Towards the Implementation of First-Order Temporal Resolution: the Expanding Domain Case

Boris Konev*
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, U.K.
B.Konev@csc.liv.ac.uk

Anatoli Degtyarev
Department of Computer Science
King's College London
Strand, London WC2R 2LS, U.K.
Anatoli@dcs.kcl.ac.uk

Clare Dixon
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, U.K.
C.Dixon@csc.liv.ac.uk

Michael Fisher
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, U.K.
M.Fisher@csc.liv.ac.uk

Ullrich Hustadt
Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, U.K.
U.Hustadt@csc.liv.ac.uk

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. In this paper, we develop a clausal resolution method for the monodic fragment of first-order temporal logic over expanding domains. We first define a normal form for monodic formulae and then introduce novel resolution calculi that can be applied to formulae in this normal form. We state correctness and completeness results for the method. We illustrate the method on a comprehensive example. The method is based on classical first-order resolution and can, thus, be efficiently implemented.

1. Introduction

In its propositional form, linear, discrete *temporal logic* has been widely used in the formal specification and verification of reactive systems [18, 15, 12]. Although recognised a powerful formalism, *first-order* temporal logic has generally been avoided due to complexity problems (e.g. there is no finite axiom system for general first-order temporal logic). However, recent work by Hodkinson *et al.* [11] has showed that a particular fragment of first-order temporal logic, termed the *monodic* fragment, has completeness (sometimes even decidability) properties. This breakthrough has led to considerable research activity examining the monodic fragment, in terms of decidable classes, extensions, applications and mechanisation, etc.

Concerning the mechanisation of monodic temporal logics, general tableau and resolution calculi have already been defined, in [13] and [5, 3], respectively. However, neither of these is particularly practical: the tableau method requires representation of *all* possible first-order models, while the resolution method requires the maximal combination of *all* temporal clauses. In this paper, we focus on an important subclass of temporal models, having a wide range of applications, for example in spatio-temporal logics [21, 10] and temporal description logics [1], namely those models that have *expanding domains*. In such models, the domains over which first-order terms range can increase at each temporal step. The focus on this class of models allows us to produce a simplified clausal resolution calculus, termed a *fine-grained* calculus, which is more amenable to efficient implementation.

*On leave from Steklov Institute of Mathematics at St.Petersburg

Thus, we will define the expanding domain monodic fragment, a fine-grained resolution calculus, and provide completeness results for the fine-grained calculus relative to the completeness of the general resolution calculus [6]. A number of examples will be given, showing how the fine-grained calculus works in practice and, finally, conclusions and future work will be provided.

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 linear and discrete model of time (isomorphic to $\mathbb{N}$, and the most commonly used model of time). The first-order temporal language is constructed in a standard way [9, 11] from: *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* $c_0, c_1, \ldots$; *Boolean operators* $\wedge$, $\neg$, $\vee$, $\Rightarrow$, $\equiv$, **true** ('true'), **false** ('false'); *quantifiers* $\forall$ and $\exists$; together with *temporal operators* $\Box$ ('always in the future'), $\Diamond$ ('sometime in the future'), $\bigcirc$ ('at the next moment'), $\mathcal{U}$ (until), and $\mathcal{W}$ (weak until). There are no function symbols or equality in this FOTL language, but it does contain constants. 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).

Formulae in FOTL are interpreted in *first-order temporal structures* of the form $\mathfrak{M} = \langle D_n, I_n \rangle$, $n \in \mathbb{N}$, where every D_n is a non-empty set such that whenever $n < m$, $D_n \subseteq D_m$, and I_n is an interpretation of predicate and constant symbols over D_n. We require that the interpretation of constants is *rigid*. Thus, for every constant c and all moments of time $i, j \geq 0$, we have $I_i(c) = I_j(c)$.

A *(variable) assignment* $\mathfrak{a}$ is a function from the set of individual variables to $\cup_{n\in\mathbb{N}} D_n$. (This definition implies that variable assignments are rigid as well.) We denote the set of all assignments by $\mathfrak{V}$.

For every moment of time n, there is a corresponding *first-order* structure, $\mathfrak{M}_n = \langle D_n, I_n \rangle$; the corresponding set of variable assignments $\mathfrak{V}_n$ is a subset of the set of all assignments, $\mathfrak{V}_n = \{\mathfrak{a} \in \mathfrak{V} \mid \mathfrak{a}(x) \in D_n$ for every variable $x\}$; clearly, $\mathfrak{V}_n \subseteq \mathfrak{V}_m$ if $n < m$. Intuitively, FOTL formulae are interpreted in sequences of *worlds*, $\mathfrak{M}_0, \mathfrak{M}_1, \ldots$ with truth values in different worlds being connected via temporal operators.

The *truth* relation $\mathfrak{M}_n \models^{\mathfrak{a}} \phi$ in a structure $\mathfrak{M}$, *only for those assignments* $\mathfrak{a}$ *that satisfy the condition* $\mathfrak{a} \in \mathfrak{V}_n$, is defined inductively in the usual way under the following understanding of temporal operators:

$\mathfrak{M}_n \models^{\mathfrak{a}} \bigcirc\phi$	iff	$\mathfrak{M}_{n+1} \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} \Diamond\phi$	iff	there exists $m \geq n$ such that $\mathfrak{M}_m \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} \Box\phi$	iff	for all $m \geq n$, $\mathfrak{M}_m \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} (\phi \mathcal{U} \psi)$	iff	there exists $m \geq n$, such that $\mathfrak{M}_m \models^{\mathfrak{a}} \psi$, and for all $i \in \mathbb{N}$, $n \leq i < m$ implies $\mathfrak{M}_i \models^{\mathfrak{a}} \phi$;
$\mathfrak{M}_n \models^{\mathfrak{a}} (\phi \mathcal{W} \psi)$	iff	$\mathfrak{M}_n \models^{\mathfrak{a}} (\phi \mathcal{U} \psi)$ 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 there exists an assignment $\mathfrak{a}$ in D_0 such that $\mathfrak{M}_0 \models^{\mathfrak{a}} \phi$. A formula is *satisfiable* if it has a model. A formula is *valid* 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*. 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. It is known [19] that satisfiability over expanding domains can be reduced to satisfiability over constant domains.

Example 1 The formula $\forall x P(x) \wedge \Box(\forall x P(x) \Rightarrow \bigcirc\forall x Q(x)) \wedge \Diamond\neg Q(c)$ is unsatisfiable over both expanding and constant domains; the formula $\forall x P(x) \wedge \Box(\forall x (P(x) \Rightarrow \bigcirc Q(x))) \wedge \Diamond\neg Q(c)$ is unsatisfiable over constant domains but has a model with an expanding domain.

This logic is complex. It is known that even "small" fragments of FOTL, such as the *two-variable monadic* fragment (all predicates are unary), are not recursively enumerable [16, 11]. However, the set of valid *monodic* formulae (see Definition 1 below) is known to be finitely axiomatisable [20].

Definition 1 An FOTL-formula ϕ is called *monodic* if any subformulae 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 $\mathcal{U}$, $\mathcal{W}$), contains at most one free variable.

3. Divided Separated Normal Form (DSNF)

Definition 2 A *temporal step clause* is a formula either of the form $p \Rightarrow \bigcirc l$, where p is a proposition and l is a propositional literal, or $(P(x) \Rightarrow \bigcirc M(x))$, where $P(x)$ is a unary predicate and $M(x)$ is a unary literal. We call a clause of the the first type an (original) *ground* step clause, and of the second type an (original) *non-ground* step clause.

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

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

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

Note that, in a monodic temporal problem, we do not allow two different temporal step clauses with the same left-hand sides. A problem 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 mean properties of the associated formula.

Arbitrary monodic FOTL-formulae can be transformed into DSNF in a satisfiability equivalence preserving way using a renaming technique replacing non-atomic subformulae with new propositions and removing all occurrences of the U and W operators [9, 5].

4. Completeness Calculus

A resolution-like procedure for the monodic fragment over constant domains has been introduced in [5]. Although satisfiability over expanding domains can be reduced to satisfiability over constant domains [19], it has been proved in [6] that a simple modification of the procedure can be directly applied to the expanding domain case. We sketch the monodic temporal resolution system here to make the paper self-contained. We use this 'completeness calculus' to show relative completeness of the calculus presented in the next section. More details on the completeness calculus, as well as proofs of the properties stated below, can be found in [5] and [6] for the constant and expanding domain cases, respectively.

Let P be a monodic temporal problem, and let

$$P_{i_1}(x) \Rightarrow \bigcirc M_{i_1}(x), \ldots, P_{i_k}(x) \Rightarrow \bigcirc M_{i_k}(x) \qquad (1)$$

be a subset of the set of its original non-ground step clauses. Then formulae of the form

$$P_{i_j}(c) \quad \Rightarrow \quad \bigcirc M_{i_j}(c), \qquad (2)$$

$$\exists x \bigwedge_{j=1}^{k} P_{i_j}(x) \quad \Rightarrow \quad \bigcirc \exists x \bigwedge_{j=1}^{k} M_{i_j}(x), \qquad (3)$$

$$\forall x \bigvee_{j=1}^{k} P_{i_j}(x) \quad \Rightarrow \quad \bigcirc \forall x \bigvee_{j=1}^{k} M_{i_j}(x) \qquad (4)$$

are called *derived* step clauses, where $c \in const(\mathsf{P})$ and $j = 1 \ldots k$. Formulae of the form (2) and (3) are called *e-derived* step clauses. Note that formulae of the form (2) and (3) are logical consequences of (1) in the *expanding domain* case; while formulae of the form (2), (3), *and* (4) are logical consequences of (1) in the *constant domain* case. As Example 1 shows, (4) is not a logical consequence of (1) in the expanding domain case.

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

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

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

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

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

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

Let P be a monodic temporal problem,

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

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

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

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

- *Step resolution rule w.r.t.* $\mathcal{U}$: $\dfrac{\mathcal{A} \Rightarrow \bigcirc\mathcal{B}}{\neg\mathcal{A}}\ (\bigcirc^{\mathcal{U}}_{res})$, where $\mathcal{U} \cup \{\mathcal{B}\} \vdash \perp$.
- *Initial termination rule w.r.t.* $\mathcal{U}$*:* The contradiction $\perp$ is derived and the derivation is (successfully) terminated if $\mathcal{U} \cup I \vdash \perp$.
- *Eventuality resolution rule w.r.t.* $\mathcal{U}$*:*

$$\frac{\begin{array}{ll}\forall x(\mathcal{A}_1(x) \Rightarrow \bigcirc(\mathcal{B}_1(x))) & \\ \vdots & \Diamond L(x) \\ \forall x(\mathcal{A}_n(x) \Rightarrow \bigcirc(\mathcal{B}_n(x))) & \end{array}}{\forall x \bigwedge_{i=1}^{n} \neg\mathcal{A}_i(x)}\ (\Diamond^{\mathcal{U}}_{res}),$$

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

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

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

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

A *derivation* is a sequence of universal parts, $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \mathcal{U}_2 \subseteq \ldots$, extended little by little by the conclusions of the inference rules. Successful termination means that the given problem is unsatisfiable. The I, $\mathcal{S}$ and $\mathcal{E}$ parts of the temporal problem are not changed in a derivation.

Theorem 1 (see [5], theorems 2 and 3) The rules of $\mathfrak{I}_c$ preserve satisfiability over constant domains. If a monodic temporal problem P is unsatisfiable over constant domains, then there exists a successfully terminating derivation in $\mathfrak{I}_c$ from P^c.

Theorem 2 (see [6], theorems 2 and 3) The rules of $\mathfrak{I}_e$ preserve satisfiability over expanding domains. If a monodic temporal problem P is unsatisfiable over expanding domains, then there exists a successfully terminating derivation in $\mathfrak{I}_e$ from P^c.

Example 2 The need for constant flooding can be demonstrated by the following example. None of the rules of temporal resolution can be applied directly to the (unsatisfiable) temporal problem given by

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

If, however, we add to the problem an eventuality clause $\Diamond l$ and a universal clause $l \Rightarrow \neg P(c)$, the step clause $q \Rightarrow \bigcirc q$ will be a loop in $\Diamond l$, and the eventuality resolution rule would derive $\neg$**true**[2].

5. Fine-Grained Resolution for the Expanding Domain Case

The main drawback of the calculi introduced in the previous section is that the notion of a merged step clause is quite involved and the search for appropriate merging of simpler clauses is computationally hard. Finding *sets* of such full merged step clauses needed for the temporal resolution rule is even more difficult.

From now on we focus on the expanding domain case. This is simpler firstly because merged e-derived step clauses are simpler (formulae of the form (4) do not contribute to them) and, secondly, because conclusions of all inference rules of $\mathfrak{I}_e$ are first-order clauses.

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

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

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

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

step resolution rule is applied to another merged e-derived step clause, $\mathcal{A}' \Rightarrow \bigcirc\mathcal{B}'$. The side condition, $\mathcal{B}' \cup \mathcal{U}' \vdash \bot$, again can be checked by a first-order resolution procedure. Since we never add new existential formulae, $\mathcal{U}'$ can be skolemised in exactly the same way as $\mathcal{U}$. Therefore, we can actually keep $\mathcal{U}$ in clausal form.

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

Fine-grained resolution might generate additional step clauses of the form

$$C \Rightarrow \bigcirc D. \tag{5}$$

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

Definition 4 Let P be a constant flooded temporal problem; the set of clauses $\mathbf{S}(\mathsf{P})$, called *the result of preprocessing*, consists of step clauses from P and

1. For every original non-ground step clause

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

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

   $$P(c) \Rightarrow \bigcirc M(c) \tag{6}$$

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

2. Clauses obtained by clausification of the universal and initial parts, as if there is no connection with temporal logic at all, are in $\mathbf{S}(\mathsf{P})$. The resulting clauses are called *universal clauses* and *initial clauses* resp. Originally, universal and initial clauses do not have common Skolem constants and functions. Initial and universal clauses are kept separately.

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

5.1. Fine-grained step resolution

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

Deduction rules

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

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

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

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

   where $C_1 \Rightarrow \bigcirc(D_1 \vee L)$ and $C_2 \Rightarrow \bigcirc(D_2 \vee \neg M)$ are step clauses and σ is an mgu 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.*[3]

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

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

4. *Right factor*

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

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

5. *Left factor*

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

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

6. *Clause conversion*
   a step clause of the form $C \Rightarrow \bigcirc$**false** is rewritten into the *universal clause* $\neg C$.

Deletion rules

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

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

2. *Temporal deletion:*
A universal clause D_2 *subsumes* a step clause $C_1 \Rightarrow \bigcirc D_1$ if D_2 subsumes D_1 or D_2 subsumes[4] $\neg C_1$.
A step clause $C_1 \Rightarrow \bigcirc D_1$ *subsumes* a step clause $C_2 \Rightarrow \bigcirc D_2$ if there exists a substitution σ such that $D_1\sigma \subseteq D_2$ and $\neg C_1\sigma \subseteq \neg C_2$.
A step clause $C \Rightarrow \bigcirc D$ is a *tautology* if D is a tautology. (Note that, since we do not have negative occurrences to the left-hand side of step clauses, C cannot be false). Tautologies are deleted.

We adopt the terminology from [2]. A (linear) *proof* by fine-grained resolution of a clause C from a set of clauses $\mathbf{S}$ is a sequence of clauses $C_1, \dots, C_m$ such that $C = C_m$ and each clause C_i is either an element of $\mathbf{S}$ or else the conclusion by a deduction rule from $C_1, \dots, C_{i-1}$. A proof of **false** is called a *refutation*. A (theorem proving) *derivation* by fine-grained resolution is a sequence of sets of clauses $\mathbf{S_0} \triangleright \mathbf{S_1} \triangleright \dots$ such that every $\mathbf{S_{i+1}}$ differs from $\mathbf{S_i}$ by either adding the conclusion of a deduction rule or else deleting a clause by a deletion rule. We say that a clause C is *derived by fine-grained resolution from* $\mathbf{S_0}$ if $C \in \mathbf{S_i}$ for some i.

Note 1 Fine-grained step resolution without the restriction on substitutions would, certainly, lead to unsoundness: The monodic problem given by

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

which is satisfiable, would wrongly be declared unsatisfiable without this restriction (After skolemisation, $\mathcal{U}^s = \{us1 : \neg Q(c),\ us2 : P(x) \vee Q(x)\}$, then unrestricted resolution would derive $us3 : \neg P(c)$ from $us1$ and $s1$, and then the contradiction from $us1$, $us2$, and $us3$.)

Example 3 It might seem that the restriction on mgus is too strong and destroys completeness of the calculus. For example, at first glance it may appear that under this restriction it is not possible to deduce a contradiction from the following (unsatisfiable) temporal problem P given by

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

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

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

A formal statement of completeness follows.

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

Definition 5 A clause of the form $C \Rightarrow \bigcirc\mathbf{false}$, where C is of the same form as in (5), is called *a final clause*.

Lemma 3 Let $\mathsf{P} = \langle \mathcal{U}, I, \mathcal{S}, \mathcal{E} \rangle$ be a monodic temporal problem and $\mathbf{S} = \mathbf{S}(\mathsf{P})$ be the result of preprocessing. Let $C \Rightarrow \bigcirc\mathbf{false}$ be an arbitrary final clause derived by fine-grained step resolution from $\mathbf{S}$. Then there exists a derivation $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \dots$ by the step resolution rule of $\mathfrak{I}_e$ and a merged e-derived step clause $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ such that $\mathcal{B} \cup \mathcal{U}_i \vdash \perp$, for some $i \geq 0$, and $\mathcal{A} = \widetilde{\exists} C$, where $\widetilde{\exists}$ means existential quantification over all free variables.

Proof (Sketch). Since $C \Rightarrow \bigcirc\mathbf{false}$ is derivable, there exists its proof Γ by fine-grained resolution. We prove the lemma by induction on the number of applications of the clause conversion rule in Γ. Suppose we proved the lemma for proofs containing less than n applications of the clause conversion rule, and let Γ contains n such applications. Then every conclusion of the clause conversion rule is also a conclusion by the step resolution rule of $\mathfrak{I}_e$. It can be shown that both the induction basis and induction step follow from the following claim.

Claim. Let Δ be a proof of $C \Rightarrow \bigcirc\mathbf{false}$ by the rules of fine-grained resolution, *except the clause conversion rule*, from a set of step clauses $\mathcal{S}$ and a set of universal clauses $\mathcal{U}$. Then there exists a merged e-derived step clause $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ such that $\mathcal{B} \cup \mathcal{U} \vdash \perp$ and $\mathcal{A} = \widetilde{\exists} C$.

Let

$$\begin{array}{rcl} \{P_i(x_i) & \Rightarrow & \bigcirc M_i(x_i) \mid i = 1 \dots K\} \\ \{p_i & \Rightarrow & \bigcirc l_i \mid i = 1 \dots L\} \end{array}$$

be the set of all step clauses from $\mathcal{S}$ involved in Δ where $p_i \Rightarrow \bigcirc l_i$ denotes either a ground step clause, or an e-derived step-clause of the form (6) added by preprocessing (w.l.o.g., we assumed that all the variables x_1,.., x_K are pairwise distinct). We assume that Δ is *tree-like*, that is, no clause in Δ is used more than once as an assumption for an inference rule; we may make copies of the clauses in Δ in order to make it tree-like.

Note that (by accumulating the mgus 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 mgus used in the proof are empty[5]. That is, there exist substitutions $\{\sigma_{i,j} \mid i = 1 \dots K, j = 1 \dots s_i\}$ such that

$$\begin{array}{rcl} \{P_i(x_i)\sigma_{i,j} & \Rightarrow & \bigcirc M_i(x_i)\sigma_{i,j} \mid i = 1 \dots K, j = 1 \dots s_i\} \\ \{p_i & \Rightarrow & \bigcirc l_i \mid i = 1 \dots L\} \end{array} \tag{7}$$

[5] 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*.

(together with some instances of universal clauses) contribute to the proof of $C \Rightarrow \bigcirc$**false** where all mgus used in the proof are empty, and, furthermore,

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

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

Let us group the instances of the step clauses according to the value of the substitutions. We introduce an equivalence relation Σ on the clauses from (7) as follows: For every i, j, i', j' we have $(P_i(x_i)\sigma_{i,j} \Rightarrow \bigcirc M_i(x_i)\sigma_{i,j},\ P_{i'}(x_{i'})\sigma_{i',j'} \Rightarrow \bigcirc M_{i'}(x_{i'})\sigma_{i',j'}) \in \Sigma$ iff $x_i\sigma_{i,j} = x_{i'}\sigma_{i',j'}$ (it can be easily checked that Σ is indeed an equivalence relation). Let N be the number of equivalence classes of (7) by Σ; let I_k be the set of indexes of the k-th equivalence class (we refer to clauses from (7) by indexes of the corresponding substitutions).

Let $C_k = \bigwedge_{(i,j)\in I_k} P_i(x_i)\sigma_{i,j}$, for every k, $1 \le k \le N$; let $C_0 = \bigwedge_{i=1}^{L} p_i$. Note that $C = \bigwedge_{k=1}^{N} C_k \wedge C_0$ and this partition of C is disjoint. Let $D_k = \bigwedge_{(i,j)\in I_k} M_i(x_i)\sigma_{i,j}$, let $D_0 = \bigwedge_{i=1}^{L} l_i$, let $D = \bigwedge_{k=1}^{N} D_k \wedge D_0$. Note that $\widetilde{\forall} D \wedge \mathcal{U} \vdash \perp$. 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 mgus applied to universal and intermediate clauses only). It follows that $\bigwedge_{k=1}^{N} \exists x D_k(x) \wedge D_0 \wedge \mathcal{U} \vdash \perp$.

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

Lemma 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 preprocessing. Let $\mathcal{U} = \mathcal{U}_0 \subseteq \mathcal{U}_1 \subseteq \ldots$ be a derivation by the step resolution rule of $\mathfrak{I}_e$. Let $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ be a merged e-derived step clause such that $\mathcal{B} \cup \mathcal{U}_i \vdash \perp$, for some $i \ge 0$. Then there exists a final clause $C \Rightarrow \bigcirc$**false**, derived by fine-grained resolution from **S**, such that $\mathcal{A} \Rightarrow \widetilde{\exists} C$.

Proof (Sketch). As in the proof of the previous lemma, it suffices to prove that under conditions of the lemma there exists a proof of a final clause $C \Rightarrow \bigcirc$**false** from the set of step clauses from **S** and the (current) universal part, $\mathcal{U}_n$, by the rules of fine-grained resolution, *except the clause conversion rule*, such that $\mathcal{A} \Rightarrow \widetilde{\exists} C$.

The clause $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ is merged from derived clauses of the form (2) and (3). Note that all derived clauses of the form (2) are in **S** ought to preprocessing. Let for every derived rule of the form (3),

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

consider a set of instances of non-ground step clauses from **S**,

$$\{P_{i_j}(c) \Rightarrow M_{i_j}(c) \mid j = 1 \ldots s\},$$

where c is a new constant.

Since $\mathcal{B} \cup \mathcal{U}_n \vdash \perp$, there exists a set of instances of step clauses (we simplify indexing for the sake of presentation)

$$\begin{array}{rcl} \{P_j(c_i) & \Rightarrow & \bigcirc M_j(c_i)\} \mid i = 1 \ldots K, j = 1 \ldots s_i\} \\ \{p_i & \Rightarrow & \bigcirc l_i \mid i = 1 \ldots L\}, \end{array}$$

where $c_1, \ldots, c_K$ are new (Skolem) constants, such that $\bigwedge_{i=1}^{K} \bigwedge_{j=1}^{s_i} M_j(c_i) \wedge \bigwedge_{i=1}^{L} l_i \wedge \mathcal{U}_n \vdash \perp$ (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 (6) added by preprocessing).

Let Δ be a (first-order) resolution proof of $\perp$ from $\mathcal{U}_n$ and the following set of clauses $\{M_j(c_i) \mid i = 1 \ldots K, j = 1 \ldots s_i\} \cup \{l_i \mid i = 1 \ldots L\}$. 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). Then there exists a proof Γ by fine-grained step resolution from

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

(and universal clauses) of a final clause $C \Rightarrow \bigcirc$**false**, where $C = \bigwedge_{(i,j)\in I} P_j(c_i) \wedge \bigwedge_{j\in J} p_i$.

We assume, for simplicity of the proof, that the lifting theorem (cf. e.g. [14]) holds for Δ, that is, 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 $\Delta \le_s \Delta'$ in the terminology of [14]: Every clause C'_i of Δ' is a generalisation of the corresponding clause C_i of Δ.

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

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

(and universal clauses) of a final clause $C' \Rightarrow \bigcirc$**false** such that $\Gamma' \ge_s \Gamma$, that is, every intermediate clause $C'_i \Rightarrow \bigcirc D'_i$ from Γ' is a generalisation of a corresponding clause from Γ. (The only difficulty is to ensure the requirement on mgus 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', a constant or a functional term was substituted into a variable occurring in the left-hand side of a clause, this clause would not be a generalisation of any clause from Γ.) This implies the conclusion of the lemma. ❑

Lemma 3 ensures soundness of fine-grained step resolution. Lemma 4 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 5 The calculus consisting of the rules of fine-grained step resolution, together with the (both ground and non-ground) eventuality resolution rule, is sound and complete for the monodic fragment over expanding domains.

Note 2 The proof of completeness given above might be hard to fulfil in the presence of various *refinements* of resolution and/or *redundancy deletion*. As a remedy, we suggest considering *constrained calculi*, like e.g. resolution over constrained clauses with constraint inheritance. It is known that such inference systems are complete and moreover compatible with redundancy elimination rules and many (liftable) refinements (see e.g. [17], theorems 5.11 and 5.12, subsections 5.4 and 5.5, resp.). Here we take into account that there are no clauses with equality, and therefore all sets are *well-constrained* in the terminology of [17].

Then instead of ground clauses of the form

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

we consider their *constrained* representations

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

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

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

is just

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

So, all clauses originating from the universal part have empty constraints and all temporal clauses have constraints 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 it is impossible 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.

5.2. Loop search

Next we use fine-grained step resolution to find the appropriate set of full e-merged clauses to apply the (ground or non-ground) eventuality resolution rule. It has been noticed in [5] that in order to effectively find a loop in $\Diamond L(x) \in \mathcal{E}$, given a formula with one free variable $\Phi(x)$ we have to be able to find the set of all full e-merged clauses of the form $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc\mathcal{B}(x))$ such that the formula

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

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

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

Definition 6 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 7 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

$$P(c^l) \Rightarrow \bigcirc M(c^l). \tag{8}$$

We add the clause[6] $\mathbf{true} \Rightarrow \bigcirc\neg\Phi(c^l)$ to $\mathrm{LT}(\mathbf{S})$ and apply the rules of fine-grained step resolution *except the clause conversion rule* to it.

Lemma 6 Let $\mathbf{S}$ be a set of universal and step clauses, and let $C \Rightarrow \bigcirc\mathbf{false}$ be a final clause derived by the rules of fine-grained step resolution *except the clause conversion rule* from $\mathrm{LT}(\mathbf{S})\cup\{\mathbf{true} \Rightarrow \bigcirc\neg\Phi(c^l)\}$ such that at least one of the clauses originating from $\mathbf{true} \Rightarrow \bigcirc\neg\Phi(c^l)$ is involved in the derivation. Then there exists a full e-merged (from $\mathbf{S}$) clause $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc\mathcal{B}(x))$ such that the formula $\forall x(B(x)\wedge\mathcal{U} \Rightarrow \Phi(x))$ is valid and $\mathcal{A}(x) = (\widetilde{\exists}C)\{c^l \to x\}$.

Proof (Sketch). By Lemma 3, there exists a merged (from $LT(S)$) e-derived clause $\mathcal{A} \Rightarrow \bigcirc\mathcal{B}$ such that $\{\neg\Phi(c^l)\}\cup\mathcal{B}\cup\mathcal{U}\vdash\perp$ and $\mathcal{A} = \widetilde{\exists}C$. It suffices to notice that $\forall x((\mathcal{A} \Rightarrow \bigcirc\mathcal{B})\{c^l \to x\})$ is a full merged (from S) step clause and $\exists x(\Phi(x)\wedge\mathcal{B}\{c^l \to x\}\wedge\mathcal{U})$ is unsatisfiable. ❑

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

Function BFS

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

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

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

Figure 1. Breadth-first search using fine-grained step resolution.

Lemma 7 Let **S** be a set of universal and step clauses, and let $\forall x(\mathcal{A}(x) \Rightarrow \bigcirc\mathcal{B}(x))$ be a full e-merged (from **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 $\forall x(\mathcal{A}(x) \Rightarrow (\widetilde{\exists} C\{c^l \to x\}))$.

Proof (Sketch). The proof is analogous to the proof of Lemma 4. As we already noticed, $\exists x(\mathcal{B}(x) \wedge \mathcal{U} \wedge \neg\Phi(x))$ is unsatisfiable, and this can be checked by a first-order resolution procedure. Since c^l does not occur in the problem, we can skolemise this existential quantifier with c^l. We lift now all Skolem constants but c^l. ❑

Then the loop search algorithm from [5] can be reformulated as shown in Fig. 1. (This algorithm is essentially based on the BFS algorithm for propositional temporal resolution [7].)

Lemma 8 The BFS algorithm terminates provided that all calls of saturation by step resolution terminate. If BFS returns non-false value, its output is a loop formula in $L(x)$.

Note 3 Termination of calls by step resolution can be achieved for the cases when there exists a (first-order) resolution decision procedure [8] for formulae in the universal part, see also [4].

Theorem 9 The calculus consisting of the rules of fine-grained step resolution, together with the (both ground and non-ground) eventuality resolution rule, is complete for the monodic fragment over expanding domains even if we restrict ourselves to loops found by the BFS algorithm.

5.3. Example

Let us consider a monodic temporal problem P given by $I = \emptyset$, $\mathcal{U} = \{\forall x(B(x) \Rightarrow A(x) \wedge \neg L(x)),\ l \Rightarrow \exists x A(x)\}$, $\mathcal{S} = \{s1 : A(x) \Rightarrow \bigcirc B(x)\}$, $\mathcal{E} = \{e1 : \Diamond L(x),\ e2 : \Diamond l\}$. We especially chose such a trivial example to be able to demonstrate thoroughly the steps of our proof search algorithm. We clausify $\mathcal{U}$ resulting in $\mathcal{U}^s = \{u1 : (\neg B(x) \vee A(x)),\ u2 : (\neg B(x) \vee \neg L(x)),\ u3 : \neg l \vee A(c)\}$.

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

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

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

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

  The set of clauses is saturated. Then $N_1 = \{A(c^l) \Rightarrow \bigcirc\mathbf{false}\}$, $H_1(x) = A(x)$. Obviously, $\forall x(H_0(x) \Rightarrow H_1(x))$ is not true.

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

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

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

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

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

- Step resolution

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

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

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

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

Note 4 As the example shows, the presence of clauses of the form (6), introduced by preprocessing, and (8), introduced by the transformation for loop search, might lead to repeated derivations (with free variables and with constants). This can be avoided, however, if instead of generating these clauses, we relax the conditions on substitutions in the definition of rules of fine-grained resolution by allowing original constants and the loop constant to be substituted to variables occurring in the left-hand side of a step clause. It can be seen that the set of derived final clauses would be the same.

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

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

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

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

6. Conclusion

We have described a fine-grained resolution calculus for monodic first order temporal logics over expanding domains. Soundness of the fine-grained inference steps is easy to prove and completeness is shown relative to the completeness proof for the expanding domain for the non-fine grained version [6]. While the implementation based on the general calculus would involve generating all subsets of the step clauses with which to apply the step and eventuality resolution rules, the fine-grained resolution inference rules can be implemented directly using any appropriate first-order theorem prover for classical logics. This makes the new calculus presented here particularly amenable to efficient implementation.

As part of our future work, we will examine the extension of this approach to the case of temporal models with constant domains. We also aim to implement and test the calculus defined here.

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

References

[1] A. Artale, E. Franconi, F. Wolter, and M. Zakharyaschev. A temporal description logic for reasoning over conceptual schemas and queries. In *Proceedings of JELIA'02*, volume 2424 of *LNCS*, pages 98–110. Springer, 2002.

[2] L. Bachmair and H. Ganzinger. Resolution theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, chapter 2, pages 19–99. Elsevier, 2001.

[3] A. Degtyarev and M. Fisher. Towards first-order temporal resolution. In *KI 2001, Proceedings*, volume 2174 of *LNCS*, pages 18–32. Springer, 2001.

[4] A. Degtyarev, M. Fisher, and B. Konev. Exploring the monodic fragment of first-order temporal logic using clausal temporal resolution. Technical Report ULCS-03-012, University of Liverpool, Department of Computer Science, 2003. `http://www.csc.liv.ac.uk/research/`.

[5] A. Degtyarev, M. Fisher, and B. Konev. Monodic temporal resolution. In *Proc. CADE-19, to appear*, LNAI. Springer, 2003. Available as Technical report ULCS-03-001 from `http://www.csc.liv.ac.uk/research/`.

[6] A. Degtyarev, M. Fisher, and B. Konev. Monodic temporal resolution: the expanding domain case.

Technical Report ULCS-03-004, University of Liverpool, Department of Computer Science, 2003. `http://www.csc.liv.ac.uk/research/`.

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

[8] C. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. Resolution decision procedures. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 25, pages 1791–1850. Elsevier, 2001.

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

[10] D. Gabelaia, R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev. On the computational complexity of spatio-temporal logics. To appear in the proceedings of *FLAIRS*, 2003.

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

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

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

[14] Alexander Leitsch. *The Resolution Calculus*. Springer, 1997.

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

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

[17] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, chapter 7, pages 371–443. Elsevier, 2001.

[18] A Pnueli. The Temporal Logic of Programs. In *Proceedings of the Eighteenth Symposium on the Foundations of Computer Science*, 1977.

[19] F. Wolter and M. Zakharyaschev. Decidable fragments of first-order modal logics. *Journal of Symbolic Logic*, 66:1415–1438, 2001.

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

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