T} \psi_2$, where $\mathcal{T}$ is one of U, W), contains at most one free variable.

Example 2 The formulae

$$\forall x \square \exists y P(x,y) \quad \text{and} \quad \forall x \square P(x,c)$$

are monodic, whereas the formula

$$\forall x,y(P(x,y) \Rightarrow \square P(x,y))$$

is non-monodic.

We note that the addition of either equality or function symbols to the monodic fragment generally leads to the loss of recursive enumerability [15]. Moreover, although the two variable monodic fragment *without* equality is decidable [4], it was proved in [16] that the *two variable monadic monodic fragment with equality* is not even recursively enumerable. However, in [17] it was shown that the *guarded monodic fragment with equality* is decidable.

3 Divided Separated Normal Form (DSNF)

Resolution calculi for first-order logic require that first-order formulae are transformed to a *clause normal form* [18], before the inference rules of the calculus can be applied. Similarly, the completeness calculus which will be defined in Section 4 as well as the fine-grained resolution which will be defined in Section 5 require first-order temporal formulae to be transformed into a particular normal form. We define this normal form in the following. :

Definition 2 A *temporal step clause* is a formula either of the form $p \Rightarrow \bigcirc l$, where p is a proposition and l is a propositional literal, or $(P(x) \Rightarrow \bigcirc M(x))$, where $P(x)$ is a unary predicate and $M(x)$ is a unary literal.

These are the key elements of the the normal form, providing a description of how information is transferred from one temporal step to the next. We call a clause of the the first type an (original) *ground* step clause, and of the second type an (original) *non-ground* step clause. (Note that use of the term 'original' here is used to distinguish these clauses from other that are generated later in the preprocessing stage.)

Definition 3 A *monodic temporal problem in Divided Separated Normal Form (DSNF)* is a quadruple $\langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$, where

(1) the universal part, $\mathcal{U}$, is given by a set of arbitrary closed first-order formulae;
(2) the initial part, $\mathcal{I}$, is, again, given by a set of arbitrary closed first-order formulae;
(3) the step part, $\mathcal{S}$, is given by a set of original (ground and non-ground) temporal step clauses, the left-hand sides of step clauses are pairwise distinct; and
(4) the eventuality part, $\mathcal{E}$, is given by a set of clauses of the form $\Diamond L(x)$ (a *non-ground* eventuality clause) and $\Diamond l$ (a *ground eventuality* clause), where l is a propositional literal and $L(x)$ is a unary non-ground literal.

The sets $\mathcal{U}$, $\mathcal{I}$, $\mathcal{S}$, and $\mathcal{S}$ are finite.

Note that, in a monodic temporal problem, we do not allow two different temporal step clauses with the same left-hand sides. (A problem containing two different temporal step clauses with the same left-hand sides can be easily transformed by renaming into one without.)

In what follows, we will not distinguish between a finite set of formulae $\mathcal{X}$ and the conjunction $\bigwedge \mathcal{X}$ of formulae within the set. With each monodic temporal problem, we associate the formula

$$\mathcal{I} \wedge \Box \mathcal{U} \wedge \Box \forall x \mathcal{S} \wedge \Box \forall x \mathcal{E}.$$

Now, when we talk about particular properties of a temporal problem (e.g., satisfiability, validity, logical consequences etc) we refer to properties of the associated formula.

Arbitrary monodic first-order temporal formulae can be transformed into DSNF. The reduction is based on using a renaming technique to substitute non-atomic subformulae and replacing temporal operators by their fixed point definitions as described e.g. in [19]. The translation can be described as a number of steps.

(1) Translate a given monodic formula to negation normal form. (To assist understanding of the translation, we list here some equivalent FOTL formulae.)

$$\forall x(\neg\bigcirc\phi(x) \equiv \bigcirc\neg\phi(x));\ \forall x(\neg\Box\phi(x) \equiv \Diamond\neg\phi(x));\ \forall x(\neg\Diamond\phi(x) \equiv \Box\neg\phi(x);$$
$$\forall x(\neg(\phi(x)\,\mathsf{U}\,\psi(x)) \equiv \neg\psi(x)\,\mathsf{W}\,(\neg\phi(x)\wedge\neg\psi(x))));$$
$$\forall x(\neg(\phi(x)\,\mathsf{W}\,\psi(x)) \equiv \neg\psi(x)\,\mathsf{U}\,(\neg\phi(x)\wedge\neg\psi(x))).$$

If the transformations above are applied in a straightforward way, the size of the result may grow exponentially; we may have to use *renaming* [20,18] in order to keep it linear.

(2) Recursively rename each innermost temporal subformulae, $\bigcirc\phi(x)$, $\Diamond\phi(x)$, $\Box\phi(x)$, $\phi(x)\,\mathsf{U}\,\psi(x)$, $\phi(x)\,\mathsf{W}\,\psi(x)$ by a new unary predicate $P(x)$ (using a new name for each subformula). Renaming introduces formulae defining $P(x)$ (since we are only interested in satisfiability, we use implications instead of equivalences renaming positive occurrences of subformulae, see also [20,18]) of the following form:

$$(a)\ \Box\forall x(P(x) \Rightarrow \bigcirc\phi(x));\qquad (c)\ \Box\forall x(P(x) \Rightarrow \Box\phi(x));$$
$$(b)\ \Box\forall x(P(x) \Rightarrow \phi(x)\,\mathsf{W}\,\psi(x));\ (d)\ \Box\forall x(P(x) \Rightarrow \phi(x)\,\mathsf{U}\,\psi(x));$$
$$(e)\ \Box\forall x(P(x) \Rightarrow \Diamond\phi(x)).$$

Formulae of the form (a) are already in DSNF after renaming complex first-order formulae from the right-hand side; first-order clauses from this renaming are put in the universal part. This kind of renaming is assumed implicitly in the following.

(3) Formulae of the form (b), (c), and (d) require extra reduction by removing the temporal operators using their fixed point definitions (see [19]):
$\Box\forall x(P(x) \Rightarrow \Box\phi(x))$ is satisfiability equivalent to

$$\Box\forall x(P(x) \Rightarrow R(x)) \wedge \Box\forall x(R(x) \Rightarrow \bigcirc R(x)) \wedge \Box\forall x(R(x) \Rightarrow \phi(x)),$$

$\Box\forall x(P(x) \Rightarrow (\phi(x)\,\mathsf{U}\,\psi(x)))$ is equivalent (w.r.t. satisfiability) to

$$\begin{aligned}&\Box\forall x(P(x) \Rightarrow \Diamond\psi(x)) \wedge \Box\forall x(P(x) \Rightarrow \phi(x)\vee\psi(x))\\ &\wedge\Box\forall x(P(x) \Rightarrow S(x)\vee\psi(x)) \wedge \Box\forall x(S(x) \Rightarrow \bigcirc(\phi(x)\vee\psi(x)))\\ &\wedge\Box\forall x(S(x) \Rightarrow \bigcirc(S(x)\vee\psi(x))),\end{aligned}$$

and $\Box\forall x(P(x) \Rightarrow (\phi(x)\,\mathsf{W}\,\psi(x)))$ is equivalent (w.r.t. satisfiability) to

$$\begin{aligned}&\Box\forall x(P(x) \Rightarrow \phi(x)\vee\psi(x)) \wedge \Box\forall x(P(x) \Rightarrow S(x)\vee\psi(x))\\ &\wedge\Box\forall x(S(x) \Rightarrow \bigcirc(\phi(x)\vee\psi(x))) \wedge \Box\forall x(S(x) \Rightarrow \bigcirc(S(x)\vee\psi(x))),\end{aligned}$$

where $R(x)$ and $S(x)$ are new unary predicates.

(4) Finally, formulae of the form (e) are translated into DSNF as follows: $\Box\forall x(P(x) \Rightarrow \Diamond L(x))$ is equivalent (w.r.t. satisfiability, [21]) to

$$\Box\forall x(((P(x) \land \neg L(x)) \Rightarrow \mathit{waitforL}(x))) \tag{1}$$
$$\Box\forall x(\mathit{waitforL}(x) \Rightarrow \bigcirc(\mathit{waitforL}(x) \lor L(x))) \tag{2}$$
$$\Box\forall x(\Diamond\neg \mathit{waitforL}(x)) \tag{3}$$

where $\mathit{waitforL}(x)$ is a new unary predicate.

Theorem 1 (see [21], Theorem 1) The transformation described above reduces any monodic first-order temporal formula, in a satisfiability equivalence preserving way, to DSNF with at most a linear increase in size of the problem.

Example 3 Let us consider the temporal formula $\exists x\,\Box\Diamond\forall y\forall z\exists u\,\Phi(x,y,z,u)$ where $\Phi(x,y,z,u)$ does not contain temporal operators and reduce it to DSNF. First, we rename the innermost temporal subformula by a new predicate, P_1,

$$\exists x\,\Box P_1(x) \land \Box\forall x[P_1(x) \Rightarrow \Diamond\forall y\forall z\exists u\,\Phi(x,y,z,u)].$$

Now, we rename the first '$\Box$'-formula (introducing P_3) and the subformula under the '$\Diamond$' operator (introducing P_2),

$$\begin{aligned}\exists x P_3(x) &\land \Box\forall x[P_1(x) \Rightarrow \Diamond P_2(x)]\\ &\land \Box\forall x[P_2(x) \Rightarrow \forall y\forall z\exists u\,\Phi(x,y,z,u)]\\ &\land \Box\forall x[P_3(x) \Rightarrow \Box P_1(x)],\end{aligned}$$

"unwind" the '$\Box$' operator (introducing P_4)

$$\begin{aligned}\exists x P_3(x) &\land \Box\forall x[P_1(x) \Rightarrow \Diamond P_2(x)]\\ &\land \Box\forall x[P_2(x) \Rightarrow \forall y\forall z\exists u\,\Phi(x,y,z,u)]\\ &\land \Box\forall x[P_3(x) \Rightarrow P_4(x)]\\ &\land \Box\forall x[P_4(x) \Rightarrow \bigcirc P_4(x)]\\ &\land \Box\forall x[P_4(x) \Rightarrow P_1(x)],\end{aligned}$$

and, finally, reduce the conditional eventuality to an unconditional one.

$$\begin{aligned}\exists x P_3(x) &\land \Box\forall x[P_2(x) \Rightarrow \forall y\forall z\exists u\,\Phi(x,y,z,u)]\\ &\land \Box\forall x[P_3(x) \Rightarrow P_4(x)]\\ &\land \Box\forall x[P_4(x) \Rightarrow \bigcirc P_4(x)]\\ &\land \Box\forall x[P_4(x) \Rightarrow P_1(x)]\\ &\land \Box\forall x[(P_1(x) \land \neg P_2(x)) \Rightarrow \mathit{waitforP_2}(x)]\\ &\land \Box\forall x[\mathit{waitforP_2}(x) \Rightarrow \bigcirc(\mathit{waitforP_2}(x) \lor P_2(x))]\\ &\land \Box\forall x\Diamond\neg\mathit{waitforP_2}(x).\end{aligned}$$

The parts of this formula form the following monodic temporal problem (we also rename the complex $P_2(x) \vee waitforP_2(x)$ expression by $P_5(x)$):

$$\mathcal{U} = \left\{ \begin{array}{l} \forall x(P_2(x) \Rightarrow \forall y \forall z \exists u \Phi(x,y,z,u)), \\ \forall x(P_3(x) \Rightarrow P_4(x)), \\ \forall x(P_4(x) \Rightarrow P_1(x)), \\ \forall x((P_1(x) \wedge \neg P_2(x)) \Rightarrow waitforP_2(x)), \\ \forall x(P_5(x) \Rightarrow P_2(x) \vee waitforP_2(x)) \end{array} \right\}, \quad \mathcal{I} = \left\{ \exists x P_3(x) \right\},$$

$$\mathcal{S} = \left\{ \begin{array}{l} P_4(x) \Rightarrow \bigcirc P_4(x), \\ waitforP_2(x) \Rightarrow \bigcirc P_5(x) \end{array} \right\}, \qquad \mathcal{E} = \left\{ \forall x \Diamond \neg waitforP_2(x) \right\}.$$

4 Completeness Calculus

A resolution-like method for the monodic fragment over constant domains has been introduced in [6]. Although satisfiability over expanding domains can be reduced to satisfiability over constant domains [13], it has been proved in [11] that a simple modification of the method can be directly applied to the expanding domain case.

We sketch the monodic temporal resolution system here to make the paper self-contained. We use this 'completeness calculus' to show relative completeness of the calculus presented in the next section. More details on the completeness calculus, as well as proofs of the properties stated below, can be found in [21].

Let P be a monodic temporal problem, and let

$$P_{i_1}(x) \Rightarrow \bigcirc M_{i_1}(x), \ldots, P_{i_k}(x) \Rightarrow \bigcirc M_{i_k}(x) \tag{4}$$

be a subset of the set of its step clauses. Then formulae of the form

$$P_{i_j}(c) \Rightarrow \bigcirc M_{i_j}(c), \tag{5}$$

$$\exists x \bigwedge_{j=1}^{k} P_{i_j}(x) \Rightarrow \bigcirc \exists x \bigwedge_{j=1}^{k} M_{i_j}(x), \tag{6}$$

$$\forall x \bigvee_{j=1}^{k} P_{i_j}(x) \Rightarrow \bigcirc \forall x \bigvee_{j=1}^{k} M_{i_j}(x) \tag{7}$$

are called *derived* step clauses, where $c \in const(\mathsf{P})$ and $j = 1 \ldots k$. Formulae of the form (5) and (6) are called *e-derived* step clauses. Note that formulae of the form (5) and (6) are logical consequences of (4) in the *expanding domain* case; while formulae of the form (5), (6), *and* (7) are logical consequences of (4) in the *constant domain* case. As Example 1 shows, (7) is not a logical consequence of (4) in the expanding domain case.

Let $\{\Phi_1 \Rightarrow \bigcirc\Psi_1, \ldots, \Phi_n \Rightarrow \bigcirc\Psi_n\}$ be a set of derived (e-derived) step clauses or original *ground* step clauses. Then

$$\bigwedge_{i=1}^{n} \Phi_i \Rightarrow \bigcirc \bigwedge_{i=1}^{n} \Psi_i$$

is called a *merged derived step clause* (and *merged e-derived step clause*, respectively).

Let $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ be a merged derived (e-derived) step clause, let $P_1(x) \Rightarrow \bigcirc M_1(x)$, $\ldots$, $P_k(x) \Rightarrow \bigcirc M_k(x)$ be a subset of the original step clauses, and let $\mathcal{A}(x) \rightleftharpoons \mathcal{A} \wedge \bigwedge_{i=1}^{k} P_i(x)$, $\mathcal{B}(x) \rightleftharpoons \mathcal{B} \wedge \bigwedge_{i=1}^{k} M_i(x)$. Then

$$\forall x(\mathcal{A}(x) \Rightarrow \bigcirc\mathcal{B}(x))$$

is called a *full merged step clause* (*full e-merged step clause*, respectively).

Let P be a monodic temporal problem,

$$\mathsf{P}^c = \mathsf{P} \cup \{\Diamond L(c) \mid \Diamond L(x) \in \mathcal{E}, c \in const(\mathsf{P})\}$$

is the *constant flooded form* of P. Evidently, P^c is satisfiability equivalent to P.

In present now two calculi, $\mathfrak{I}_c$ and $\mathfrak{I}_e$, aimed at the constant and expanding domain cases, respectively [21]. The inference rules of these calculi coincide; the only difference is in the merging operation. The calculus $\mathfrak{I}_c$ utilises merged derived and full merged step clauses; whereas $\mathfrak{I}_e$ utilises merged e-derived and full e-merged step clauses.

4.1 Inference Rules.

In what follows, $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ and $\mathcal{A}_i \Rightarrow \bigcirc\mathcal{B}_i$ denote merged derived (e-derived) step clauses, $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc(\mathcal{B}(x)))$ and $\forall x(\mathcal{A}_i(x) \Rightarrow \bigcirc(\mathcal{B}_i(x)))$ denote full merged (full e-merged) step clauses, and $\mathcal{U}$ denotes the (current) universal part of the problem.

- *Step resolution rule w.r.t.* $\mathcal{U}$:

$$\frac{\mathcal{A} \Rightarrow \bigcirc\mathcal{B}}{\neg\mathcal{A}}\ (\bigcirc^{\mathcal{U}}_{res}), \text{ where } \mathcal{U} \cup \{\mathcal{B}\} \models \textbf{false}.$$

- *Initial termination rule w.r.t. $\mathcal{U}$:*

$$\frac{}{\textbf{false}}\ (\bot^{\mathcal{U}}_{res})\,,\ \text{if}\ \mathcal{U}\cup I\models \textbf{false}.$$

- *Eventuality resolution rule w.r.t. $\mathcal{U}$:*

$$\frac{\forall x(\mathcal{A}_1(x)\Rightarrow \bigcirc(\mathcal{B}_1(x)))\quad \dots\quad \forall x(\mathcal{A}_n(x)\Rightarrow \bigcirc(\mathcal{B}_n(x)))\qquad \Diamond L(x)}{\forall x \bigwedge_{i=1}^{n}\neg\mathcal{A}_i(x)}\ (\Diamond^{\mathcal{U}}_{res})\,,$$

  where $\forall x(\mathcal{A}_i(x)\Rightarrow \bigcirc\mathcal{B}_i(x))$ are full merged (full e-merged) step clauses such that for all $i\in\{1,\dots,n\}$, the *loop* side conditions $\forall x(\mathcal{U}\wedge\mathcal{B}_i(x)\Rightarrow\neg L(x))$ and $\forall x(\mathcal{U}\wedge\mathcal{B}_i(x)\Rightarrow\bigvee_{j=1}^{n}(\mathcal{A}_j(x)))$ are both valid[2].

  The set of full merged (full e-merged) step clauses, satisfying the loop side conditions, is called a *loop in* $\Diamond L(x)$ and the formula $\bigvee_{j=1}^{n}\mathcal{A}_j(x)$ is called a *loop formula*.
- *Ground eventuality resolution rule w.r.t. $\mathcal{U}$:*

$$\frac{\mathcal{A}_1\Rightarrow\bigcirc\mathcal{B}_1\quad \dots\quad \mathcal{A}_n\Rightarrow\bigcirc\mathcal{B}_n\qquad \Diamond l}{\bigwedge_{i=1}^{n}\neg\mathcal{A}_i}\ (\Diamond^{\mathcal{U}}_{res})\,,$$

  where $\mathcal{A}_i\Rightarrow\bigcirc\mathcal{B}_i$ are merged derived (e-derived) step clauses such that the *loop* side conditions $\mathcal{U}\wedge\mathcal{B}_i\models\neg l$ and $\mathcal{U}\wedge\mathcal{B}_i\models\bigvee_{j=1}^{n}\mathcal{A}_j$ for all $i\in\{1,\dots,n\}$ are both valid. *Ground loop* and *ground loop formula* are defined similarly to the case above.

For a temporal problem P, by $\mathrm{TRes}(\mathsf{P})$ we denote the set of all formulae which can be obtained from P applying the inference rules above.

Similarly to classical first-order resolution, temporal resolution is a refutationally complete saturation-based theorem proving method, i.e., a contradiction can be deduced from any unsatisfiable problem, and the search for a contradiction proceeds by saturation the universal part of a given problem.

Definition 4 A *derivation* is a sequence of universal parts, $\mathcal{U}=\mathcal{U}_0\subseteq\mathcal{U}_1\subseteq\mathcal{U}_2\subseteq\dots$, extended little by little by the conclusions of the inference rules. A derivation *terminates* if, and only if, either **false** is derived, in which case we say that the derivation is *successfully terminating*, or if no new clauses can be derived by further inference steps. The I, $\mathcal{S}$ and $\mathcal{E}$ parts of the temporal problem are not changed during a derivation.

A derivation $\mathcal{U}=\mathcal{U}_0\subseteq\mathcal{U}_1\subseteq\mathcal{U}_2\subseteq\dots$, is *fair* if it either successfully terminates or, for any $i\geq 0$ and formula $\phi\in\mathrm{TRes}(\langle\mathcal{U}_i, I,\mathcal{S},\mathcal{E}\rangle)$, there exists $j\geq i$ such that $\phi\in\mathcal{U}_j$.

[2] In the case $\mathcal{U}\models\forall x\neg L(x)$, the *degenerate clause*, $\textbf{true}\Rightarrow\bigcirc\textbf{true}$, can be considered as a premise of this rule; the conclusion of the rule is then $\neg\textbf{true}$ and the derivation successfully terminates.

Note 1 We intentionally do not include into consideration here the classical concept of redundancy (see [22]) and deletion rules over sets of first-order formulae $\mathcal{U}_i$ because we only concentrate on temporal reasoning. Therefore, all formulae that appear in a derivation are persistent. We come to the issue of redundancy and deletion rules in Section 5 where we present a machine-oriented calculus.

Theorem 2 (see [21], theorems 2 and 3) The rules of $\mathfrak{I}_c$ preserve satisfiability over constant domains. If a monodic temporal problem P is unsatisfiable over constant domains, then any *fair* derivation in $\mathfrak{I}_c$ from P^c successfully terminates.

Theorem 3 (see [21], Theorem 10) The rules of $\mathfrak{I}_e$ preserve satisfiability over expanding domains. If a monodic temporal problem P is unsatisfiable over expanding domains, then any *fair* derivation in $\mathfrak{I}_e$ from P^c successfully terminates.

Example 4 The need for constant flooding can be demonstrated by the following example. None of the rules of temporal resolution can be applied directly to the (unsatisfiable) temporal problem given by

$$\begin{aligned} I &= \{P(c)\}, & \mathcal{S} &= \{q \Rightarrow \bigcirc q\}, \\ \mathcal{U} &= \{q \equiv P(c)\}, & \mathcal{E} &= \{\Diamond\neg P(x)\}. \end{aligned}$$

After constant flooding, however, when we add to the problem an eventuality clause

$$\Diamond\neg P(c)$$

(or after renaming, an eventuality $\Diamond l$ and a universal clause $l \Rightarrow \neg P(c)$), the step clause $q \Rightarrow \bigcirc q$ will be a loop in $\Diamond l$, and the eventuality resolution rule would derive a contradiction [3].

5 Fine-Grained Resolution for the Expanding Domain Case

The main drawback of the calculi introduced in the previous section is that the notion of a merged step clause is quite involved and the search for (full) merged (e-)derived step clauses that may serve as premises in an inference is computationally hard. Finding *sets* of such full merged step clauses needed for the temporal resolution rule is even more difficult.

From now on we focus on the expanding domain case. This is simpler firstly because merged e-derived step clauses are simpler (formulae of the form (7) do not

[3] Note that the non-ground eventuality $\Diamond\neg P(x)$ is not used. It was shown in [21] that if all step clauses are ground, for constant flooded problems we can neglect non-ground eventualities.

contribute to them) and, secondly, because the conclusion of every inference rule of $\mathfrak{I}_e$ is a first-order clause.

We now introduce a calculus where the inference rules of $\mathfrak{I}_e$ are refined into smaller steps, more suitable for effective implementation. First, we concentrate on the step resolution inference rule; then we show how to effectively find premises for the eventuality resolution rule by means of step resolution.

The calculus is inspired by the following consideration: Suppose that $\mathfrak{I}_e$ applies the step resolution rule to a merged e-derived step clause $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$. The rule can be applied if $\mathcal{B} \cup \mathcal{U} \models$ **false** and this fact can be established by a first-order resolution procedure (that would skolemise the universal part). Then the conclusion of the rule, $\neg\mathcal{A}$, is added to $\mathcal{U}$ resulting in a new universal part $\mathcal{U}'$. Suppose that the step resolution rule is applied to another merged e-derived step clause, $\mathcal{A}' \Rightarrow \bigcirc\mathcal{B}'$. The side condition, $\mathcal{B}' \cup \mathcal{U}' \models$ **false**, again can be checked by a first-order resolution procedure.

What is essential, we are not only going to check side conditions for the rules of the $\mathfrak{I}_e$ by means of first-order resolution but also *search for clauses to merge* at the same time.

We note also that fine-grained resolution might generate additional step clauses of the form

$$C \Rightarrow \bigcirc D. \tag{8}$$

Here, C is a *conjunction* of propositions, unary predicates of the form $P(x)$, and ground formulae of the form $P(c)$, where P is a unary predicate symbol and c is a constant occurring in the *problem given originally*; D is a *disjunction* of arbitrary literals.

Definition 5 Let P be a constant flooded temporal problem; the set of clauses $\mathbf{S}(\mathsf{P})$, called *the result of preprocessing*, consists of step clauses from P and

(1) For every original non-ground step clause

$$P(x) \Rightarrow \bigcirc M(x)$$

and every constant $c \in const(\mathsf{P})$, the clause

$$P(c) \Rightarrow \bigcirc M(c) \tag{9}$$

is in $\mathbf{S}(\mathsf{P})$.

(2) Clauses obtained by clausification of the universal and initial parts, as if there is no connection with temporal logic at all, are also in $\mathbf{S}(\mathsf{P})$. The resulting clauses are called *universal clauses* and *initial clauses* respectively. In the

beginning, universal and initial clauses do not have common Skolem constants and functions. Initial and universal clauses are kept separate.

In sections 5.1 and 5.2, we assume that each given problem is preprocessed.

5.1 Fine-grained step resolution

Fine-grained step resolution consists of a set of *deduction* and *deletion* rules. We implicitly assume that different premises and conclusions of the deduction rules have no variables in common; variables may be renamed if necessary.

5.1.1 Deduction rules

(1) *Arbitrary (first-order) resolution between universal clauses.* The result is a universal clause.

(2) *Arbitrary (first-order) resolution between initial and universal clauses (or just between initial clauses).* The result is again an initial clause.

(3) *Fine-grained (restricted) step resolution.*

$$\frac{C_1 \Rightarrow \bigcirc(D_1 \vee L) \quad C_2 \Rightarrow \bigcirc(D_2 \vee \neg M)}{(C_1 \wedge C_2)\sigma \Rightarrow \bigcirc(D_1 \vee D_2)\sigma},$$

where $C_1 \Rightarrow \bigcirc(D_1 \vee L)$ and $C_2 \Rightarrow \bigcirc(D_2 \vee \neg M)$ are step clauses and σ is an most general unifier of the literals L and M *such that* σ *does not map variables from* C_1 *or* C_2 *into a constant or a functional term*[4] .

$$\frac{C_1 \Rightarrow \bigcirc(D_1 \vee L) \quad D_2 \vee \neg N}{C_1\sigma \Rightarrow \bigcirc(D_1 \vee D_2)\sigma},$$

where $C_1 \Rightarrow \bigcirc(D_1 \vee L)$ is an step clause, $D_2 \vee \neg N$ is a universal clause, and σ is an most general unifier of the literals L and N *such that* σ *does not map variables from* C_1 *into a constant or a functional term.*

(4) *Right factor.*

$$\frac{C \Rightarrow \bigcirc(D \vee L \vee M)}{C\sigma \Rightarrow \bigcirc(D \vee L)\sigma},$$

where σ is an most general unifier of the literals L and M *such that* σ *does not map variables from* C *into a constant or a functional term.*

[4] This restriction justifies skolemisation: Skolem constants and functions do not 'sneak' into the left-hand side of step clauses, and, hence, Skolem constants from different moments of time do not mix.

(5) *Left factor.*

$$\frac{(C \wedge L \wedge M) \Rightarrow \bigcirc D}{(C \wedge L)\sigma \Rightarrow \bigcirc D\sigma},$$

where σ is an most general unifier of the literals L and M *such that* σ *does not map variables from C into a constant or a functional term.*

(6) *Clause conversion.*
A step clause of the form $C \Rightarrow \bigcirc\mathbf{false}$ is rewritten into the *universal clause* $\neg C$.

5.1.2 Deletion rules

(1) *First-order deletion.* (First-order) subsumption and tautology deletion in universal clauses; subsumption and tautology deletion in initial clauses; subsumption of initial clauses by universal clauses (but not vice versa).

(2) *Temporal deletion.*
A universal clause D_2 *subsumes* a step clause $C_1 \Rightarrow \bigcirc D_1$ if D_2 subsumes D_1 or D_2 subsumes[5] $\neg C_1$.
A step clause $C_1 \Rightarrow \bigcirc D_1$ *subsumes* a step clause $C_2 \Rightarrow \bigcirc D_2$ if there exists a substitution σ such that $D_1\sigma \subseteq D_2$ and $\neg C_1\sigma \subseteq \neg C_2$.
A step clause $C \Rightarrow \bigcirc D$ is a *tautology* if D is a tautology. (Note that, since we do not have negative occurrences to the left-hand side of step clauses, C cannot be false). Tautologies are deleted.

We adopt the terminology from [22]. A (linear) *proof* by fine-grained resolution of a clause C from a set of clauses $\mathbf{S}$ is a sequence of clauses $C_1, \dots, C_m$ such that $C = C_m$ and each clause C_i is either an element of $\mathbf{S}$ or else the conclusion by a deduction rule from $C_1, \dots, C_{i-1}$. A proof of **false** is called a *refutation*. A (theorem proving) *derivation* by fine-grained resolution is a sequence of sets of clauses $\mathbf{S_0} \rhd \mathbf{S_1} \rhd \dots$ such that every $\mathbf{S_{i+1}}$ differs from $\mathbf{S_i}$ by either adding the conclusion of a deduction rule or else deleting a clause by a deletion rule. We say that a clause C is *derived by fine-grained resolution from* $\mathbf{S_0}$ if $C \in \mathbf{S_i}$ for some i.

Note 2 Fine-grained step resolution without the restriction on substitutions would, certainly, lead to unsoundness: The monodic problem given by

$$\mathcal{U} = \{u1 : \exists x \neg Q(x), u2 : \forall x (P(x) \vee Q(x))\}, \ \mathcal{I} = \emptyset,$$
$$\mathcal{S} = \{s1 : P(x) \Rightarrow \bigcirc Q(x)\}, \qquad \mathcal{E} = \emptyset,$$

which is satisfiable, would wrongly be declared unsatisfiable without this restriction (After skolemisation, $\mathcal{U}^s = \{us1 : \neg Q(c),\ us2 : P(x) \vee Q(x)\}$, then unrestricted

[5] Here, and further, $\neg(L_1(x) \wedge \dots \wedge L_k(x))$ abbreviates $(\neg L_1(x) \vee \dots \vee \neg L_k(x))$.

resolution would derive $us3 : \neg P(c)$ from $us1$ and $s1$, and then the contradiction from $us1$, $us2$, and $us3$.)

Example 5 It might seem that the restriction on most general unifiers is too strong and may destroy completeness of the calculus. For example, at first glance it may appear that, under this restriction, it is not possible to deduce a contradiction from the following (unsatisfiable) temporal problem P given by

$$\begin{aligned} I &= \{\forall x P(x)\}, & \mathcal{U} &= \{\neg Q(c)\}, \\ \mathcal{S} &= \{P(x) \Rightarrow \bigcirc Q(x)\}, & \mathcal{E} &= \emptyset. \end{aligned}$$

However we *can* derive a contradiction because we apply our calculus to $\mathbf{S}(\mathsf{P})$ which contains an additional step clause

$$P(c) \Rightarrow \bigcirc Q(c).$$

A formal statement of completeness follows.

Definition 6 A clause of the form $C \Rightarrow \bigcirc\mathbf{false}$, where C is of the same form as in (8), is called *a final clause*.

Lemma 4 Let Δ be a proof of a final clause $C \Rightarrow \bigcirc\mathbf{false}$ by the rules of fine-grained resolution, *except the clause conversion rule,* from a set of step clauses $\mathcal{S}$ and a set of universal clauses $\mathcal{U}$. Then there exists a merged e-derived step clause $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ such that $\mathcal{B} \cup \mathcal{U} \models \mathbf{false}$ and $\mathcal{A} = \widetilde{\exists} C$, where $\widetilde{\exists}$ denotes existential quantification over all free variables.

Proof Let

$$\{P_i(x_i) \ \Rightarrow \ \bigcirc M_i(x_i) \mid i = 1 \ldots K\},$$
$$\{p_i \ \Rightarrow \ \bigcirc l_i \mid i = 1 \ldots L\}$$

be the set of all step clauses from $\mathcal{S}$ involved in Δ where $p_i \Rightarrow \bigcirc l_i$ denotes either a ground step clause, or an e-derived step-clause of the form (9) added by preprocessing (w.l.o.g., we assume that all the variables x_1,.., x_K are pairwise distinct). We assume that Δ is *tree-like*, that is, no clause in Δ is used more than once as an assumption for an inference rule; we may make copies of the clauses in Δ in order to make it tree-like.

Note that (by accumulating the most general unifiers used in the proof) it is possible to construct a finite set of instances of these clauses (and universal clauses) such that there exists a tree-like proof of $C \Rightarrow \bigcirc\mathbf{false}$ from this new set of clauses and all most general unifiers used in the proof are empty [6] . That is, there exist substitutions

[6] The condition that premises of the non-ground binary resolution rule should be variable disjoint may be violated here; note, however, that this condition is needed for *completeness*, not *correctness*.

$\{\sigma_{i,j} \mid i = 1 \ldots K, j = 1 \ldots s_i\}$ such that

$$\begin{aligned} &\{P_i(x_i)\sigma_{i,j} \Rightarrow \bigcirc M_i(x_i)\sigma_{i,j} \mid i = 1 \ldots K, j = 1 \ldots s_i\}, \\ &\qquad \{p_i \Rightarrow \bigcirc l_i \mid i = 1 \ldots L\} \end{aligned} \tag{10}$$

(together with some instances of universal clauses) contribute to the proof of $C \Rightarrow \bigcirc\mathbf{false}$ where all most general unifiers used in the proof are empty, and, furthermore,

$$C = \bigwedge_{i=1}^{K} \bigwedge_{j=1}^{s_i} P_i(x_i)\sigma_{i,j} \wedge \bigwedge_{i=1}^{L} p_i.$$

Note further (induction) that due to our restriction on the step resolution rule, for any i, j, the substitution $\sigma_{i,j}$ maps x_i into a free variable.

Let us group the instances of the step clauses according to the value of the substitutions. We introduce an equivalence relation Σ on the clauses from (10) as follows: For every i, j, i', j' we have $\big(P_i(x_i)\sigma_{i,j} \Rightarrow \bigcirc M_i(x_i)\sigma_{i,j},\ P_{i'}(x_{i'})\sigma_{i',j'} \Rightarrow \bigcirc M_{i'}(x_{i'})\sigma_{i',j'}\big) \in \Sigma$ if, and only if, $x_i\sigma_{i,j} = x_{i'}\sigma_{i',j'}$ (it can be easily checked that Σ is indeed an equivalence relation). Let N be the number of equivalence classes of (10) by Σ; let I_k be the set of indexes of the k-th equivalence class (we refer to clauses from (10) by indexes of the corresponding substitutions).

Let $C_k = \bigwedge_{(i,j)\in I_k} P_i(x_i)\sigma_{i,j}$, for every k, $1 \le k \le N$; let $C_0 = \bigwedge_{i=1}^{L} p_i$. Note that $C = \bigwedge_{k=1}^{N} C_k \wedge C_0$ and this partition of C is disjoint. Let $D_k = \bigwedge_{(i,j)\in I_k} M_i(x_i)\sigma_{i,j}$, let $D_0 = \bigwedge_{i=1}^{L} l_i$, let $D = \bigwedge_{k=1}^{N} D_k \wedge D_0$. Note that $\widetilde{\forall} D \wedge \mathcal{U} \models \mathbf{false}$. Note further that if we replace the free variable of D_k with a fresh constant, c_k, there still exists a refutation from $\bigwedge_{k=1}^{N} D(c_k) \wedge D_0$ and universal clauses (with most general unifiers applied to universal and intermediate clauses only). It follows that $\bigwedge_{k=1}^{N} \exists x D_k(x) \wedge D_0 \wedge \mathcal{U} \models \mathbf{false}$.

It suffices to note that $(\bigwedge_{k=1}^{N} \exists x C_k(x) \wedge C_0) \Rightarrow \bigcirc(\bigwedge_{k=1}^{N} \exists x D_k(x) \wedge D_0)$ is a merged e-derived step clause. ❑

Lemma 5 Let $\mathsf{P} = \langle \mathcal{U}, I, \mathcal{S}, \mathcal{E} \rangle$ be a monodic temporal problem and $\mathbf{S} = \mathbf{S}(\mathsf{P})$ be the result of preprocessing. Let $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ be a merged e-derived step clause such that $\mathcal{B} \cup \mathcal{U} \models \mathbf{false}$. Then there exists a final clause $C \Rightarrow \bigcirc\mathbf{false}$, derived by fine-grained resolution from $\mathbf{S}$, such that $\mathcal{A}$ implies $\widetilde{\exists} C$.

Proof We show that the final clause needed, $C \Rightarrow \bigcirc\mathbf{false}$, can be derived from $\mathbf{S}$ by the deduction rules of fine-grained resolution except the clause conversion rule.

The clause $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ is merged from derived clauses of the form (6):

$$\{\exists x \bigwedge_{j=1}^{k} P_{i_j}(x) \Rightarrow \bigcirc \exists x \bigwedge_{j=1}^{k} M_{i_j}(x) \mid i = 1 \ldots K\}$$

and original ground step clauses and derived clauses of the form (5) (again, as in the proof of Lemma 4, $p_i \Rightarrow \bigcirc l_i$ denotes either an original ground step clause or a clause of the form (5)):

$$\{p_i \Rightarrow \bigcirc l_i \mid i = 1 \ldots L\}.$$

(We simplify indexing for the sake of presentation.) The result of merging is:

$$\underbrace{\bigwedge_{i=1}^{K} \left(\exists x \bigwedge_{j=1}^{s_i} P_{i_j}(x) \right) \wedge \bigwedge_{i=1}^{L} p_i}_{\mathcal{A}} \Rightarrow \bigcirc \underbrace{\left(\bigwedge_{i=1}^{K} \left(\exists x \bigwedge_{j=1}^{s_i} M_{i_j}(x) \right) \wedge \bigwedge_{i=1}^{L} l_i \right)}_{\mathcal{B}}.$$

Since $\mathcal{B} \cup \mathcal{U} \models \textbf{false}$, we have $\bigwedge_{i=1}^{K} \bigwedge_{j=1}^{s_i} M_{i_j}(c_i) \wedge \bigwedge_{i=1}^{L} l_i \wedge \mathcal{U} \models \textbf{false}$ where $c_1, \ldots, c_K$ are fresh (Skolem) constants.

Consider now a set of instances of step clauses

$$\begin{aligned} &\{P_{i_j}(c_i) \Rightarrow \bigcirc M_{i_j}(c_i) \mid i = 1 \ldots K, j = 1 \ldots s_i\}, \\ &\{p_i \Rightarrow \bigcirc l_i \mid i = 1 \ldots L\}. \end{aligned}$$

Let Δ be a (first-order) resolution proof of **false** from $\mathcal{U}$ and the following set of clauses $\{M_{i_j}(c_i) \mid i = 1 \ldots K, j = 1 \ldots s_i\} \cup \{l_i \mid i = 1 \ldots L\}$.

Let $\{M_j(c_i) \mid (i, j) \in I\} \cup \{l_i \mid i \in J\}$, for some sets of indexes I and J, be its subset containing all clauses involved in Δ (and only the clauses involved in Δ). It is easy to see that there exists a proof Γ by fine-grained step resolution from

$$\begin{aligned} &\{P_j(c_i) \Rightarrow \bigcirc M_j(c_i) \mid (i, j) \in I\}, \\ &\{p_i \Rightarrow \bigcirc l_i \mid i \in J\} \end{aligned}$$

(and universal clauses) of a final clause $C \Rightarrow \bigcirc\textbf{false}$, where

$$C = \bigwedge_{(i,j) \in I} P_j(c_i) \wedge \bigwedge_{j \in J} p_i.$$

Let us remind that by the lifting theorem (cf. e.g. [23], page 79), there exists a non-ground (first-order) refutation Δ' from $\{M_j(x_j) \mid (i, j) \in I\} \cup \{l_i \mid i \in J\}$, such that $\Delta' \leq_s \Delta$ in the terminology of [23], that is, Δ' is of the same length as Δ, and every clause C'_i of Δ' is more general than the corresponding clause C_i of Δ.

It can be seen that the lifting theorem can be transferred to fine-grained inferences, and there exists a proof Γ' from the set of original step clauses

$$\{P_j(x_j) \Rightarrow \bigcirc M_j(x_j) \mid (i,j) \in I\},$$
$$\{p_i \Rightarrow \bigcirc l_i \mid i \in J\}$$

(and universal clauses) of a final clause $C' \Rightarrow \bigcirc\mathbf{false}$ such that $\Gamma' \leq_s \Gamma$, that is, every intermediate clause $C'_i \Rightarrow \bigcirc D'_i$ from Γ' is more general than the corresponding clause from Γ. The only difficulty in transferring the lifting theorem to fine-grained resolution is to ensure the requirement on most general unifiers imposed by our inference system. Note that none of the (Skolem) constants $c_1,\ldots,c_K$ occurs in Γ'. If, in the proof Γ', the requirement on most general unifiers is violated, and a constant or a functional term is substituted by an most general unifier into a variable occurring in the left-hand side of a step clause, this clause would not be a generalisation of any clause from Γ.

This implies the conclusion of the lemma. ❑

Lemma 6 Let $\mathsf{P} = \langle \mathcal{U}, I, \mathcal{S}, \mathcal{E}\rangle$ be a monodic temporal problem and $\mathbf{S} = \mathbf{S}(\mathsf{P})$ be the result of preprocessing. Let $C \Rightarrow \bigcirc\mathbf{false}$ be an arbitrary final clause derived by fine-grained step resolution from $\mathbf{S}$. Then there exists a derivation $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \ldots$ by the step resolution rule of $\mathfrak{I}_e$ and a merged e-derived step clause $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ such that $\mathcal{B} \cup \mathcal{U}_i \models \mathbf{false}$, for some $i \geq 0$, and $\mathcal{A} = \widetilde{\exists} C$.

Proof Since $C \Rightarrow \bigcirc\mathbf{false}$ is derivable, there exists a proof Γ of this final clause, by fine-grained resolution. We prove the lemma by induction on the number of applications of the clause conversion rule in Γ. For the base case, when the clause conversion rule does not contribute to Γ, the statement of the Lemma follows directly from Lemma 4.

Suppose that we have proved the lemma for proofs containing less than n applications of the clause conversion rule, and let Γ contain n such applications. Let us consider the first application of the clause conversion rule in Γ, and let this rule be applied to a final clause $D \Rightarrow \bigcirc\mathbf{false}$. By the induction hypothesis, there exists a derivation $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \ldots \mathcal{U}_k$ by the step resolution rule in $\mathfrak{I}_e$ and a merged e-derived step clause $\mathcal{A}' \Rightarrow \bigcirc\mathcal{B}'$ such that $\mathcal{B}' \cup \mathcal{U}_k \models \mathbf{false}$ and $\mathcal{A}' = \widetilde{\exists} D$. Let the temporal problem P' be $\langle \mathcal{U}_k, I, \mathcal{S}, \mathcal{E}\rangle$. Since $\mathcal{U} \subset \mathcal{U}_k$, there exists a proof Γ' of $C \Rightarrow \bigcirc\mathbf{false}$ from $\mathbf{S}(\mathsf{P}')$ which contains $n-1$ applications of the clause conversion rule. Thus, the Lemma follows from the induction hypothesis. ❑

Lemma 7 Let $\mathsf{P} = \langle \mathcal{U}, I, \mathcal{S}, \mathcal{E}\rangle$ be a monodic temporal problem and $\mathbf{S} = \mathbf{S}(\mathsf{P})$ be the result of preprocessing. Let $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \ldots$ be a derivation by the step resolution rule of $\mathfrak{I}_e$. Let $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ be a merged e-derived step clause such that $\mathcal{B} \cup \mathcal{U}_i \models \mathbf{false}$, for some $i \geq 0$. Then there exists a final clause $C \Rightarrow \bigcirc\mathbf{false}$, derived by fine-grained resolution from $\mathbf{S}$, such that $\mathcal{A}$ implies $\widetilde{\exists} C$.

Proof The Lemma easily follows from Lemma 5 by induction on i. ❑

Lemma 6 ensures the soundness of fine-grained step resolution. Lemma 7 says that the conclusion of an application of the clause conversion rule, $\neg C$, subsumes the conclusion of an application of the step resolution rule of $\mathfrak{I}_e$, $\neg\mathcal{A}$.

Theorem 8 The calculus consisting of the rules of fine-grained step resolution, together with the ground and non-ground eventuality resolution rules, is sound and complete for the monodic fragment over expanding domains.

Proof Soundness follows from Lemma 6.

Suppose now that a temporal problem $\mathsf{P} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$ is unsatisfiable. Then there exists a terminating derivation $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \mathcal{U}_2 \subseteq \ldots \subseteq \mathcal{U}_n$. Let us for every $i = 1, \ldots, n$ consider the formula $u_i = \mathcal{U}_i \setminus \mathcal{U}_{i-1}$ (that is, u_i is the formula added to $\mathcal{U}_{i-1}$ by a rule of $\mathfrak{I}_e$; in particular, $u_n = \mathbf{false}$). Let u_k be derived by the step resolution rule. Then, by Lemma 7, there exists a final clause $C_k \Rightarrow \bigcirc\mathbf{false}$ derived by fine-grained resolution from $\mathbf{S}(\langle \mathcal{U}_{k-1}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle)$ such that $\neg u_k$ implies $\widetilde{\exists} C_k$, which means $\widetilde{\forall}\neg C_k$ implies u_k, where $\widetilde{\forall}$ denotes universal quantification over free variables. Let $u'_k = \widetilde{\forall}\neg C_k$. It can be easily seen that the result of replacing u with u' in the derivation $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \mathcal{U}_2 \subseteq \ldots \subseteq \mathcal{U}_n$ is still a correct terminating derivation in $\mathfrak{I}_e$. ❑

Note 3 The proof of Lemma 4 given above might be hard to fulfil in the presence of *redundancy deletion*. As a remedy, we suggest considering *constrained calculi*, that is, resolution over constrained clauses with constraint inheritance (see [24]). It is known that such inference systems are complete and, moreover, compatible with practical redundancy elimination rules (see e.g. [25]). Here, we take into account that there are no clauses with equality, and therefore all sets are *well-constrained* in the terminology of [25].

Then, instead of ground clauses of the form

$$P_j(c_i) \;\Rightarrow\; \bigcirc M_j(c_i)$$

we consider their *constrained* representations

$$P_j(x_i) \;\Rightarrow\; \bigcirc M_j(x_i) \cdot \{x_i = c_i\}.$$

Recall that, in accordance with the semantics of constrained clauses, a clause $C \cdot T$ represents the set of all ground instances $C\sigma$ where σ is a solution of T. In our case, there is exactly one solution of $x_i = c_i$ given by the substitution $\{x_i \mapsto c_i\}$. So, the semantics of

$$P_j(x_i) \;\Rightarrow\; \bigcirc M_j(x_i) \cdot \{x_i = c_i\}$$

is just

$$P_j(c_i) \;\Rightarrow\; \bigcirc M_j(c_i).$$

So, all clauses originating from the universal part have empty constraints and all temporal clauses have constraints as defined above, and there exists a non-ground proof of a constrained final clause with constraint inheritance. Note that the (Skolem) constants $c_1, \ldots, c_k$ may only occur in constraints but not in clauses themselves. It suffices to note that in this case inferences with constraint inheritance admit only two kinds of substitutions into x_i: either $\{x_i \mapsto c_i\}$ (however this will not occur because c_i occurs only in constraints), or $\{x_i \mapsto x_{i'}\}$ where $x_{i'}$ is bound by the same constraint $\{x_{i'} = c_i\}$. The case of matching x_i and y where y originates from the universal part is solved by the substitution $\{y \mapsto x_i\}$. A non-ground inference of a final clause, satisfying the conditions on substitutions in the fine-grained resolution rules, can be extracted from this constrained proof implying, thus, the conclusion of Lemma 7.

5.2 Loop search

Now we use fine-grained step resolution to find the appropriate set of full e-merged clauses to apply the ground or non-ground eventuality resolution rule. Our method is based on the breadth-first search algorithm presented in [6]; we give the algorithm in Fig. 1. (In turn, the breadth-first search algorithm for *monodic* temporal resolution is essentially based on the search algorithm for *propositional* temporal resolution [26].)

The results from [6] (lemmas 7–10, Theorem 9) can be summarised as follows.

Theorem 9 The BFS algorithm terminates subject to termination of all first-order validity tests. If BFS returns a value other than **false**, then its output is a loop formula in $L(x)$. Temporal resolution is complete if we restrict ourselves to loops found by the BFS algorithm.

In order to effectively find a loop by the breadth-first search algorithm, given a formula with at most one free variable $\Phi(x)$ we have to be able to find the set of all full e-merged clauses of the form $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc \mathcal{B}(x))$ such that the formula

$$\forall x(\mathcal{B}(x) \wedge \mathcal{U} \Rightarrow \Phi(x))$$

is valid (where $\Phi(x) = H(x) \wedge \neg L(x)$ and $H(x)$ is a disjunction of the left-hand sides of some full e-merged step clauses).

Let $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc \mathcal{B}(x))$ be a full e-merged step clause such that $\forall x(\mathcal{B}(x) \wedge \mathcal{U} \Rightarrow \Phi(x))$. Note that $\forall x(\mathcal{B}(x) \wedge \mathcal{U} \Rightarrow \Phi(x))$ is valid if, and only if, $\exists x(\mathcal{B}(x) \wedge \mathcal{U} \wedge \neg\Phi(x))$ is unsatisfiable.

Definition 7 Let c^l be a distinguished constant to be used in loop search that we

Function BFS

Input: A temporal problem P and an eventuality clause $\Diamond L(x) \in \mathcal{E}$.

Output: A formula $H(x)$ with at most one free variable.

Method:
(1) Let $H_0(x) = \mathbf{true}$; $N_0 = \emptyset$; $i = 0$.
(2) Let $N_{i+1} = \{\forall x(\mathcal{A}_j^{(i+1)}(x) \Rightarrow \bigcirc\mathcal{B}_j^{(i+1)}(x))\}_{j=1}^k$ be the set of *all* full merged step clauses such that for every $j \in \{1\ldots k\}$, $\forall x(\mathcal{U} \wedge \mathcal{B}_j^{(i+1)}(x) \Rightarrow (\neg L(x) \wedge H_i(x)))$ holds. (The set N_{i+1} possibly includes the degenerate clause $\mathbf{true} \Rightarrow \bigcirc\mathbf{true}$ in the case $\mathcal{U} \models \forall x(\neg L(x) \wedge H_i(x))$.)
(3) If $N_{i+1} = \emptyset$, return **false**; else let $H_{i+1}(x) = \bigvee_{j=1}^{k} (\mathcal{A}_j^{(i+1)}(x))$.
(4) If $\forall x(H_i(x) \Rightarrow H_{i+1}(x))$ return $H_{i+1}(x)$.
(5) $i = i + 1$; goto 2.

Fig. 1. Breadth First Search algorithm.

call the *loop constant*. We assume that the loop constant does not occur in a given problem and is not used for skolemisation.

Definition 8 Let us define a *transformation for loop search* on a set of universal and step clauses $\mathbf{S}$ as follows. $\mathrm{LT}(\mathbf{S})$ is the minimal set of clauses containing $\mathbf{S}$ such that for every *original* non-ground step clause $(P(x) \Rightarrow \bigcirc M(x)) \in \mathbf{S}$, the set $\mathrm{LT}(\mathbf{S})$ contains the clause

$$P(c^l) \Rightarrow \bigcirc M(c^l). \tag{11}$$

We add the clause[7] $\mathbf{true} \Rightarrow \bigcirc\neg\Phi(c^l)$ to $\mathrm{LT}(\mathbf{S})$ and apply the rules of fine-grained step resolution *except the clause conversion rule* to it.

Lemma 10 Let $\mathbf{S}$ be a set of universal and step clauses, and let $C \Rightarrow \bigcirc\mathbf{false}$ be a final clause derived by the rules of fine-grained step resolution *except the clause conversion rule* from $\mathrm{LT}(\mathbf{S}) \cup \{\mathbf{true} \Rightarrow \bigcirc\neg\Phi(c^l)\}$ such that at least one of the clauses originating from $\mathbf{true} \Rightarrow \bigcirc\neg\Phi(c^l)$ is involved in the derivation. Then there exists a full e-merged (from $\mathbf{S}$) clause $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc\mathcal{B}(x))$ such that the formula $\forall x(B(x) \wedge \mathcal{U} \Rightarrow \Phi(x))$ is valid and $\mathcal{A}(x) = (\widetilde{\exists} C)\{c^l \mapsto x\}$.

Proof By Lemma 6, there exists a merged (from $LT(S)$) e-derived clause $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ such that $\{\neg\Phi(c^l)\} \cup \mathcal{B} \cup \mathcal{U} \models \mathbf{false}$ and $\mathcal{A} = \widetilde{\exists} C$. It suffices to notice that $\forall x((\mathcal{A} \Rightarrow \bigcirc\mathcal{B})\{c^l \mapsto x\})$ is a full merged (from S) step clause and $\exists x(\neg\Phi(x) \wedge \mathcal{B}\{c^l \mapsto x\} \wedge \mathcal{U})$ is unsatisfiable. ❑

Lemma 11 Let $\mathbf{S}$ be a set of universal and step clauses, and let $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc\mathcal{B}(x))$ be a full e-merged (from $\mathbf{S}$) step clause such that $\forall x(\mathcal{B}(x) \wedge \mathcal{U} \Rightarrow \Phi(x))$. Then there

[7] In fact, a set of clauses since $\neg H(x)$, and consequently $\neg\Phi(x)$, is a set of first-order clauses.

Function FG-BFS

Input: A set $\mathbf{S}$ of universal and step clauses, saturated by fine-grained resolution and an eventuality clause $\Diamond L(x) \in \mathcal{E}$.

Output: A formula $H(x)$ with at most one free variable.

Method:
(1) Let $H_0(x) = \mathbf{true}$; $N_0 = \emptyset$; $i = 0$.
(2) Let $\mathbf{S_{i+1}} = \mathrm{LT}(\mathbf{S}) \cup \{\mathbf{true} \Rightarrow \bigcirc(\neg H_i(c^l) \vee L(c^l))\}$. Apply the rules of fine-grained step resolution *except the clause conversion rule* to $\mathbf{S_{i+1}}$. If we obtain a contradiction, then return the loop **true** (in this case $\forall x \neg L(x)$ is implied by the universal part).
Otherwise let $N_{i+1} = \{C_j \Rightarrow \bigcirc\mathbf{false}\}_{j=1}^{k}$ be the set of all *new* final clauses from $\mathbf{S_{i+1}}$.
(3) If $N_{i+1} = \emptyset$, return **false**; else let $H_{i+1}(x) = \bigvee_{j=1}^{k} C_j\{c^l \to x\}$.
(4) If $\forall x(H_i(x) \Rightarrow H_{i+1}(x))$ return $H_{i+1}(x)$.
(5) $i = i + 1$; goto 2.

Fig. 2. Breadth-first search using fine-grained step resolution.

exists a derivation by the rules of fine-grained step resolution *except the clause conversion rule* from $\mathrm{LT}(\mathbf{S})$ of a final clause $C \Rightarrow \bigcirc\mathbf{false}$ such that $\forall x(\mathcal{A}(x)$ implies $(\widetilde{\exists}C)\{c^l \mapsto x\})$.

Proof The proof is analogous to the proof of lemmas 5 and 7. As we already noticed, $\exists x(\mathcal{B}(x) \wedge \mathcal{U} \wedge \neg\Phi(x))$ is unsatisfiable, and this can be checked by a first-order resolution procedure. Since c^l does not occur in the problem, we can skolemise this existential quantifier with c^l. We then lift all Skolem constants but c^l. ❑

Then the BFS algorithm can be reformulated as shown in Fig. 2.

The proof of the following two statements is a direct adaptation of the proofs from [6] taking into consideration lemmas 10 and 11 and arguments similar to those given in the proof of Theorem 8.

Lemma 12 The FG-BFS algorithm terminates provided that all calls of saturation by step resolution terminate. If FG-BFS returns a value other than **false**, then its output is a loop formula in $L(x)$.

Note 4 Termination of calls by step resolution can be achieved for the cases when there exists a (first-order) resolution decision procedure [27] for formulae in the universal part, see also [28].

Theorem 13 The calculus consisting of the rules of fine-grained step resolution, together with the (both ground and non-ground) eventuality resolution rule, is complete for the monodic fragment over expanding domains even if we restrict ourselves to loops found by the FG-BFS algorithm.

5.3 Example

Let us consider a monodic temporal problem P given by

$$\mathcal{U} = \left\{ \begin{array}{l} \forall x(B(x) \Rightarrow A(x) \wedge \neg L(x)), \\ l \Rightarrow \exists x A(x) \end{array} \right\}, \qquad I = \emptyset,$$

$$\mathcal{S} = \{s1 : A(x) \Rightarrow \bigcirc B(x)\}, \qquad \mathcal{E} = \{e1 : \Diamond L(x),\ e2 : \Diamond l\}.$$

We chose such a trivial example specifically to be able to demonstrate thoroughly the steps of our proof search algorithm.
We clausify $\mathcal{U}$ and obtain

$$\mathcal{U}^s = \left\{ \begin{array}{l} u1 : (\neg B(x) \vee A(x)), \\ u2 : (\neg B(x) \vee \neg L(x)), \\ u3 : \neg l \vee A(c) \end{array} \right\}.$$

- Step resolution
  We can deduce the following clauses by fine-grained step resolution:

$$\begin{array}{ll} s2 : A(x) \Rightarrow \bigcirc A(x) & (\,s1, u1) \\ s3 : A(x) \Rightarrow \bigcirc \neg L(x) & (\,s1, u2) \end{array}$$

  The set of clauses is saturated. Now we try finding a loop in $\Diamond L(x)$.
- Loop search (1)
  The set $\mathbf{S} = \{u1, u2, u3, s1, s2, s3\}$; $H_0(x) = \mathbf{true}$; $N_0 = \emptyset$; $i = 0$. $\mathrm{LT}(\mathbf{S}) = \{lt1 : A(c^l) \Rightarrow \bigcirc B(c^l)\}$.
  We deduce the following clauses by fine-grained step resolution (except the clause conversion rule) from $\mathbf{S_1} = \mathrm{LT}(\mathbf{S}) \cup \{l1 : \mathbf{true} \Rightarrow \bigcirc L(c^l)\}$:

$$\begin{array}{ll} l2 : A(c^l) \Rightarrow \bigcirc A(c^l) & (\,lt1, u1) \\ l3 : A(c^l) \Rightarrow \bigcirc \neg L(c^l) & (\,lt1, u2) \\ l4 : \mathbf{true} \Rightarrow \bigcirc \neg B(c^l) & (\,u2, l1) \\ l5 : A(c^l) \Rightarrow \bigcirc \mathbf{false} & (\,l3, l1) \end{array}$$

  The set of clauses is saturated. Then $N_1 = \{A(c^l) \Rightarrow \bigcirc \mathbf{false}\}$, $H_1(x) = A(x)$. Obviously, $\forall x(H_0(x) \Rightarrow H_1(x))$ is not true.
  Now the set $\mathbf{S_2} = \mathrm{LT}(\mathrm{S}) \cup \{l6 : \mathbf{true} \Rightarrow \bigcirc(\neg A(c^l) \vee L(c^l))\}$ and we deduce

from it the following:

$$l7 : A(c^l) \Rightarrow \bigcirc A(c^l) \qquad (\,lt1, u1)$$
$$l8 : A(c^l) \Rightarrow \bigcirc \neg L(c^l) \qquad (\,lt1, u2)$$
$$l9 : \textbf{true} \Rightarrow \bigcirc(\neg B(c^l) \vee L(c^l)) \qquad (\,u1, l6)$$
$$l10 : \textbf{true} \Rightarrow \bigcirc(\neg B(c^l) \vee \neg A(c^l)) \qquad (\,u2, l6)$$
$$l11 : A(c^l) \Rightarrow \bigcirc L(c^l) \qquad (\,l7, l6)$$
$$l12 : A(c^l) \Rightarrow \bigcirc \neg A(c^l) \qquad (\,l8, l6)$$
$$l13 : \textbf{true} \Rightarrow \bigcirc \neg B(c^l) \qquad (\,u2, l9)$$
$$l14 : A(c^l) \Rightarrow \bigcirc \neg B(c^l) \qquad (\,l8, l9)$$
$$l15 : A(c^l) \Rightarrow \bigcirc \textbf{false} \qquad (\,l8, l11)$$

The set of clauses is saturated. $N_2 = \{A(c^l) \Rightarrow \bigcirc\textbf{false}\}$, $H_2(x) = A(x)$. As $\forall x(H_1(x) \Rightarrow H_2(x))$, the loop is $A(x)$.

- Eventuality resolution
  We can apply now the eventuality resolution rule whose conclusion is

$$u4 : \neg A(x).$$

- Step resolution

$$u5 : \neg l \;(\,u3, u4)$$

- Loop search (2)
  $\mathbf{S} = \{u1, u2, u3, u4, u5, s1, s2, s3\}$; $H_0(x) = \textbf{true}$; $N_0 = \emptyset$; $i = 0$; $\mathrm{LT}(\mathbf{S}) = \{lt1 : A(c^l) \Rightarrow \bigcirc B(c^l)\}$; $\mathbf{S_1} = \mathrm{LT}(\mathbf{S}) \cup \{l16 : \textbf{true} \Rightarrow \bigcirc l\}$; and we can deduce:

$$l17 : \textbf{true} \Rightarrow \bigcirc\textbf{false} \;(\,l16, u5)$$

  that is, a contradiction. The loop is **true**.
- Eventuality resolution
  We can apply now the eventuality resolution rule whose conclusion is $\neg$**true**.

  The problem is unsatisfiable.

Note 5 As the example above shows, the presence of clauses of the form (9), introduced by preprocessing, and (11), introduced by the transformation for loop search, might lead to repeated derivations (with free variables and with constants). This can be avoided, however, if instead of generating these clauses, we relax the conditions on substitutions in the definition of rules of fine-grained resolution by allowing original constants and the loop constant to be substituted to variables occurring in the left-hand side of a step clause. It can be seen that the set of derived final clauses would be the same.

Taking into consideration this note, we do not use the reduction for loop search, and clauses $l2$, $l3$, $l7$, $l8$ would not be derived. Instead, at the first iteration of BFS on $L(x)$, we would deduce the following clauses from $\mathbf{S_1} = \mathbf{S} \cup \{l1 : \mathbf{true} \Rightarrow \bigcirc L(c^l)\}$:

$$
\begin{array}{ll}
l4' : \mathbf{true} \Rightarrow \bigcirc \neg B(c^l) & (u2, l1) \\
l5' : A(c^l) \Rightarrow \bigcirc \mathbf{false} & (s3, l1);
\end{array}
$$

and at the second iteration from $\mathbf{S_2} = \mathrm{LT(S)} \cup \{l6 : \mathbf{true} \Rightarrow \bigcirc(\neg A(c^l) \vee L(c^l))\}$:

$$
\begin{array}{ll}
l9' : \mathbf{true} \Rightarrow \bigcirc(\neg B(c^l) \vee L(c^l)) & (u1, l6) \\
l10' : \mathbf{true} \Rightarrow \bigcirc(\neg B(c^l) \vee \neg A(c^l)) & (u2, l6) \\
l11' : A(c^l) \Rightarrow \bigcirc L(c^l) & (s2, l6) \\
l12' : A(c^l) \Rightarrow \bigcirc \neg A(c^l) & (s3, l6) \\
l13' : \mathbf{true} \Rightarrow \bigcirc \neg B(c^l) & (u2, l9') \\
l14' : A(c^l) \Rightarrow \bigcirc \neg B(c^l) & (s3, l9') \\
l15' : A(c^l) \Rightarrow \bigcirc \mathbf{false} & (s3, l11').
\end{array}
$$

6 Implementation

The deduction rules of fine-grained resolution are close enough to classical first-order resolution to allow us to use first-order resolution provers to provide an implementation of our calculus.

Let P be a temporal problem and $\mathbf{S} = \mathbf{S}(\mathsf{P})$ be the result of preprocessing. For every k-ary predicate, P, occurring in $\mathbf{S}$, we introduce new $(k+1)$-ary predicate $\widetilde{P}$. Let ϕ be a first-order formula in the vocabulary of $\mathbf{S}$. We denote by $[\phi]^0$ the result of substituting of all occurrences of predicates in ϕ with their "tilded" counterparts with "0" as the first argument (e.g. $P(x_1, x_2)$ is replaced with $\widetilde{P}(0, x_1, x_2)$), where 0 is a constant; we assume that 0 does not occur elsewhere in $\mathbf{S}$. By $[\phi]^t$ we denote the result of substituting of all occurrences of predicates in ϕ with their "tilded" counterparts with "t" as the first argument, where t is a variable not occurring in $\mathbf{S}$. The variable t is assumed to be bound by the universal quantifier.

Now, in order to realise fine-grained step resolution by means of classical first-order resolution, we define a set of first-order clauses $\mathbf{FO}(\mathbf{S})$ as follows.

- For every initial clause C from $\mathbf{S}$, the clause $[C]^0$ is in $\mathbf{FO}(\mathbf{S})$.
- For every universal clause D from $\mathbf{S}$, the clause $[D]^t$ is in $\mathbf{FO}(\mathbf{S})$.

- For every ground step clause $p \Rightarrow l$ from $\mathbf{S}$, the clause $\neg\widetilde{p}(t) \vee \widetilde{l}(s(t))$ is in $\mathbf{FO}(\mathbf{S})$, and for every non-ground step clause $P(x) \Rightarrow \bigcirc M(x)$, the clause $\neg\widetilde{P}(t,x) \vee \widetilde{M}(s(t),h(x))$ is in $\mathbf{FO}(\mathbf{S})$, where s and h are function symbols not occurring in $\mathbf{S}$.

It is a straightforward routine to check that fine-grained step resolution on $\mathbf{S}$, including (implicitly) the clause conversion rule, is modelled by classical ordered first-order resolution with selection [22] on $\mathbf{FO}(\mathbf{S})$ together with an additional requirement: If a clause contains a literal whose first argument starts with the function symbol s, the only literals from this clause that are allowed to resolve with should also have the first argument starting with the function symbol s. This requirement can be easily enforced by the choice of an appropriate literal ordering and clause selection function.

Transformation for loop search differs in the way that we treat step clauses. When performing loop search, we do not want the clause conversion rule to apply, and this rule is implicit in the construction above. To avoid that, we transform a ground step clause $p \Rightarrow l$ from LT($\mathbf{S}$) to $\neg\widetilde{p}(0) \vee \widetilde{l}(s(x))$ and non-ground step clause $P(x) \Rightarrow \bigcirc M(x)$ to $\neg\widetilde{P}(0,x) \vee \widetilde{M}(s(x),h(x))$. Recall that initial clauses do not contribute to loop search. Then clauses consisting only of literals whose first argument is 0 exactly corresponds to final clauses (up to negation).

7 Constant domains

In order to be able to use our procedure for establishing unsatisfiability of *constant domain* problems, we simply reduce satisfiability over constant domains to satisfiability over expanding domains.

Let $\mathsf{P} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$ be a temporal problem. The temporal problem Exp(P) is defined as follows. The universal, initial, and eventuality parts of Exp(P) are that of P; the step part consists of all step clauses from $\mathcal{S}$ *together with* all derived step clauses of the form (7). (In fact, in order to fit the definition of a temporal problem, we have to rename complex expressions in the derived clauses.)

Proposition 14 A temporal problem P is satisfiable over constant domains if, and only if, the temporal problem Exp(P) is satisfiable over constant domains.

Proof Over constant domains, derived step clauses of the form (7) are logical consequences of the step part $\mathcal{S}$. ❑

Proposition 15 For any temporal problem P, the temporal problem Exp(P) is satisfiable over constant domains if, and only if, the temporal problem Exp(P) is satisfiable over expanding domains.

Proof Follows from the comparison of the calculi $\mathfrak{I}_e$ and $\mathfrak{I}_c$. Indeed, any refutation of the problem P in $\mathfrak{I}_c$ can be simulated by a refutation of $\mathrm{Exp}(\mathsf{P})$ in $\mathfrak{I}_e$ and vice versa. ❑

Example 6 The following temporal problem P, which is unsatisfiable over constant domains,

$$\mathcal{U} = \left\{ \begin{array}{l} \exists x \neg R(x) \\ \forall x (Q(x) \wedge \exists y \neg P(y) \Rightarrow L(x)) \end{array} \right\}, \quad I = \{\exists x Q(x)\},$$
$$\mathcal{S} = \left\{ \begin{array}{l} Q(x) \Rightarrow \bigcirc Q(x) \\ P(x) \Rightarrow \bigcirc R(x) \end{array} \right\}, \quad \mathcal{E} = \{\Diamond \neg L(x)\}$$

has an expanding domain model. Indeed, let for every $i \geq 1$, the domain D_i be the set of natural numbers $\{1, \ldots, i\}$. Then we choose the values of predicates P, Q, R, L as follows. P is true and L is false on every element of every domain; Q is only true on the element 1 and false for every other element. In the domain D_i, the predicate R is false on the element i and true everywhere else.

Fine-grained resolution would not be able to deduce a contradiction from P; however, after the above transformation, the temporal problem $\mathrm{Exp}(\mathsf{P})$, i.e.,

$$\mathcal{U} = \left\{ \begin{array}{ll} \exists x \neg R(x), & \forall x (P(x) \vee Q(x)) \Rightarrow p_3, \\ \forall x (Q(x) \wedge \exists y \neg R(y) \Rightarrow L(x)), & q_1 \Rightarrow \forall x Q(x), \\ \forall x (Q(x)) \Rightarrow p_1, & q_2 \Rightarrow \forall x R(x), \\ \forall x (P(x)) \Rightarrow p_2, & q_3 \Rightarrow \forall x (Q(x) \vee R(x)) \end{array} \right\},$$

$$\mathcal{S} = \left\{ \begin{array}{ll} Q(x) \Rightarrow \bigcirc Q(x), & p_1 \Rightarrow \bigcirc q_1, \\ P(x) \Rightarrow \bigcirc R(x), & p_2 \Rightarrow \bigcirc q_2, \\ \multicolumn{2}{c}{p_3 \Rightarrow \bigcirc q_3} \end{array} \right\}, \quad I = \{\exists x Q(x)\}, \quad \mathcal{E} = \{\Diamond \neg L(x)\}.$$

becomes unsatisfiable over both constant and expanding domains and its unsatisfiability can be established directly by fine-grained temporal resolution.

Our preliminary experiments show that if the step part of a temporal problem is of a moderate size, this approach performs adequately well.

8 Conclusion

We have here described a fine-grained resolution calculus for monodic first order temporal logics over expanding domains. Soundness of the fine-grained inference steps is easy to prove and completeness is shown relative to the completeness proof for the expanding domain for the non-fine grained version [11]. While the implementation based on the general calculus would involve generating all subsets of the step clauses with which to apply the step and eventuality resolution rules, the fine-grained resolution inference rules can be implemented directly using any appropriate first-order theorem prover for classical logics. This makes the new calculus presented here particularly amenable to efficient implementation. Within this paper, we have also shown a simple method for extending the applicability of the implementation to constant domain problems.

Our future work consists of two main aspects: improved implementation and application in practical examples. At the moment, we are performing some preliminary experiments combining the propositional resolution temporal prover TRP++[29] and Vampire [30]. We are also interested in application, and are currently considering practical verification problems where this work will be of use, both in academia and industry.

Finally, we wish to acknowledge support for this work from EPSRC via research grant GR/R45376/01.

References

[1] A. Pnueli, The Temporal Logic of Programs, in: Proceedings of the Eighteenth Symposium on the Foundations of Computer Science, 1977.

[2] Z. Manna, A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems: Specification, Springer, 1992.

[3] G. J. Holzmann, The model checker Spin, IEEE Trans. on Software Engineering 23 (5) (1997) 279–295.

[4] I. Hodkinson, F. Wolter, M. Zakharyaschev, Decidable fragments of first-order temporal logics, Annals of Pure and Applied Logic 106 (2000) 85–134.

[5] R. Kontchakov, C. Lutz, F. Wolter, M. Zakharyaschev, Temporalising tableaux, *Studia Logica*, to appear.

[6] A. Degtyarev, M. Fisher, B. Konev, Monodic temporal resolution, in: Proc. CADE-19, Vol. 2741 of LNAI, Springer, 2003, available as Technical report ULCS-03-001 from `http://www.csc.liv.ac.uk/research/`.

[7] A. Degtyarev, M. Fisher, Towards first-order temporal resolution, in: Proc. KI 2001, Vol. 2174 of LNCS, Springer, 2001, pp. 18–32.

[8] F. Wolter, M. Zakharyaschev, Qualitative spatio-temporal representation and reasoning: a computational perspective, in: Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 2002, pp. 175–216.

[9] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, M. Zakharyaschev, On the computational complexity of spatio-temporal logics, in: Proceedings of the Sixteenth International Florida Artificial Intelligence Research Symposium Conference (FLAIRS 2003), AAAI Press, 2003, pp. 460–464.

[10] A. Artale, E. Franconi, F. Wolter, M. Zakharyaschev, A temporal description logic for reasoning over conceptual schemas and queries, in: Proceedings of JELIA'02, Vol. 2424 of LNCS, Springer, 2002, pp. 98–110.

[11] A. Degtyarev, M. Fisher, B. Konev, Monodic temporal resolution: the expanding domain case, Tech. Rep. ULCS-03-004, University of Liverpool, Department of Computer Science (2003).

[12] M. Fisher, A normal form for temporal logics and its applications in theorem proving and execution, Journal of Logic and Computation 7 (4) (1997) 429–456.

[13] F. Wolter, M. Zakharyaschev, Decidable fragments of first-order modal logics, Journal of Symbolic Logic 66 (2001) 1415–1438.

[14] S. Merz, Decidability and incompleteness results for first-order temporal logic of linear time, Journal of Applied Non-Classical Logics 2 (1992) 139–156.

[15] F. Wolter, M. Zakharyaschev, Axiomatizing the monodic fragment of first-order temporal logic, Annals of Pure and Applied logic 118 (2002) 133–145.

[16] A. Degtyarev, M. Fisher, A. Lisitsa, Equality and monodic first-order temporal logic, Studia Logica 72 (2) (2002) 147–156.

[17] I. Hodkinson, Monodic packed fragment with equality is decidable, Studia Logica 72 (2002) 185–197.

[18] A. Nonnengart, C. Weidenbach, Computing small clause normal forms, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier, 2001, Ch. 6, pp. 335–370.

[19] M. Fisher, C. Dixon, M. Peim, Clausal temporal resolution, ACM Transactions on Computational Logic 2 (1) (2001) 12–56.

[20] G. Tseitin, On the complexity of derivations in propositional calculus, in: J. Siekmann, G. Wrightson (Eds.), Automation of Reasoning (Classical papers on Computational Logic), Vol. 2, Springer Verlag, 1983, pp. 466–483, original paper (in Russian) appeared in 1968.

[21] A. Degtyarev, M. Fisher, B. Konev, Monodic temporal resolution, ACM Transactions on Computational Logic. To appear.

[22] L. Bachmair, H. Ganzinger, Resolution theorem proving, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier, 2001, Ch. 2, pp. 19–99.

[23] A. Leitsch, The Resolution Calculus, Springer, 1997.

[24] R. Nieuwenhuis, A. Rubio, Theorem proving with ordering and equality constrained clauses, Journal of Symbolic Computation 19 (4) (1995) 321–351.

[25] R. Nieuwenhuis, A. Rubio, Paramodulation-based theorem proving, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier, 2001, Ch. 7, pp. 371–443.

[26] C. Dixon, Temporal resolution using a breadth-first search algorithm, Annals of Mathematics and Artificial Intelligence 22 (1998) 87–115.

[27] C. Fermüller, A. Leitsch, U. Hustadt, T. Tammet, Resolution decision procedures, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Vol. II, Elsevier, 2001, Ch. 25, pp. 1791–1850.

[28] A. Degtyarev, M. Fisher, B. Konev, Exploring the monodic fragment of first-order temporal logic using clausal temporal resolution, Tech. Rep. ULCS-03-012, University of Liverpool, Department of Computer Science (2003).

[29] U. Hustadt, B. Konev, TRP++ 2.0: A temporal resolution prover, in: Proc. CADE-19, Vol. 2741 of LNAI, Springer, 2003.

[30] A. Riazanov, A. Voronkov, Vampire 1.1 (System description), in: R. Goré, A. Leitsch, T. Nipkow (Eds.), Proc. Intl. Joint Conf. on Automated Reasoning, Siena, Italy, no. 2083 in LNCS, Springer-Verlag, 2001.