Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information and Computation 199 (2005) 55–86
Information and Computation
www.elsevier.com/locate/ic

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, UK*
[b] *Department of Computer Science, King's College London, Strand, London, UK*

Received 1 December 2003; revised 31 August 2004
Available online 8 February 2005

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 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. By careful encoding, parts of the calculus can be implemented using 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.

Keywords: Clausal resolution; Temporal logics; Monodic fragment

* Corresponding author.
E-mail addresses: B.Konev@csc.liv.ac.uk (B. Konev), Anatoli@dcs.kcl.ac.uk (A. Degtyarev), C.Dixon@csc.liv.ac.uk (C. Dixon), M.Fisher@csc.liv.ac.uk (M. Fisher), U.Hustadt@csc.liv.ac.uk (U. Hustadt).
[1] On leave from Steklov Institute of Mathematics at St. Petersburg.

0890-5401/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2004.10.005

1. Introduction

In its propositional form, linear, discrete *temporal logic* [15,38,43] has been used in a wide variety of areas within Computer Science and Artificial Intelligence, for example robotics [48], databases [51], hardware verification [33], and agent-based systems [45]. In particular, propositional temporal logics have been applied to:

- the specification and verification of reactive (e.g., distributed or concurrent) systems [38];
- the synthesis of programs from temporal specifications [36,44];
- the semantics of executable temporal logic [18,19];
- algorithmic verification via model-checking [6,32]; and
- knowledge representation and reasoning [2,16,53].

Although recognised as both a much more powerful and natural formalism [25,27], *first-order* temporal logic has generally been avoided due to completeness problems. In particular, the set of valid formulae of this logic is not recursively enumerable [1,49,50]. However, recent work by Hodkinson et al. [31] has shown that a specific fragment of first-order temporal logic, termed the *monodic* fragment, has the completeness (and 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 proof methods. Concerning proof methods for monodic temporal logics, general tableau and resolution calculi have already been defined, in [35] and [7,8], 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 methods 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 [28,55] and temporal description logics [3], namely those models that have *expanding domains*. In such models, the domains over which first-order terms range can only 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 implementable calculus, consider its properties and extend the results to constant-domain problems. We also describe an implementation of this fine-grained calculus and its use on a range of problems. This represents the first practically useful tool for handling monodic first-order temporal logics.

The organisation of the paper is the following. In Section 2 we define the expanding domain monodic fragment. In Section 3 we introduce the divided separated normal form (DSNF) for monodic temporal formulae and describe how monodic temporal formulae are translated into DSNF. In Sections 4 and 5 we introduce the fine-grained resolution calculus and provide completeness results for the fine-grained resolution calculus relative to the completeness of the general resolution calculus [7]. A number of examples will be given, showing how the fine-grained resolution calculus works in practice. In Section 7 we briefly describe how constant-domain problems can be handled,

through a translation of the formulae. Sections 8 and 9 are devoted to an implementation of fine-grained resolution and to its applications. Finally, in Section 10 we 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* $\Box$ ('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 the standard way [20,31]:

- Booleans **true** and **false** are FOTL-formulae;
- if P is an n-ary predicate symbol and t_i, $1 \leqslant i \leqslant n$, 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 $\Box\phi$, $\Diamond\phi$, $\bigcirc\phi$, $\phi\,\mathsf{U}\,\psi$, and $\phi\,\mathsf{W}\,\psi$.

A *literal* is an atomic formula or its negation.

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). A 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 $n \geqslant 0$, 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 $i, j \geqslant 0$, we have $I_i(c) = I_j(c)$.

A *(variable) assignment* $\mathfrak{a}$ is a function from the set of individual variables to $\cup_{n\in\mathbb{N}} D_n$. We denote the set of all assignments by $\mathfrak{A}$. The set of variable assignments $\mathfrak{A}_n$ corresponding to $\mathfrak{M}_n$ is a

subset of the set of all assignments, $\mathfrak{A}_n = \{\mathfrak{a} \in \mathfrak{A} \mid \mathfrak{a}(x) \in D_n \text{ for every variable } x\}$; clearly, $\mathfrak{A}_n \subseteq \mathfrak{A}_m$ 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* $\mathfrak{a} \in \mathfrak{A}_n$:

$\mathfrak{M}_n \models^{\mathfrak{a}}$ **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 $\mathfrak{M}_n \models^{\mathfrak{a}} ((\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi))$
$\mathfrak{M}_n \models^{\mathfrak{a}} \forall x\phi$	iff $\mathfrak{M}_n \models^{\mathfrak{b}} \phi$ for every assignment $\mathfrak{b}$ that may differ from $\mathfrak{a}$ only in x and such that $\mathfrak{b}(x) \in D_n$
$\mathfrak{M}_n \models^{\mathfrak{a}} \exists x\phi$	iff $\mathfrak{M}_n \models^{\mathfrak{b}} \phi$ for some assignment $\mathfrak{b}$ that may differ from $\mathfrak{a}$ only in x and such that $\mathfrak{b}(x) \in D_n$
$\mathfrak{M}_n \models^{\mathfrak{a}} \bigcirc\phi$	iff $\mathfrak{M}_{n+1} \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} \Diamond\phi$	iff there exists $m \geqslant n$ such that $\mathfrak{M}_m \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} \Box\phi$	iff for all $m \geqslant n$, $\mathfrak{M}_m \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} (\phi \mathsf{U} \psi)$	iff there exists $m \geqslant n$, such that $\mathfrak{M}_m \models^{\mathfrak{a}} \psi$ and for all $i \in \mathbb{N}, n \leqslant i < m$ 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

$$\forall x P(x) \wedge \Box(\forall x P(x) \Rightarrow \bigcirc \forall x P(x)) \wedge \Diamond \exists y \neg P(y) \tag{1}$$

is unsatisfiable over both expanding and constant domains; the formula

$$\forall x P(x) \wedge \Box(\forall x(P(x) \Rightarrow \bigcirc P(x))) \wedge \Diamond \exists y \neg P(y) \tag{2}$$

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

It is known [54] that satisfiability over expanding domains can be reduced to satisfiability over constant domains with a polynomial increase in the size of formulae.

The set of valid formulae of first-order temporal logic is not recursively enumerable. So, 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 [31,39]. However, the set of valid *monodic* formulae (see Definition 1 below) is known to be finitely axiomatisable [56].

Definition 1. An FOTL-formula ϕ is called *monodic* if, and only if, any subformula of the form $\mathcal{T}\psi$, where $\mathcal{T}$ is one of $\bigcirc, \Box, \Diamond$ (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

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

are monodic, whereas the formula

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

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 [56]. Moreover, although the two variable monodic fragment *without* equality is decidable [31], it was proved in [9] that the *two variable monadic monodic fragment with equality* is not even recursively enumerable. However, in [30] 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 classical *clause normal form* [42], before the inference rules of the calculus can be applied. Similarly, the temporal resolution 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 below.

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 atom and $M(x)$ is a unary literal. We call a clause of the first type an (original) *ground* step clause, and of the second type an (original) *non-ground* step clause. Note that the term 'original' is used here to distinguish these step clauses from other notions (derived, merged, etc. step clauses) that are introduced later.

Temporal step clauses are the key elements of the normal form, providing a description of how information is transferred from one temporal instant to the next.

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 finite set of arbitrary closed first-order formulae;
(2) the initial part, $\mathcal{I}$, is, again, given by a finite set of arbitrary closed first-order formulae;

(3) the step part, $\mathcal{S}$, is given by a finite 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 finite 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.

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 transformation 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 [21]; it consists of a sequence of steps.

(1) Transform a given monodic formula to negation normal form. (To assist understanding of the transformation, we list here some equivalences used in this step.)

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

If the transformations above are applied in a naive way, the size of the result may grow exponentially; we may have to use *renaming* [42,52] in order to keep the size of the transformed formula linear in the size of the original formula.

(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 $P(x)$ of the following form:

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

Since we are only interested in satisfiability, we use implications instead of equivalences renaming positive occurrences of subformulae, see also [42,52].

Formulae of the form (a) above 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 transformation by removing the temporal operators using their fixed point definitions (see [21]):
$\Box\forall x(P(x) \Rightarrow \Box\phi(x))$ 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 satisfiability equivalent 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 satisfiability equivalent 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 transformed into DSNF as follows:
$\Box\forall x(P(x) \Rightarrow \Diamond L(x))$ is satisfiability equivalent (see [7]) to

$$\Box\forall x((P(x) \wedge \neg L(x)) \Rightarrow \mathit{waitforL}(x)) \tag{3}$$

$$\Box\forall x(\mathit{waitforL}(x) \Rightarrow \bigcirc(\mathit{waitforL}(x) \vee L(x))) \tag{4}$$

$$\Box\forall x(\Diamond\neg\mathit{waitforL}(x)) \tag{5}$$

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

Theorem 1 (see [7], Theorem 1). *The transformation described above reduces any monodic first-order temporal formula ϕ to monodic temporal problem* P *in DSNF with at most linear increase in the size of the problem such that ϕ is satisfiable over constant and expanding domains if, and only if,* P *is satisfiable over constant and expanding domains, respectively.*

Example 3. 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. To reduce it to DSNF, we first, rename the innermost temporal subformula by a new predicate, P_1,

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

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

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

"unwind" the '$\Box$' operator, introducing P_4,

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

and reduce the subformula with the eventuality.

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

Finally, we rename the non-literal formula $P_2(x) \vee waitforP_2(x)$ on the right-hand side of $\Box\forall x[waitforP_2(x) \Rightarrow \bigcirc(waitforP_2(x) \vee P_2(x))]$ by $P_5(x)$ and add the definition $\Box\forall x[P_5(x) \Rightarrow (P_2(x) \vee waitforP_2(x))]$ for $P_5(x)$.

The conjuncts of the resulting formula form the following monodic temporal problem:

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

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

4. Monodic temporal resolution for the expanding domain case

A resolution method for the monodic fragment over expanding domains has been introduced in [7]. We sketch the monodic temporal resolution system presented in [7] here to make the paper self-contained. We slightly change and simplify the notions from [7] for the sake of presentation. We use this calculus from [7], which has been shown complete, to prove completeness of the fine-grained calculus introduced in the next section. That is, we prove completeness of the fine-grained calculus

relative to the completeness of the calculus in [7]. We assume expanding domains throughout this section.

Let P be a monodic temporal problem, and let

$$P_{i_1}(x) \Rightarrow \bigcirc M_{i_1}(x), \ldots, P_{i_k}(x) \Rightarrow \bigcirc M_{i_k}(x) \tag{6}$$

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

$$P_{i_j}(c) \Rightarrow \bigcirc M_{i_j}(c), \tag{7}$$

$$\exists x \bigwedge_{j=1}^{k} P_{i_j}(x) \Rightarrow \bigcirc \exists x \bigwedge_{j=1}^{k} M_{i_j}(x), \tag{8}$$

are called *derived* step clauses,[2] where $c \in const(\mathsf{P})$ and $j = 1 \ldots k$. Note that formulae of the form (7) and (8) are logical consequences of (6).

Let $\{\Phi_1 \Rightarrow \bigcirc \Psi_1, \ldots, \Phi_n \Rightarrow \bigcirc \Psi_n\}$ be a set of 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*.

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

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

is called a *full merged step clause*.

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

We now define the temporal resolution calculus, $\mathfrak{I}_e$, for the expanding domain case.

The inference rules of $\mathfrak{I}_e$ are the following.

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

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

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

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

[2] In [7] derived step clauses are termed e-derived step clauses.

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

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

where $\forall x(\mathcal{A}_i(x) \Rightarrow \bigcirc \mathcal{B}_i(x))$ are full merged step clauses such that for all $i \in \{1, \ldots, n\}$, 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.[3]

The set of full 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. $\mathcal{U}$:*

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

where $\mathcal{A}_i \Rightarrow \bigcirc \mathcal{B}_i$ are merged derived 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 $i \in \{1, \ldots, n\}$ are both valid. *Ground loop* and *ground loop formula* are defined similarly to the case above.

Let P be a temporal problem. By $\mathrm{TRes}(\mathsf{P})$ we denote the set of all possible conclusions of the inference rules above applied to P.

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 saturating the universal part of a given problem.

Definition 4 (Derivation). A *derivation* is a sequence of universal parts, $\mathcal{U} = \mathcal{U}_0 \subset \mathcal{U}_1 \subset \mathcal{U}_2 \subset \cdots$, extended by the conclusions of the inference rules such that $\mathcal{U}_{i+1}$ is obtained from $\mathcal{U}_i$ by applying an inference rule to $\mathcal{I}$, $\mathcal{S}$, $\mathcal{E}$, and $\mathcal{U}_i$ and adding its conclusion to $\mathcal{U}_i$ to obtain $\mathcal{U}_{i+1}$. The $\mathcal{I}$, $\mathcal{S}$ and $\mathcal{E}$ parts of the temporal problem are not changed during a derivation.

A derivation *terminates* if, and only if, either the contradiction is derived, in which case we say that the derivation *successfully terminates*, or if no new formulae can be derived by further inference steps. Note that since there exist only finitely many different full merged step clauses, the number of different conclusions of the inference rules of temporal resolution is finite. Therefore, every derivation is finite. If a (finite) derivation does not terminate, we call it *partial.* Any partial derivation can be continued yielding a terminating derivation.

A derivation $\mathcal{U} = \mathcal{U}_0 \subset \mathcal{U}_1 \subset \mathcal{U}_2 \subset \cdots \subset \mathcal{U}_n$ is called *fair* (we adopt terminology from [4]) if for any $i \geqslant 0$ and formula $\phi \in \mathrm{TRes}(\langle \mathcal{U}_i, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle)$, there exists $j \geqslant i$ such that $\phi \in \mathcal{U}_j$.

Note that all the inference rules have side conditions which are first-order problems. In general, these side conditions will therefore only be semi-decidable and in the case a side condition is false, it may happen that the test of this side condition does not terminate. So, to ensure fairness we must make sure that each such test cannot indefinitely block the investigation of alternative applications of inference rules in a derivation.

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

Note 1. In this section, we intentionally do not consider the classical concept of redundancy (see [4]) and deletion rules over sets of first-order formulae $\mathcal{U}_i$. We come to the issue of redundancy and deletion rules in Section 5 where we present a machine-oriented calculus.

Let $\mathsf{P} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$ be a monodic temporal problem, then the set of formulae

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

is termed the *constant flooded form* of P. (Strictly speaking, P^c is not in DSNF: we have to rename ground eventualities by propositions.) Evidently, P^c is satisfiability equivalent to P.

Theorem 2 (see [7, 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 P given by

$$\mathcal{I} = \{P(c)\}, \qquad \mathcal{S} = \{q \Rightarrow \bigcirc q\},$$
$$\mathcal{U} = \{q \equiv P(c)\}, \quad \mathcal{E} = \{\Diamond \neg P(x)\}.$$

The constant flooded form P^c is the extension of P by the eventuality clause

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

or, after renaming, an eventuality $\Diamond l$ and a universal clause $l \Rightarrow \neg P(c)$. Now, the step clause $q \Rightarrow \bigcirc q$ will be a loop in $\Diamond l$ and the eventuality resolution rule will derive a contradiction.[4]

5. Fine-grained resolution for the expanding domain case

The main drawback of the calculus introduced in the previous section is that the search for (full) merged 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 eventuality resolution rule is even more difficult.

In this section we formulate a clausal calculus, called fine-grained resolution, 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 derived step clause $\mathcal{A} \Rightarrow \bigcirc \mathcal{B}$. The rule can only be applied if $\mathcal{B} \cup \mathcal{U} \models \mathbf{false}$ and this fact can be established by a first-order resolution procedure (that would skolemise the universal part). However, a refutation of $\mathcal{B} \cup \mathcal{U}$ by resolution would also tell us which formulae in $\mathcal{B} \cup \mathcal{U}$ are

[4] Note that the non-ground eventuality $\Diamond \neg P(x)$ is not used. It was shown in [7] that if all step clauses are ground, for constant flooded problems we can ignore all non-ground eventualities.

involved in the derivation of a contradiction. Thus, not only can we check the side conditions for the rules of $\mathfrak{I}_e$ by means of first-order resolution, but also *search for clauses to merge* at the same time.

In contrast to $\mathfrak{I}_e$ which can be applied to monodic temporal problems $\langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$ where the universal part $\mathcal{U}$ and the initial part $\mathcal{I}$ are sets of arbitrary closed first-order formulae, fine-grained resolution operates on temporal problems where $\mathcal{U}$ and $\mathcal{I}$ are given by sets of first-order clauses.

Definition 5. Let P be a monodic temporal problem. We define a set $\mathbf{S}(\mathsf{P})$ of initial, universal, and step clauses, called *the result of clausification* P, as follows.

(1) Step clauses from P are in $\mathbf{S}(\mathsf{P})$.
(2) 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 $\mathbf{S}(\mathsf{P})$.
(3) 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.

Note that fine-grained resolution operates *typed clauses* (every clause is marked as "initial," "universal," or "step"). In contrast to derivations by $\mathfrak{I}_e$ which proceed by extending the universal part of a temporal problem while keeping all other parts of a temporal problem constant, fine-grained resolution might not only derive new universal clauses, but also additional initial clauses and step clauses of the form

$$C \Rightarrow \bigcirc D, \tag{10}$$

where 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.

We consider initial and universal clauses as literal multisets, and step clauses of the form (10) as ordered pairs of literal multisets. We assume basic knowledge of classical first-order resolution (see, for example, [4,5,37]).

Fine-grained resolution differs from the calculus $\mathfrak{I}_e$ in two aspects: First, instead of the step resolution and the initial termination rule of $\mathfrak{I}_e$, we use *fine-grained step resolution*, a set of deduction and deletion rules operating on clausified problems; second, we use a particular algorithm, called FG-BFS, to determine the loops to which we apply the ground and non-ground eventuality resolution rule of $\mathfrak{I}_e$.

Let us first define the *deduction* rules of fine-grained step resolution. In the following, we assume that different premises and conclusions of the deduction rules have no variables in common; variables may be renamed if necessary.

- *First-order resolution between two universal clauses and factoring on a universal clause*. The result is a universal clause.
- *First-order resolution between an initial and a universal clause, between two initial clauses, and factoring on an initial clause*. The result is an initial clause.
- *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 a 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*.[5]

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

where $C_1 \Rightarrow \bigcirc(D_1 \vee L)$ is a step clause, $D_2 \vee \neg N$ is a universal clause, and σ is a 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.*

- *Right factor.*

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

where σ is a 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.*

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

A (linear) *proof* by fine-grained resolution of a clause C from a set of clauses $\mathbf{S}$ is a sequence of clauses $C_1, \ldots, C_m$ 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 applied to clauses from premises $C_1, \ldots, C_{i-1}$. A proof of **false** is called a *refutation.*

Example 5. Fine-grained step resolution without the restriction on substitutions would, certainly, lead to unsoundness: Consider the monodic problem given by

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

which is satisfiable. After skolemisation, $\mathcal{U}^s = \{us1 : \neg Q(c),\ us2 : P(x) \vee Q(x)\}$. Unrestricted resolution would derive $us3 : \neg P(c)$ from $us1$ and $s1$, and then a contradiction from $us1$, $us2$, and $us3$.

[5] This restriction justifies skolemisation of the universal part: Skolem constants from one moment of time do not propagate to the previous moment.

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

The restriction that a most general unifier σ applied to step clauses $C \Rightarrow \bigcirc D$ does not map a variable in C into a Skolem constant is violated since σ maps the variable x in $s1$ to the constant c. Thus, the restriction we impose on step resolution inferences prevents the derivation of the clause $us3$ and, therefore, prevents the derivation of a contradiction.

Example 6. 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

$$\mathcal{I} = \{\forall x P(x)\}, \qquad \mathcal{U} = \{\neg Q(c)\},$$
$$\mathcal{S} = \{P(x) \Rightarrow \bigcirc Q(x)\}, \quad \mathcal{E} = \emptyset.$$

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

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

Then, a contradiction can be easily derived from the additional, universal, and initial clauses.

We will now prove some basic results which will help us to establish the completeness of our calculus.

Definition 6. A clause of the form $C \Rightarrow \bigcirc \mathbf{false}$, where C is of the same form as in (10), is called *a final clause*.

Lemma 3. *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 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 derived step-clause of the form (9) added by clausification (w.l.o.g., we assume that all the variables $x_1, \ldots, x_K$ are pairwise distinct). We assume that Δ is *tree-like*, that is, no clause in Δ is used more than once as an premise 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 identity substitutions.[7] That is, there exist substitutions $\{\sigma_{i,j} \mid i = 1 \ldots K, j = 1 \ldots s_i\}$ such that

[7] 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*.

$$\begin{array}{l}\{P_i(x_i)\sigma_{i,j} \Rightarrow \bigcirc M_i(x_i)\sigma_{i,j} \mid i = 1 \ldots K, j = 1 \ldots s_i\},\\ \quad \{p_i \Rightarrow \bigcirc l_i \mid i = 1 \ldots L\}\end{array} \tag{11}$$

(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 identity substitutions, 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 $\sigma_{i,j}$. We introduce an equivalence relation Σ on the clauses from (11) as follows: For every i, j, i', j' we have

$$\left(P_i(x_i)\sigma_{i,j} \Rightarrow \bigcirc M_i(x_i)\sigma_{i,j},\ P_{i'}(x_{i'})\sigma_{i',j'} \Rightarrow \bigcirc M_{i'}(x_{i'})\sigma_{i',j'}\right) \in \Sigma$$

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 (11) by Σ; let $\mathcal{I}_k$ be the set of indexes of the kth equivalence class (we refer to clauses from (11) by indexes of the corresponding substitutions).

Let $C_k = \bigwedge_{(i,j)\in\mathcal{I}_k} P_i(x_i)\sigma_{i,j}$, for every k, $1 \leqslant k \leqslant N$; let $C_0 = \bigwedge_{i=1}^{L} p_i$. Note that $C = \bigwedge_{k=1}^{N} C_k \wedge C_0$, and that the C_k are pairwise disjoint. Let $D_k = \bigwedge_{(i,j)\in\mathcal{I}_k} M_i(x_i)\sigma_{i,j}$, 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 derived step clause. □

Lemma 4. *Let* $\mathsf{P} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E}\rangle$ *be a monodic temporal problem and* $\mathbf{S} = \mathbf{S}(\mathsf{P})$ *be the result of clausification* P. *Let* $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ *be a merged derived step clause such that* $\mathcal{B} \cup \mathcal{U} \models \mathbf{false}$. *Then there exists a final clause* $C \Rightarrow \bigcirc\mathbf{false}$, *proved 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 proved 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 (8):

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

and original ground step clauses and derived clauses of the form (7):

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

Again, as in the proof of Lemma 3, $p_i \Rightarrow \bigcirc l_i$ denotes either an original ground step clause or a clause of the form (7). (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 $\mathcal{B} \cup \mathcal{U} \models \mathbf{false}$, we have $\bigwedge_{i=1}^{K} \bigwedge_{j=1}^{s_i} M_{i_j}(c_i) \wedge \bigwedge_{i=1}^{L} l_i \wedge \mathcal{U} \models \mathbf{false}$ where $c_1, \ldots, c_K$ are fresh (Skolem) constants.

Consider now a set of instances of step clauses

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

Let Δ be a (first-order) refutation of $\mathcal{U}$ 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

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

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

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

By the lifting theorem for first-order resolution (cf., e.g. [37, p. 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 $\Delta' \leqslant_s \Delta$ in the terminology of [37], 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.

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 $\Gamma' \leqslant_s \Gamma$, 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 a 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 5. *Let* $\mathsf{P} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$ *be a monodic temporal problem and* $\mathbf{S} = \mathbf{S}(\mathsf{P})$ *be the result of clausification. Let* $C \Rightarrow \bigcirc \mathbf{false}$ *be an arbitrary final clause proved by fine-grained step resolution from* $\mathbf{S}$ *. Then there exists a derivation* $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \cdots$ *by the step resolution rule of* $\mathfrak{I}_e$ *and a merged derived step clause* $\mathcal{A} \Rightarrow \bigcirc \mathcal{B}$ *such that* $\mathcal{B} \cup \mathcal{U}_i \models \mathbf{false}$, *for some* $i \geqslant 0$, *and* $\mathcal{A} = \widetilde{\exists} C$.

Proof. Since $C \Rightarrow \bigcirc \mathbf{false}$ is provable, 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 3.

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 $D \Rightarrow \bigcirc \mathbf{false}$. Thus, the proof Γ is of the form: $\Delta, D \Rightarrow \bigcirc \mathbf{false}, \neg D, \Delta'$, where Δ does not contain applications of the clause conversion rule, and Δ' contains $n-1$ application of this rule. By Lemma 3, there exists a merged derived step clause $\mathcal{A}' \Rightarrow \bigcirc \mathcal{B}'$ such that $\mathcal{B}' \cup \mathcal{U} \models \mathbf{false}$ and $\mathcal{A}' = \widetilde{\exists} D$. Let the temporal problem P' be $\langle \mathcal{U} \cup \{\neg D\}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$. Now, it is possible to repeat the subproof Δ by fine-grained resolution from $\mathbf{S}(\mathsf{P}')$, the clause $\neg D$ is already in $\mathbf{S}(\mathsf{P}')$, so the subproof Δ' can also be repeated. Therefore, 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. Then the Lemma follows from the induction hypothesis. □

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

Proof. The lemma easily follows from Lemma 4 by induction on i. □

Lemma 5 ensures the soundness of fine-grained step resolution. Lemma 6 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 7. *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 constant flooded monodic temporal problems over expanding domains.*

Proof. Soundness follows from Lemma 5.

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 $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \mathcal{U}_2 \subseteq \cdots \subseteq \mathcal{U}_n$. 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 6, there exists a final clause $C_k \Rightarrow \bigcirc \mathbf{false}$ proved 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 $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \mathcal{U}_2 \subseteq \cdots \subseteq \mathcal{U}_n$ is still a correct terminating derivation in $\mathfrak{I}_e$. □

So far, we have not discussed *redundancy elimination* and various *refinements* (e.g., ordering refinements) for fine-grained resolution. However, no modern automated proof-search procedure would be successful without these computer-oriented techniques [4]. We extend fine-grained resolution

with *deletion rules*, which eliminate redundant clauses during proof search, and discuss briefly the possibility of refinements here.

- *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).
- *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 $\neg C_1$.
  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 and subsumed clauses are deleted.

We have to consider now derivations over *sets* of clauses. We adopt the terminology from [4,41]. If a clause C is a tautology or is subsumed by a clause from a set of clauses $\mathbf{S}$, we say that C is *redundant* with respect to $\mathbf{S}$. A (theorem proving) *derivation* by fine-grained resolution is a sequence of sets of clauses $\mathbf{S_0} \rhd \mathbf{S_1} \rhd \cdots$ such that every $\mathbf{S_{i+1}}$ differs from $\mathbf{S_i}$ by either adding the conclusion of a deduction rule (that is, a deduction rule of fine-grained step resolution or the ground or non-ground eventuality resolution 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. A clause C is called *persistent* in a derivation $\mathbf{S_0} \rhd \mathbf{S_1} \rhd \cdots$ if there exists $j \geqslant 0$ such that for every $k \geqslant j$ we have $C \in \mathbf{S_k}$. A theorem proving derivation $\mathbf{S_0} \rhd \mathbf{S_1} \rhd \cdots$ is *fair* if for every clause C that can be obtained by a deduction rule from non-redundant persistent clauses of the derivation, either $C \in \bigcup_{j\geqslant 0} \mathbf{S_j}$ or C is redundant w.r.t. $\bigcup_{j\geqslant 0} \mathbf{S_i}$, that is, in a fair derivation no inference from non-redundant persistent clauses is postponed indefinitely.

Note that the proof of completeness given above (in particular, the proof of Lemma 4) may not work in the presence of deletion rules. To show that any fair theorem derivation by fine-grained resolution from the clausification of an unsatisfiable temporal problem will eventually produce a clause set $\mathbf{S_i}$ containing a contradiction, we consider *constrained calculi*, that is, resolution over constrained clauses with constraint inheritance [40]. It is known that such inference systems are complete and, moreover, compatible with practical redundancy elimination rules [41]. Here, we take into account that there are no clauses with equality, and therefore all sets are *well-constrained* in the terminology of [41].

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

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

Then all clauses originating from the universal part have empty constraints and all step 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 4.

Then the machinery from [40] can be used to prove completeness of fine-grained resolution with redundancy elimination and refinements.

Theorem 8. *If a constant flooded monodic temporal problem* P *is unsatisfiable, then any fair theorem proving derivation by fine-grained resolution from* $\mathbf{S}(\mathsf{P})$ *will derive a contradiction.*

6. Loop search

In this section we show that fine-grained step resolution can also be used to find the appropriate set of full merged clauses to which we need to apply the ground or non-ground eventuality resolution rule. Our method is based on the breadth-first search (BFS) algorithm presented in [7]; 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 [10,11].) As in [7], for simplicity, we consider non-ground eventualities only. The algorithm and the proof of its properties for the ground case can be obtained by considering merged derived step clauses instead of the general case and by deleting the parameter "x" and quantifiers.

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

Theorem 9. *Let* P *be a temporal problem and* $\Diamond L(x) \in \mathcal{E}$. *Then 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* $L(x)$.

In addition, 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 merged clauses of the form $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc \mathcal{B}(x))$ such that the formula

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

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 $j \in \{1 \dots k\}$, $\forall x(\mathcal{U} \wedge \mathcal{B}_j^{(i+1)}(x) \Rightarrow (\neg L(x) \wedge H_i(x)))$ 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.

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 merged step clauses).

Let $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc\mathcal{B}(x))$ be a full 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. This observation suggest searching for such full merged step clauses with the help of fine-grained resolution as we did to search for merged derived step clauses in the previous section.

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

$$P(c^l) \Rightarrow \bigcirc M(c^l). \tag{12}$$

We add the clause[8] $\mathbf{true} \Rightarrow \bigcirc\neg\Phi(c^l)$ 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 deduction 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 proof. Then there exists a full 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* $\mathcal{A}(x) = (\widetilde{\exists}C)\{c^l \mapsto x\}$.

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

Proof. By Lemma 5, there exists a merged (from $LT(S)$) 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 merged (from* $\mathbf{S}$*) step clause such that* $\forall x(\mathcal{B}(x) \wedge \mathcal{U} \Rightarrow \Phi(x))$. *Then there 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 Lemma 4. As we have already seen, $\exists x(\mathcal{B}(x) \wedge \mathcal{U} \wedge \neg\Phi(x))$ is unsatisfiable, and this can be checked by a first-order resolution procedure. Since the constant 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 [7] taking into consideration Lemmas 10 and 11 and arguments similar to those given in the proof of Theorem 7.

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 2. Termination of all calls by step resolution can be achieved for the cases when there exists a (first-order) resolution decision procedure [17] for formulae in the universal part. In this case, the class of corresponding monodic formulae is decidable by temporal resolution. This is discussed in more detail in [7].

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 in the saturation of $\mathbf{S_{i+1}}$.

(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.

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

Example 7. Let us consider a monodic temporal problem P given by

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

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

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

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

Since the original problem does not contain any constants (the only constant c is introduced by skolemisation), constant flooding does not introduce additional step clauses.

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

  $s2 : A(x) \Rightarrow \bigcirc A(x)$ $\quad (s1, u1)$
  $s3 : A(x) \Rightarrow \bigcirc \neg L(x)$ $\quad (s1, u2)$

  With this additional clauses, the set of clauses is saturated. Now we try to find a loop in $\Diamond L(x)$.
- Loop search for $\Diamond L(x)$
  The input to the BFS algorithm is the set $\mathbf{S} = \{u1, u2, u3, s1, s2, s3\}$ and the eventuality $\Diamond L(x)$. In step (1) of the algorithm, we set $H_0(x) = \mathbf{true}$; $N_0 = \emptyset$; and $i = 0$. In step (2), we first compute $\mathrm{LT}(\mathbf{S})$ which is given by $\{lt1 : A(c^l) \Rightarrow \bigcirc B(c^l)\}$.
  From $\mathbf{S_1} = \mathrm{LT}(\mathbf{S}) \cup \{\mathbf{true} \Rightarrow \bigcirc L(c^l)\}$ we deduce the following clauses by fine-grained step resolution (except the clause conversion rule):

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

  The set of clauses is saturated. The set N_1 of all new final clauses is $\{A(c^l) \Rightarrow \bigcirc \mathbf{false}\}$. Since N_1 is not empty, in step (3) we set $H_1(x)$ to $A(x)$. Obviously, $\forall x(H_0(x) \Rightarrow H_1(x))$ is not true, so we go back to step (2) of the algorithm.

Now the set $\mathbf{S_2} = \mathrm{LT(S)} \cup \{l6 : \mathbf{true} \Rightarrow \bigcirc(\neg A(c^l) \vee L(c^l))\}$ and we deduce from it the following clauses:

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

The set of clauses is saturated. The set N_2 of all new final clauses is $\{A(c^l) \Rightarrow \bigcirc \mathbf{false}\}$. Since N_2 is not empty, in step (3) we obtain $H_2(x) = A(x)$.
As $\forall x(H_1(x) \Rightarrow H_2(x))$, the algorithm terminates in step (4) and returns the loop $A(x)$.

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

  $u4 : \neg A(x)$.

- Step resolution
  Since this conclusion extends the set of universal clauses, additional inferences by step resolution may now be possible. Indeed, we are able to derive

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

  using $u4$.

  Since the set of clauses has been extended, it is also possible that additional loops can be found. However, instead of searching for a loop for $\Diamond L(x)$ again, we now focus on the eventuality $\Diamond l$.
- Loop search for $\Diamond l$
  The input to the FG-BFS is $\mathbf{S} = \{u1, u2, u3, u4, u5, s1, s2, s3\}$. In step (1) of the algorithm, we set $H_0(x) = \mathbf{true}$; $N_0 = \emptyset$; and $i = 0$. In step (2), we first compute $\mathrm{LT}(\mathbf{S})$, which is given by $\{lt1 : A(c^l) \Rightarrow \bigcirc B(c^l)\}$. From $\mathbf{S_1} = \mathrm{LT}(\mathbf{S}) \cup \{l16 : \mathbf{true} \Rightarrow \bigcirc l\}$ we can deduce:

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

  that is, a contradiction. Therefore, the loop is **true**.
- Eventuality resolution
  We can apply now the eventuality resolution rule whose conclusion is $\neg\mathbf{true}$.
  This means that we have derived a contradiction, and the problem is unsatisfiable.

Note 3. As the example above shows, the presence of clauses of the form (9), introduced by clausification, and (12), 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 the original constants and the loop constant to be substituted for 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 transformation 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 $\mathbf{S_1} = \mathbf{S} \cup \{l1 : \mathbf{true} \Rightarrow \bigcirc L(c^l)\}$:

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

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

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

7. Constant domains

A resolution method for the monodic fragment over constant domains has also been introduced in [7]. The only difference between the calculus $\mathfrak{I}_e$ of Section 4 aimed at the expanding domain case and the calculus $\mathfrak{I}_c$, which we sketch here, aimed at the constant-domain case is in the way how *derived step clauses* are defined.

Let P be a monodic temporal problem, and let

$$P_{i_1}(x) \Rightarrow \bigcirc M_{i_1}(x), \ldots, P_{i_k}(x) \Rightarrow \bigcirc M_{i_k}(x) \tag{13}$$

be a subset of the set of its step clauses. For the constant-domain case, derived step clauses are defined as formulae of the form

$$P_{i_j}(c) \Rightarrow \bigcirc M_{i_j}(c), \tag{14}$$
$$\exists x \bigwedge\nolimits_{j=1}^{k} P_{i_j}(x) \Rightarrow \bigcirc \exists x \bigwedge\nolimits_{j=1}^{k} M_{i_j}(x), \tag{15}$$
$$\forall x \bigvee\nolimits_{j=1}^{k} P_{i_j}(x) \Rightarrow \bigcirc \forall x \bigvee\nolimits_{j=1}^{k} M_{i_j}(x), \tag{16}$$

where $c \in const(\mathsf{P})$ and $j = 1 \ldots k$. Formulae of the form (16), as well as formulae of the form (14) and (15), are logical consequences of (13) in the *constant-domain* case, while (16) is not a logical consequence of (13) in the expanding domain case as Example 1 shows.

Other notions (merged derived and full merged step clauses) and the inference system are defined in exactly the same way as those for the calculus $\mathfrak{I}_e$ in Section 4 based on the new definition of a derived step clause. It is shown in [7] that the calculus $\mathfrak{I}_c$ is sound and complete for monodic temporal problems over constant domains.

Note that if derived step clauses of the form (16) participate in merging, the conclusion of the step resolution rule contains an existentially quantified formula (the negation of the universally quantified left-hand side of the derived step clause). Clausifying such formulae would require extending the signature; therefore, the machinery of Section 5 does not work for the constant-domain case, and we cannot construct a fine-grained resolution calculus for the constant-domain case. Instead, 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 $\mathrm{Exp}(\mathsf{P})$ is defined as follows. The universal, initial, and eventuality parts of $\mathrm{Exp}(\mathsf{P})$ are that of P; the step part consists of all step clauses from $\mathcal{S}$ *together with* all derived step clauses of the form (16). More precisely, in order to fit the definition of a temporal problem, we have to rename complex expressions in the derived step clauses.

Proposition 14. *A temporal problem* P *is satisfiable over constant domains if, and only if, the temporal problem* $\mathrm{Exp}(\mathsf{P})$ *is satisfiable over constant domains.*

Proof. If $\mathsf{P} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$ and $\mathrm{Exp}(\mathsf{P}) = \langle \mathcal{U}', \mathcal{I}, \mathcal{S}', \mathcal{E} \rangle$ then $\mathcal{U} \subseteq \mathcal{U}'$ and $\mathcal{S} \subseteq \mathcal{S}'$. So, if $\mathrm{Exp}(\mathsf{P})$ is satisfiable then P is satisfiable. On the other hand, over constant domains, derived step clauses of the form (16) are logical consequences of the step part $\mathcal{S}$, which means that over constant domains P implies $\mathrm{Exp}(\mathsf{P})$. So, if P is satisfiable then $\mathrm{Exp}(\mathsf{P})$ is satisfiable. □

Proposition 15. *For any temporal problem* P, *the temporal problem* $\mathrm{Exp}(\mathsf{P})$ *is satisfiable over constant domains if, and only if, the temporal problem* $\mathrm{Exp}(\mathsf{P})$ *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 8. The following temporal problem P, which is unsatisfiable over constant domains,

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

$$\mathcal{S} = \left\{ \begin{array}{l} Q(x) \Rightarrow \bigcirc Q(x) \\ P(x) \Rightarrow \bigcirc R(x) \end{array} \right\}, \qquad \mathcal{E} = \{\Diamond \neg L(x)\}$$

has the following expanding domain model. For every $i \geqslant 1$, let 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 of Section 5 would not be able to deduce a contradiction from P; however, the temporal problem $\mathrm{Exp}(\mathsf{P})$ given by

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

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

is 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. Implementation

The fine-grained resolution calculus described in Section 5 has been implemented in the theorem prover **TeMP** [34], which we describe in more detail in this section.

As input **TeMP** takes the clausified form $\mathbf{S} = \mathbf{S}(\mathsf{P})$ of a constant-flooded monodic temporal problem $\mathsf{P} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$ plus the set $\mathcal{E}$ of eventuality clauses. The clauses in $\mathbf{S}$ will be called *DSNF clauses*.

The main procedure of **TeMP** consists of a loop where at each iteration (i) the set of clauses is saturated under application of the rules of fine-grained step resolution, and then (ii) for every eventuality clause in the clause set, an attempt is made to find a set of premises for an application of the eventuality resolution rules. If we find such a set, the set of clauses representing the conclusion of the application is added to the current set of clauses. The main loop terminates if the empty clause is derived, indicating that the initial set of clauses is unsatisfiable, or if no new clauses have been derived during the last iteration of the main loop, which in the absence of the empty clause indicates that the initial set of clauses is satisfiable.

The implementation takes advantage of the fact that the deduction and deletion rules of fine-grained resolution are close enough to classical first-order resolution to allow us to use a first-order resolution prover to provide the basis for the implementation of these rules. This approach requires that we have to transform the given set $\mathbf{S}$ of DSNF clauses into a set of first-order clauses.

Let P be a temporal problem and $\mathbf{S} = \mathbf{S}(\mathsf{P})$ be the result of clausification. For every k-ary predicate, P, occurring in $\mathbf{S}$, we introduce a new $(k+1)$-ary predicate $\widetilde{P}$. We will also use the constant 0 (representing the initial moment in time), and unary function symbols s (representing the successor function on time) and h, which we assume do not occur in $\mathbf{S}$. (The function h is used to ensure the restriction on the unifiers as we explain in more detail later.) Let ϕ be a first-order formula in the vocabulary of $\mathbf{S}$. We denote by $[\phi]^T$ the result of replacing all occurrences of predicates in ϕ by their "tilded" counterparts with T as the first argument (e.g., $P(x, y)$ is replaced with $\widetilde{P}(T, x, y)$).

The term T will either be the constant 0 or the variable t (intuitively, t represents a moment in time). The variable t is assumed to be universally quantified.

In the implementation we make use of two different translations FO and FO_{BFS} of $\mathbf{S}$ to a set of first-order clauses. Let us first define $\mathrm{FO}(\mathbf{S})$:

- For every initial clause C in $\mathbf{S}$, the clause $[C]^0$ is in $\mathrm{FO}(\mathbf{S})$.
- For every universal clause D in $\mathbf{S}$, the clause $[D]^t$ is in $\mathrm{FO}(\mathbf{S})$.
- For step clauses we have the following subcases:
  - For every ground step clause $p \Rightarrow \bigcirc l$ in $\mathbf{S}$, the clause $\neg\widetilde{p}(t) \vee \widetilde{l}(s(t))$ is in $\mathrm{FO}(\mathbf{S})$;
  - For every non-ground step clause $P(x) \Rightarrow \bigcirc M(x)$ in $\mathbf{S}$, the clause $\neg\widetilde{P}(t,x) \vee \widetilde{M}(s(t),h(x))$ is in $\mathrm{FO}(\mathbf{S})$;
  - For every step clause $P(c) \Rightarrow \bigcirc M(c)$ of the form (9) in $\mathbf{S}$, the clause $\neg\widetilde{P}(t,c) \vee \widetilde{M}(s(t),c)$ is in $\mathrm{FO}(\mathbf{S})$.

Note that FO does not translate the eventuality clauses in $\mathcal{E}$ and that these clauses are not included in $\mathrm{FO}(\mathbf{S})$. It can be shown that for a monodic temporal problem $\mathsf{P} = \langle \mathcal{U}, \mathcal{I}, \mathcal{S}, \mathcal{E} \rangle$ with corresponding clausified form $\mathbf{S} = \mathbf{S}(\mathsf{P})$, the temporal formula $\mathcal{I} \wedge \Box\mathcal{U} \wedge \Box\forall x\mathcal{S}$ is satisfiable over expanding domains if, and only if, $\mathrm{FO}(\mathbf{S})$ is first-order satisfiable. Furthermore, if $\mathrm{FO}(\mathbf{S})$ is first-order unsatisfiable, so is P. However, if $\mathcal{E}$ is a non-empty set of eventuality clauses and $\mathrm{FO}(\mathbf{S})$ is satisfiable, we are not able to draw any conclusion about the satisfiability of P.

It is not the aim of FO to reduce temporal satisfiability over expanding domains to first-order satisfiability. Instead the key insight motivating the translation FO is that inferences by the deduction and deletion rules defined in Section 5 on $\mathbf{S}$, including (implicitly) the clause conversion rule, can be realised using classical ordered first-order resolution with selection (see, e.g. [4]) on $\mathrm{FO}(\mathbf{S})$. For rules first-order resolution on universal and initial clauses (see Section 5), this is obvious. Step resolution restricts inferences on step clauses to literals below a '$\bigcirc$' operator. In analogy, if a clause in $\mathrm{FO}(\mathbf{S})$ contains a *next-state* literal, that is, a literal whose first argument starts with the function symbol s, a resolution inference should only be performed on such a literal. This requirement can be enforced by an appropriate literal selection strategy. Right factoring and left factoring correspond to standard factoring inferences on first-order clauses. Note that all rules performing inferences on step clauses impose the restriction on unifiers σ that σ does not map variables occurring in the left side of a step clause into a constant or a functional term. On first-order clauses, this restriction is enforced by the function symbol h introduced by FO. Each temporal literal $\bigcirc M(x)$ is mapped by FO to $\widetilde{M}(s(t),h(x))$. The function symbol h "shields" the variable x from being instantiated by a constant or functional term. Finally, clause conversion is implicit on the first-order level: If the conclusion of an inference step involving a clause C containing next-state literals results in clause D without any next-state literals, then D corresponds to the translation of a universal clause. The standard redundancy elimination mechanisms for ordered first-order resolution, such as subsumption and tautology deletion, correspond to deletion rules for temporal clauses that we have defined in Section 5.

So, in each iteration of the main loop of **TeMP**, step (i) can be achieved with the help of the translation FO and a first-order theorem prover.

As for step (ii) in the main loop of **TeMP**, the system implements the FG-BFS algorithm described in Section 5. In step (2) of the FG-BFS algorithm, the rules of fine-grained step resolution

are applied with the exception of the clause conversion rule. To enforce this restriction, we use a variant FO_{BFS} of FO. Let $\mathbf{S_{i+1}}$ be a monodic temporal problem in clausified form as defined in step (2) of the FG-BFS algorithm. Then $\mathrm{FO}_{BFS}(\mathbf{S_{i+1}})$ is defined as follows:

- For every universal clause D in $\mathbf{S_{i+1}}$, the clause $[D]^t$ is in $\mathrm{FO}_{BFS}(\mathbf{S_{i+1}})$.
- For every ground step clause $p \Rightarrow \bigcirc l$ in $\mathbf{S_{i+1}}$, the clause $\neg\widetilde{p}(0) \vee \widetilde{l}(s(t))$ is in $\mathrm{FO}_{BFS}(\mathbf{S_{i+1}})$, and for every non-ground step clause $P(x) \Rightarrow \bigcirc M(x)$ in $\mathbf{S_{i+1}}$, the clause $\neg\widetilde{P}(0,x) \vee \widetilde{M}(s(t),h(x))$ is in $\mathrm{FO}_{BFS}(\mathbf{S_{i+1}})$.

Recall that initial clauses do not contribute to loop search, so we should not include their translation into $\mathrm{FO}_{BFS}(\mathbf{S_{i+1}})$. Again, the motivation for FO_{BFS} is that saturation of $\mathbf{S_{i+1}}$ under the rules of fine-grained step resolution except the clause conversion rule corresponds to the saturation of $\mathrm{FO}_{BFS}(\mathbf{S_{i+1}})$ under ordered first-order resolution as described above. In particular, clauses consisting only of literals whose first argument is 0 in the saturation of $\mathrm{FO}_{BFS}(\mathbf{S_{i+1}})$ correspond to final clauses (up to negation). Using this criterion it is straightforward to extract those clauses from the saturation of $\mathrm{FO}_{BFS}(\mathbf{S_{i+1}})$ to form the set N_{i+1} which is the outcome of step (2) of the FG-BFS algorithm and to proceed with step (3).

Finally, note that it is straightforward to see whether a clause in $\mathrm{FO}(\mathbf{S})$ is the result of translating an initial, a universal, or a (non-)ground step clause. This makes it possible to compute $\mathrm{FO}_{BFS}(\mathbf{S})$ from $\mathrm{FO}(\mathbf{S})$ instead of from $\mathbf{S}$. Also, the conclusion of an application of one of the eventuality resolution rules can directly be computed as a set of first-order clauses of the appropriate form. Thus, there is no need to ever translate clauses in $\mathrm{FO}(\mathbf{S})$ back to DSNF clauses. Instead, after translating the input given to **TeMP** once using FO we can continue to operate with first-order clauses.

The task of saturating clause sets with ordered first-order resolution simulating step resolution is delegated to the kernel of the first-order resolution prover **Vampire** [46], which is linked to the whole system as a C++ library. Minor adjustments have been made in the functionality of **Vampire** to accommodate step resolution: a special mode for literal selection has been introduced such that in a clause containing a next-state literal only next-state literals can be selected. At the moment, the result of a previous saturation step, augmented with the result of an application of the eventuality resolution rules, is resubmitted to the **Vampire** kernel, although no non-redundant inferences are performed between the clauses from the already saturated part. This is only a temporary solution, and in the future **Vampire** will support incremental input in order to reduce communication overhead.

To illustrate the effectiveness of this approach, we have applied **TeMP** to each of the examples of monodic temporal problems in this paper. The results are given in Fig. 3. In this table, 'Time' is the time required by **TeMP** to solve the problem, measured in CPU seconds; 'BRes' is the number of clauses generated by ordered first-order resolution using **Vampire** ; 'SuccEv' is the number of times the algorithm FG-BFS was executed and successfully computed a loop; 'NREv' is the number of times the algorithm FG-BFS was executed, successfully computed a loop, and the application of eventuality rules to this loop results in non-redundant conclusion; 'FailEv' is the number of times the algorithm FG-BFS was executed but failed to find a loop. As can be seen from the results, none of the examples poses a serious problem for **TeMP**. Due to the lack of any other system able

Example	Result	Time	BRes	SuccEv	NREv	FailEv
1(1)	Unsatisfiable	0.010	17	1	1	0
1(2)	Satisfiable	0.000	3	0	0	1
4	Unsatisfiable	0.020	20	1	1	1
5	Satisfiable	0.000	3	0	0	0
6	Unsatisfiable	0.000	2	0	0	0
7	Unsatisfiable	0.020	51	2	2	1
8	Unsatisfiable	0.030	82	1	1	0

Fig. 3. Performance of **TeMP** on sample monodic temporal problems.

to solve problems in monodic first-order temporal logic, we cannot compare the performance of **TeMP** to that of other provers. However, in [34] we have compared **TeMP** to decision procedures for propositional linear time temporal logic with quite positive results.

9. Applications and case studies

We have applied **TeMP** to problems from several domains, in particular, to examples specified in the temporal logics of knowledge (the fusion of propositional linear-time temporal logic with multi-modal S5) and to verification problems.

Whilst problems formalised in the temporal logic of knowledge can be solved using proof methods for combined modal and propositional temporal logics [13], we translate them into monodic first-order temporal logic and apply **TeMP** to the result of transformation providing thus a *practical* possibility of solving those problems. Translations from the temporal logics of knowledge into FOTL are given in [26]; we also incorporate ideas from [47] to translate some of the modal logic axioms in a way suitable for **TeMP**. Moreover, the transformation maps formulae in the temporal logics of knowledge into a subclass of the monodic fragment which is decidable by temporal resolution (see Note 2).

A specification of the game Cluedo (a deduction game where reasoning about knowledge is essential) in temporal logics of knowledge is given in [12], for a particular play of the game. At points in the game, certain properties can be proved, for example that one player knows another player holds a particular card. Both the specification and properties to be proved have been translated into monodic first-order temporal logic and the proof carried out using **TeMP**.

A temporalised version of the well known muddy children problem (see, for example [16]) specified in temporal logics of knowledge has been translated to monodic first-order temporal logic and relevant properties proved using **TeMP**. A similar approach can be taken for the proof of properties of security protocols specified in temporal logics of knowledge (see, for example [14]).

Work within the Liverpool Verification Laboratory[9] has utilised **TeMP** in formal verification. In particular, in a collaboration with Motorola, **TeMP** was used to support the verification of software designs. Recent work within the laboratory has involved identifying a class of *Abstract State Machines* [29] that can be translated into a monodic fragment of first-order temporal logic suitable for input to **TeMP**. This allows **TeMP** to be used as the basis for verification of a range of software/hardware designs (given using Abstract State Machines) [24]. Recent work has examined the verification of *infinite state systems* (something that is impossible using traditional model-checking) and has shown how, for a significant class of systems, such verification problems can be translated into a monodic formula suitable for input to **TeMP** [22,23]. We successfully used **TeMP** for fully automatic verification of cache coherence protocols [22].

10. Conclusion

We have here described a fine-grained resolution calculus for monodic first-order temporal logic over expanding domains. Soundness of the fine-grained resolution calculus is easy to prove and completeness is shown relative to the completeness proof for the expanding domain for the non-fine-grained version [7]. 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 logic. This makes the new calculus presented here particularly amenable to efficient implementation. In 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: improving the implementation and application to a wider range of real world examples.

Acknowledgments

Finally, we thank the anonymous referees for helpful comments and suggestions. We acknowledge support for this from EPSRC via research Grants GR/R45376/01 and GR/R45369/01.

References

[1] M. Abadi, The power of temporal proofs, Theoretical Computer Science 65 (1989) 34–85.
[2] A. Artale, E. Franconi, Temporal description logics, in: M. Fisher, D. Gabbay, L. Vila (Eds.), Handbook of Temporal Reasoning in Artificial Intelligence, vol. 1, Elsevier, Amsterdam, 2004.
[3] A. Artale, E. Franconi, F. Wolter, M. Zakharyaschev, A temporal description logic for reasoning over conceptual schemas and queries, in: Proceedings of JELIA 2002, vol. 2424, LNCS, Springer, Berlin, 2002, pp. 98–110.
[4] L. Bachmair, H. Ganzinger, Resolution theorem proving, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier, Amsterdam, 2001, pp. 19–99 (Chapter 2).

[9] `http://www.csc.liv.ac.uk/liverlab`.

[5] C.-L. Chang, R.C.-T. Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press, New York, 1971.
[6] E.M. Clarke, O. Grumberg, D.A. Peled, Model Checking, MIT Press, Cambridge, MA, 1999.
[7] A. Degtyarev, M. Fisher, B. Konev, Monodic temporal resolution, ACM Transactions on Computational Logic, in press. Preliminary version available as Technical Report ULCS-03-001, The University of Liverpool, http://www.csc.liv.ac.uk/research.
[8] A. Degtyarev, M. Fisher, Towards first-order temporal resolution, in: Proceedings of KI 2001, vol. 2174, LNCS, Springer, Berlin, 2001, pp. 18–32.
[9] A. Degtyarev, M. Fisher, A. Lisitsa, Equality and monodic first-order temporal logic, Studia Logica 72 (2) (2002) 147–156.
[10] C. Dixon, Temporal resolution using a breadth-first search algorithm, Annals of Mathematics and Artificial Intelligence 22 (1998) 87–115.
[11] C. Dixon, Using Otter for temporal resolution, in: Advances in Temporal Logic, Kluwer, Dordrecht, 2000, pp. 149–166.
[12] C. Dixon, Miss Scarlett in the Ballroom with the Lead Piping, in: Proceedings of ECAI 2004, IOS Press, 2004.
[13] C. Dixon, M. Fisher, M. Wooldridge, Resolution for temporal logics of knowledge, Journal of Logic and Computation 8 (3) (1998) 345–372.
[14] C. Dixon, M.-C.F. Gago, M. Fisher, W. van der Hoek, Using temporal logics of knowledge in the formal verification of security protocols, in: Proceedings of TIME 2004, IEEE Computer Society Press, Silver Spring, MD, 2004, pp. 148–151.
[15] E.A. Emerson, Temporal and modal logic, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Elsevier, Amsterdam, 1990, pp. 997–1072 (Chapter 16).
[16] R. Fagin, J. Halpern, Y. Moses, M. Vardi, Reasoning about Knowledge, MIT Press, Cambridge, MA, 1995.
[17] C. Fermüller, A. Leitsch, U. Hustadt, T. Tammet, Resolution decision procedures, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, vol. 2, Elsevier, Amsterdam, 2001, pp. 1791–1850 (Chapter 25).
[18] M. Fisher, An introduction to executable temporal logics, Knowledge Engineering Review 11 (1) (1996) 43–56.
[19] M. Fisher, A temporal semantics for concurrent METATEM, Journal of Symbolic Computation 22 (5/6) (1997) 627–648.
[20] 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.
[21] M. Fisher, C. Dixon, M. Peim, Clausal temporal resolution, ACM Transactions on Computational Logic 2 (1) (2001) 12–56.
[22] M. Fisher, A. Lisitsa, Deductive verification of cache coherence protocols, in: Proceedings of 3rd International Workshop on Automated Verification of Critical Systems (AVoCS 2003), Technical Report DSSE–TR–2003–2, University of Southampton, 2003, pp. 177–186.
[23] M. Fisher, A. Lisitsa, Temporal deductive verification of cache coherence protocols, Technical Report ULCS-03-024, The University of Liverpool, http://www.csc.liv.ac.uk/research.
[24] M. Fisher, A. Lisitsa, Monodic ASMs and temporal verification, in: Proceedings ASM 2004, vol. 3065, LNCS, Springer, Berlin, 2004, pp. 95–110.
[25] D. Gabbay, I. Hodkinson, M. Reynolds, in: Temporal Logic: Mathematical Foundations and Computational Aspects, vol. 1, Oxford University Press, Oxford, 1994.
[26] D.M. Gabbay, A. Kurucz, F. Wolter, M. Zakharyaschev, Many-Dimensional Modal Logics: Theory and Applications, Elsevier, Amsterdam, 2003.
[27] D. Gabbay, M. Reynolds, M. Finger, in: Temporal Logic: Mathematical Foundations and Computational Aspects, vol. 2, Oxford University Press, Oxford, 2000.
[28] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, M. Zakharyaschev, On the computational complexity of spatio-temporal logics, in: Proceedings of FLAIRS 2003, AAAI Press, 2003, pp. 460–464.
[29] Y. Gurevich, Logician in the land of OS: Abstract State Machines at Microsoft, in: Proceedings of LICS 2001, IEEE Computer Society, Silver Spring, MD, 2001, pp. 129–136.
[30] I. Hodkinson, Monodic packed fragment with equality is decidable, Studia Logica 72 (2002) 185–197.
[31] I. Hodkinson, F. Wolter, M. Zakharyaschev, Decidable fragments of first-order temporal logics, Annals of Pure and Applied Logic 106 (2000) 85–134.
[32] G.J. Holzmann, Design and Validation of Computer Protocols, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[33] G.J. Holzmann, The model checker spin, IEEE Transactions on Software Engineering 23 (5) (1997) 279–295.
[34] U. Hustadt, B. Konev, A. Riazanov, A. Voronkov, **TeMP**: A temporal monodic prover, in: Proceedings IJCAR 2004, vol. 3097, LNAI, Springer, Berlin, 2004, pp. 326–330.
[35] R. Kontchakov, C. Lutz, F. Wolter, M. Zakharyaschev, Temporalising tableaux, Studia Logica 76 (1) (2004) 91–134.
[36] O. Kupferman, M. Vardi, Synthesis with incomplete information, in: 2nd International Conference on Temporal Logic, Manchester, 1997, pp. 91–106.
[37] A. Leitsch, The Resolution Calculus, Springer, Berlin, 1997.
[38] Z. Manna, A. Pnueli, The Temporal Logic of Reactive and Concurrent Systems: Specification, Springer, Berlin, 1992.
[39] S. Merz, Decidability and incompleteness results for first-order temporal logic of linear time, Journal of Applied Non-Classical Logics 2 (1992) 139–156.
[40] R. Nieuwenhuis, A. Rubio, Theorem proving with ordering and equality constrained clauses, Journal of Symbolic Computation 19 (4) (1995) 321–351.
[41] R. Nieuwenhuis, A. Rubio, Paramodulation-based theorem proving, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier, Amsterdam, 2001, pp. 371–443 (Chapter 7).
[42] A. Nonnengart, C. Weidenbach, Computing small clause normal forms, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier, Amsterdam, 2001, pp. 335–370 (Chapter 6).
[43] A. Pnueli, The temporal logic of programs, in: Proceedings of the Eighteenth Symposium on the Foundations of Computer Science (FOCS), 1977, pp. 46–57.
[44] A. Pnueli, R. Rosner, On the synthesis of a reactive module, in: Proceedings of Sixteenth Annual ACM Symposium on Principles of Programming Languages, 1989, pp. 179–190.
[45] A.S. Rao, M.P. Georgeff, Decision procedures for BDI logics, Journal of Logic and Computation 8 (3) (1998) 293–343.
[46] A. Riazanov, A. Voronkov, The design and implementation of Vampire, AI Communications 15 (2–3) (2002) 91–110.
[47] R.A. Schmidt, U. Hustadt, A principle for incorporating axioms into the first-order translation of modal formulae, in: Proceedings of CADE-19, vol. 2741, LNAI, Springer, Berlin, 2003, pp. 412–426.
[48] M.P. Shanahan, Solving the Frame Problem, MIT Press, Cambridge, MA, 1997.
[49] A. Szalas, Concerning the semantic consequence relation in first-order temporal logic, Theoretical Computer Science 47 (1986) 329–334.
[50] A. Szalas, L. Holenderski, Incompleteness of first-order temporal logic with until, Theoretical Computer Science 57 (1988) 317–325.
[51] A. Tansel (Ed.), Temporal Databases: Theory, Design, and Implementation, Benjamin/Cummings, Menlo Park, CA, 1993.
[52] 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, Berlin, 1983, pp. 466–483 (original paper (in Russian) appeared in 1968).
[53] F. Wolter, M. Zakharyaschev, Temporalizing description logics, in: Frontiers of Combining Systems II, 2000, pp. 379–401.
[54] F. Wolter, M. Zakharyaschev, Decidable fragments of first-order modal logics, Journal of Symbolic Logic 66 (2001) 1415–1438.
[55] F. Wolter, M. Zakharyaschev, Qualitative spatio-temporal representation and reasoning: a computational perspective, in: Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, Los Altos, CA, 2002, pp. 175–216.
[56] F. Wolter, M. Zakharyaschev, Axiomatizing the monodic fragment of first-order temporal logic, Annals of Pure and Applied logic 118 (2002) 133–145.