Extending the branching-time logic programming language Cactus

Manolis Gergatsoulis

Institute of Informatics and Telecommunications
National Centre for Scientific Research (NCSR) ‘Demokritos’
Aghia Paraskevi Attikis, Athens, Greece
email: manolis@iit.demokritos.gr

Abstract

Cactus has been proposed as a temporal logic programming language based on the branching notion of time. Cactus supports two main operators: the temporal operator first which refers to the beginning of time and the temporal operator next, which refers to the i-th child of the current moment. Actually we have a family \{next, i ∈ N\} of next operators, each one of them representing a different next moment that immediately follows the present one. In this paper we propose the extension of the language Cactus with new temporal operators such as ⊪F and ⊪G referring to ‘some next moment’ and ‘all next moments’ respectively, as well as with variants of the operators □ and ◻.

1 Introduction

Temporal programming languages [OM94, FO95] provide a powerful means for the description and implementation of dynamic systems. Most temporal logic programming languages [OM94, OWD93, Bau93, Brz93, GRP96, Org91] are based on linear flow of time. Recently some research work has been done in the direction of developing temporal logic programming languages based on the branching notion of time. In [RGP97, GRP97, RGP98] the temporal logic programming language Cactus has been presented. Cactus is based on a tree-like notion of time; that is, every moment in time may have more than one immediate next moments.

Cactus supports two main operators: the temporal operator first refers to the beginning of time (or alternatively to the root of the tree). The temporal operator next, refers to the i-th child of the current moment (or alternatively, the i-th branch of the current node in the tree). Notice that we actually have a family \{next, i ∈ N\} of next operators, each one of them representing the different next moments that immediately follow the present one.

As an example consider the non-deterministic finite automaton shown in figure 1 which accepts the regular language L = (01 ∪ 010)*. We can describe the behaviour
of this automaton in Cactus with the following program:

\[
\begin{align*}
\text{first state}(q_0). \\
\text{next}_0 \text{ state}(q_1) & \leftarrow \text{state}(q_0). \\
\text{next}_1 \text{ state}(q_0) & \leftarrow \text{state}(q_1). \\
\text{next}_1 \text{ state}(q_2) & \leftarrow \text{state}(q_1). \\
\text{next}_0 \text{ state}(q_0) & \leftarrow \text{state}(q_2).
\end{align*}
\]

Figure 1: A non-deterministic finite automaton

Notice that q0 is both the initial and the final state. Posing the goal clause:

\[
\leftarrow \text{first next}_0 \text{ next}_1 \text{ next}_0 \text{ state}(q_0).
\]

will return the answer yes which indicates that the string 010 belongs to the language \( L \).

In [RGP98], it is argued that Cactus is appropriate for describing non-deterministic computations or more generally computations that involve the manipulation of trees. Moreover, as it is shown in [RG98], a fragment of Cactus can be used as a target language for a transformation of a subclass of Datalog programs, namely the chain Datalog ones. The Cactus programs obtained by applying this transformation are simpler in structure and it is believed that they can be implemented efficiently. An evidence for this is that similar transformation techniques have been used in functional programming [RW97] giving as output zero-order branching-time functional programs which have simpler structure and can be easily evaluated using demand-driven techniques.

In this paper, we propose the extension of Cactus with more expressive temporal operators. More specifically, in the extended Cactus, besides the operators first and next, we propose to se of the operators \( \bigcirc_F \), and \( \bigcirc \) in the clause bodies and the operators \( \bigcirc_C \), and \( \Box \) in the clause head. The intuitive meaning of \( \bigcirc_F \) is "some next moment" while the intuitive meaning of \( \bigcirc \) is "some moment in the future of the first moment in time". Moreover, the intuitive meaning of \( \bigcirc_C \) is "all next moments" while the intuitive meaning of \( \Box \) is "all moments in the future of the first moment in time".
2 The temporal logic of the extended Cactus

2.1 Temporal operators

In [RGP97, GRP97, RGP98] it is presented a branching-time temporal logic (BTL) on which the language Cactus is based. In BTL, time varies over a tree-like structure. The set of moments in time can be modeled by the set \( \text{List}(\mathcal{N}) \) of lists of natural numbers \( \mathcal{N} \). Thus, each node may have a countably infinite number of branches. The empty list \([\ ]\) corresponds to the beginning of time and the list \([i:t]\) (that is, the list with head \( i \in \mathcal{N} \), and tail \( t \)) corresponds to the \( i \)-th child of the moment identified by the list \( t \). BTL has two temporal operators namely \text{first} and \text{next}, \( i \in \mathcal{N} \). The operator \text{first} is used to express the first moment in time, while \text{next} refers to the \( i \)-th child of the current moment in time.

Branching-time logics in which richer temporal operators are defined have been proposed in the literature [BAPM83, Eme90]. In this section we consider the extension of the branching-time logic BTL with the temporal operators \( \Diamond_G, \Box_G, \Diamond_F \text{ and } \Box_G \). The meaning of these temporal operators is as follows:

- \( \Box_G A \): holds at \( t \) iff the formula \( A \) is true at every immediate successor of \( t \).
- \( \Diamond_G A \): holds at \( t \) iff there is an immediate successor of \( t \) at which \( A \) is true.
- \( \Diamond_F A \): holds at \( t \) iff there is some node in the subtree rooted from \( t \) at which \( A \) is true.
- \( \Box_G A \): holds at \( t \) iff \( A \) is true at all time points (nodes) of the subtree rooted at \( t \) (including \( t \)).

These operators are simiral to the operators of the branching-time logic of Ben-Ari, Pnueli and Manna [BAPM83]\(^1\).

The syntax of the formulae of the branching-time logic extends the syntax of first order logic with six new formation rules: If \( A \) is a formula, so are \text{first} \( A \), \text{next} \( A \), \( \Box_G A, \Diamond_G A, \Diamond_F A \text{ and } \Box_G A \).

In the following we will also use the operators \( \Diamond \text{ and } \Box \), as a shorthand for the sequences of operators \text{first} \( \Diamond_F \) and \text{first} \( \Box_G \) respectively.

2.2 Semantics of the formulas of the extended BTL

The semantics of temporal formulae of the extended BTL are given using the notion of branching temporal interpretation.

\textbf{Definition 2.1.} A branching temporal interpretation or simply a temporal interpretation \( I \) of the extended BTL comprises a non-empty set \( D \), called the domain of the interpretation, over which the variables range, together with an element of \( D \) for each variable; for each \( n \)-ary function symbol, an element of \([D^n \to D] \); and for each \( n \)-ary predicate symbol, an element of \([\text{List}(\mathcal{N}) \to 2^{D^n}] \).

In the following definition, the satisfaction relation \( \models \) is defined in terms of temporal interpretations. \( \models_{I,t} A \) denotes that the formula \( A \) is true at the moment \( t \) in

\[\text{The operators } \Diamond_G, \Box_G, \text{ and } \Diamond_F \text{ are represented in [BAPM83] as } \forall G, \forall X, \text{ and } \exists F \text{ respectively.}\]
the temporal interpretation \( I \). Finally, by \( \text{sof} \) we denote the usual list concatenation of lists \( s \) and \( t \).

**Definition 2.2.** The semantics of the elements of the extended BTL are given inductively as follows:

1. If \( \text{sf}(e_0, \ldots, e_{n-1}) \) is a term, then \( I(\text{sf}(e_0, \ldots, e_{n-1})) = I(\text{sf})(I(e_0), \ldots, I(e_{n-1})) \).
2. For any \( n \)-ary predicate symbol \( p \) and terms \( e_0, \ldots, e_{n-1} \),
   \[ \models_{I,t} p(e_0, \ldots, e_{n-1}) \text{ iff } (I(e_0), \ldots, I(e_{n-1})) \in I(p)(t) \]
3. \( \models_{I,t} \neg A \text{ iff it is not the case that } \models_{I,t} A \)
4. \( \models_{I,t} A \land B \text{ iff } \models_{I,t} A \text{ and } \models_{I,t} B \)
5. \( \models_{I,t} A \lor B \text{ iff } \models_{I,t} A \text{ or } \models_{I,t} B \)
6. \( \models_{I,t} (\forall x)A \text{ iff } \models_{I[d/x],t} A \text{ for all } d \in D \text{ where the interpretation } I[d/x] \text{ is the same as } I \text{ except that the variable } x \text{ is assigned the value } d. \)
7. \( \models_{I,t} \text{first } A \text{ iff } \models_{I[i]} A \)
8. \( \models_{I,t} \text{next } A \text{ iff } \models_{I[i]} A \)
9. \( \models_{I,t} O_G A \text{ iff for all } i \in \mathcal{N}, \models_{I,[i]} A \)
10. \( \models_{I,t} O_F A \text{ iff for some } i \in \mathcal{N}, \models_{I,[i]} A \)
11. \( \models_{I,t} \diamond_F A \text{ iff for some } s \in \text{List}(\mathcal{N}), \models_{I,\text{sof}} A \)
12. \( \models_{I,t} \Box_G A \text{ iff for all } s \in \text{List}(\mathcal{N}), \models_{I,\text{sof}} A \)

Since \( \diamond \) and \( \Box \) are shorthands of the sequences ‘first \( \diamond_F \)’ and ‘first \( \Box_G \)’ respectively, it is easy to see that the semantics of \( \diamond \) and \( \Box \) are as follows:

- \( \models_{I,t} \diamond A \text{ iff for some } s \in \text{List}(\mathcal{N}), \models_{I,s} A \)
- \( \models_{I,t} \Box A \text{ iff for all } s \in \text{List}(\mathcal{N}), \models_{I,s} A \)

The semantics of formulas involving the symbols \( \leftarrow, \rightarrow, \leftrightarrow \) and \( \exists \) are defined in the usual way with respect to the semantics of \( \land, \lor, \neg \) and \( \forall \).

If a formula \( A \) is true in a temporal interpretation \( I \) at all moments in time, it is said to be true in \( I \) (we write \( \models_I A \)) and \( I \) is called a **model** of \( A \).
2.3 Tautologies

In this section we present some useful tautologies of the extended BTL. In the following, the symbol $\nabla$ stands for either of first, next, $\Box_G$, $\Box_F$, $\sqcap$, $\Box_G$, $\Diamond_F$, $\Diamond$.

Temporal operator cancellation rules:

\[
\nabla (\text{first } A) \leftrightarrow (\text{first } A) \quad (1) \\
\nabla (\Box A) \leftrightarrow \Box A \quad (2) \\
\nabla (\Diamond A) \leftrightarrow \Diamond A \quad (3)
\]

Rules 2 and 3 immediately follow from rule 1.

Temporal operator distribution rules:

\[
\text{first } (A \land B) \leftrightarrow (\text{first } A) \land (\text{first } B) \quad (4) \\
\text{next}_i (A \land B) \leftrightarrow (\text{next}_i A) \land (\text{next}_i B) \quad (5) \\
\text{first } (A \lor B) \leftrightarrow (\text{first } A) \lor (\text{first } B) \quad (6) \\
\text{next}_i (A \lor B) \leftrightarrow (\text{next}_i A) \lor (\text{next}_i B) \quad (7) \\
\nabla (A \land \Diamond B) \leftrightarrow \nabla A \land \Diamond B \quad (8) \\
\nabla (A \land \text{first } B) \leftrightarrow \nabla A \land \text{first } B \quad (9) \\
\text{first } (\neg A) \leftrightarrow \neg (\text{first } A) \quad (10) \\
\text{next}_i (\neg A) \leftrightarrow \neg (\text{next}_i A) \quad (11) \\
\Box_F (\neg A) \leftrightarrow \neg (\Box_G A) \quad (12) \\
\Diamond_F (\neg A) \leftrightarrow \neg (\Box_G A) \quad (13) \\
\Box (\neg A) \leftrightarrow \neg (\Diamond A) \quad (14)
\]

Notice that although the following formulas are valid:

\[
\Box_F (A \land B) \rightarrow \Box_F A \land \Box_F B \quad (15) \\
\Box_F (A \lor B) \rightarrow \Box_F A \land \Box_F B \quad (16)
\]

the inverse implications are not valid.

Other useful tautologies

\[
\Box_G A \rightarrow \Box_G \Box_G A \quad (17) \\
\Box_F \Box_F A \rightarrow \Box_F A \quad (18)
\]

Notice that although the extended BTL formula:

\[
\Box_F \Box_F A \leftrightarrow \Box_F \Box_F A \quad (19)
\]

is a tautology, this is not true for the formula

\[
\text{next}_i \Diamond_F A \leftrightarrow \Diamond_F \text{next}_i A \quad (20)
\]
Rigidness of variables: The following tautologies state that the temporal operators first, next, and □G can “pass inside” ∀:

\[
\text{first } (\forall X)(A) \leftrightarrow (\forall X)(\text{first } A) \\
\text{next}_i (\forall X)(A) \leftrightarrow (\forall X)(\text{next}_i A) \\
\square_G(\forall X)(A) \leftrightarrow (\forall X)(\square_GA)
\]

(21) (22) (23)

However, the operator ◇_F cannot pass inside ∀ since although the implication

\[
\Box_F(\forall X)(A) \rightarrow (\forall X)(\Box_F A)
\]

(24)
is valid, the inverse implication is not valid.

The validity of formulas 21, 22 and 23 expresses the fact that variables represent data-values which are independent of time (i.e. they are rigid).

We should also note that the formulas next; next; A and next; next; A are not equivalent in general when i ≠ j.

3 Syntax of extended Cactus programs

Programs in extended Cactus extend classical Horn clauses by allowing the use of the temporal operators first, next; □_G, and □ in the head of the clauses and the use of the temporal operators first, next; □_F, and ◇ in the bodies of the clauses. The syntax of the extended Cactus programs is given formally by the following definitions:

**Definition 3.1.** A goal is defined as follows:
- A classical atom A is an open goal.
- If G is an open goal then □_F G and next; G are also open goals.
- If G_1 and G_2 are open goals then G_1 ∧ G_2 is an open goal.
- If G is an open goal then first G, and ◇ G, are fixed goals.
- An open goal or a fixed goal is a goal.
- If G_1 and G_2 are goals then G_1 ∧ G_2 is a goal.

**Definition 3.2.** A head is defined as follows:
- An atom A is an open head.
- If H is an open head then □_G H and next; H are also open heads.
- If H is an open head then first H and □ H are fixed heads.
- A head is either an open head or a fixed head.

**Definition 3.3.** An extended Cactus clause is a formula of the form:

\[H \leftarrow G\]

where H is a head and G is a (possibly empty) goal. An extended Cactus program is a set of extended Cactus clauses.
Example 3.1. The following set of clauses is an extended Cactus program which defines the relations ‘parent’, ‘sibling’, ‘uncle’ and ‘grantparent’.

\[
\begin{align*}
(1) & \quad \text{parent}(X, Y) \leftarrow \text{node}(X), \bigcirc_F \text{node}(Y). \\
(2) & \quad \text{sibling}(X, Y) \leftarrow \bigcirc_F \text{node}(X), \bigcirc_F \text{node}(Y). \\
(3) & \quad \text{uncle}(X, Z) \leftarrow \text{sibling}(X, Y), \bigcirc_F \text{parent}(Y, Z). \\
(4) & \quad \text{grantparent}(X, Y) \leftarrow \text{node}(X), \bigcirc_F \bigcirc_F \text{node}(Y). \\
(5) & \quad \text{first \ node}(\text{john}). \\
(6) & \quad \text{first \ next}_0 \ \text{node}(\text{nick}). \\
(7) & \quad \text{first \ next}_1 \ \text{node}(\text{steve}). \\
(8) & \quad \text{first \ next}_0 \ \text{next}_0 \ \text{node}(\text{edward}). \\
(9) & \quad \text{first \ next}_0 \ \text{next}_1 \ \text{node}(\text{peter}). \\
(10) & \quad \text{first \ next}_1 \ \text{next}_0 \ \text{node}(\text{bill}). \\
(11) & \quad \text{first \ next}_1 \ \text{next}_1 \ \text{node}(\text{mike}).
\end{align*}
\]

Example 3.2. In example 3.1, the relations ‘parent’, ‘sibling’, ‘uncle’ and ‘grantparent’ are time dependent in the sense that a ground instance of one of these relations may be true in a specific time point and false in some other time points. However, we often need to define time-independent variants of these relations. This can be done as follows:

\[
\begin{align*}
(1') & \quad \text{parent}(X, Y) \leftarrow \Diamond(\text{node}(X), \bigcirc_F \text{node}(Y)). \\
(2') & \quad \text{sibling}(X, Y) \leftarrow \Diamond(\bigcirc_F \text{node}(X), \bigcirc_F \text{node}(Y)). \\
(4') & \quad \text{grantparent}(X, Y) \leftarrow \Diamond(\text{node}(X), \bigcirc_F \bigcirc_F \text{node}(Y)).
\end{align*}
\]

Definition 3.4. A head \( H \) is said to be in normal form if there is no occurrence of the operators \text{first} or \bigotimes in \( H \) in the scope of any other operator. A goal is in normal form if there is no occurrence of the operators \text{first} or \Diamond in the scope of any other operator in the goal. A clause is in normal form if its head and its body are in normal form.

From the definitions 3.1, 3.2 and 3.3 we can see that extended Cactus clauses are formulas in normal form.

4 Declarative Semantics

The declarative semantics of the extended Cactus programs is defined in terms of the minimal temporal Herbrand models. For this we are based on the notion of canonical temporal reference/atom/clause, which has been initially introduced in the context of the linear time temporal logic programming language Chronolog [Org91, OWD93].
Definition 4.1. A canonical temporal reference is a temporal reference of the form first \( \text{next}_{i_1} \ldots \text{next}_{i_n} \), where \( i_1, \ldots, i_n \in \mathcal{N} \) and \( n \geq 0 \). A canonical temporal atom is a temporal atom whose temporal reference is canonical. A canonical temporal clause is a temporal clause whose temporal atoms are canonical.

Definition 4.2. A canonical temporal instance of a temporal clause \( C \) is a canonical temporal clause \( C' \) obtained as follows:
- Replace each occurrence of \( \bigcirc \) in the head of \( C \) by \( \text{next}_{i} \) for some \( i \in \mathcal{N} \).
- Replace each occurrence of \( \bigcirc_P \) in the body of \( C \) by \( \text{next}_{j} \) for some \( j \in \mathcal{N} \).
- Replace each occurrence of \( \lozenge \) in the body of \( C \) by first \( \text{next}_{i_1} \ldots \text{next}_{i_k} \) where \( k \geq 0 \) and \( i_1, \ldots, i_k \in \mathcal{N} \).
- Replace the occurrence of \( \square \) in the head of \( C \) (if any) by first \( \text{next}_{i_1} \ldots \text{next}_{i_k} \) where \( k \geq 0 \) and \( i_1, \ldots, i_k \in \mathcal{N} \).
- Apply the same canonical temporal reference to the normal form of \( C \).

The canonical temporal instance of an extended Cactus program \( P \) is the (possibly infinite) extended Cactus program \( P' \) consisting of the canonical temporal instances of the clauses in \( P \).

Intuitively, a canonical temporal instance of a temporal clause \( C \) is an instance in time of \( C \). Using the definition above we can obtain the set of all temporal instances of a given program clause.

Example 4.1. Consider the following clause taken from example 3.1:

\[
\text{grantparent}(X,Y) \leftarrow \text{node}(X), \bigcirc_P \bigcirc_P \text{node}(Y).
\]

The set of the canonical temporal instances corresponding to this clause is:

\[
\{ \text{first } \text{next}_{i_1} \ldots \text{next}_{i_n} \text{grantparent}(X,Y) \leftarrow \text{first } \text{next}_{i_1} \ldots \text{next}_{i_n} \text{node}(X), \\
\text{first } \text{next}_{i_1} \ldots \text{next}_{i_n} \text{next}_{j} \text{node}(Y) | \\
i_1, \ldots, i_n \in \mathcal{N}, n \geq 0, j \in \mathcal{N}, k \in \mathcal{N} \}
\]

The notion of canonical instance of a clause is very important since the truth value of a given clause in a temporal interpretation, can be expressed in terms of the values of its canonical instances, as the following lemma shows:

Lemma 4.1 Let \( C \) be a clause and \( I \) a temporal interpretation of extended BTL. \( \models_I C \) if and only if \( \models_I C_I \) for all canonical instances \( C_I \) of \( C \).

The domain of the temporal Herbrand interpretations of an extended Cactus program \( P \) is its temporal Herbrand universe \( U_P \), generated by constant and function symbols that appear in \( P \). The temporal Herbrand base \( B_P \) of \( P \) consists of all canonical temporal atoms generated by the predicate symbols that appear in \( P \) with terms in \( U_P \) used as arguments. A temporal Herbrand interpretation is a subset of \( B_P \).
A temporal Herbrand model of a program $P$ is a temporal Herbrand interpretation which is a model of $P$.

It can be proved that every extended Cactus program $P$ has a unique minimal temporal Herbrand model $M_P$ which consists of all ground canonical temporal atoms which are logical consequences of $P$.

5 A proof procedure for extended Cactus programs

In this section we outline a resolution-type refutation proof procedure for extended Cactus programs. For the definition of the proof procedure, called eCSLD-resolution, we use the notion of temporal context of an atom defined as follows:

**Definition 5.1.** The temporal context $tc(A)$ of a (classical) atom $A$ in a clause $C$ is the sequence of temporal operators which have $A$ in their scope.

In the following we will use the term temporal atom to refer to a classical atom preceded by its temporal context in a clause.

**Example 5.1.** The temporal contexts of the atoms in the clause:

$$\text{first next}_0 A \leftarrow \bigcirc_F B, \bigcirc_F (C, \text{next}_1 D), E.$$ are:

- $tc(A) = \text{first next}_0$
- $tc(B) = \bigcirc_F$
- $tc(C) = \bigcirc_F$
- $tc(D) = \bigcirc_F \text{next}_1$
- $tc(E) = \bigcirc$

**Definition 5.2.** If a temporal context $T$ is fixed and $Op$ is the leftmost operator in $T$ then we say that $T$ is fixed by $Op$.

In order to facilitate the definition of the proof procedure we map each context $T$ of a head atom into a context denoted by $b(T)$, called the corresponding body context of $T$, by replacing each occurrence of $\bigcirc$ in $T$ by $\bigcirc_F$ and each occurrence of $\bigcirc$ by $\bigcirc$. Moreover, by $open(T)$ we denote the temporal context obtained as follows: If $T$ is open then $open(T) = T$, otherwise $open(T)$ is obtained by removing the leftmost operator of $T$. Finally, by $open\bigcirc(T)$ we denote the temporal context obtained as follows: If $T$ is fixed by $\bigcirc$ then $open\bigcirc(T)$ is obtained by removing the leftmost operator of $T$, otherwise $open\bigcirc(T) = T$.

**Definition 5.3.** Two temporal body contexts $T_1$ and $T_2$ are said to be non-unifiable if when we traverse in parallel $open\bigcirc(T_1)$ and $open\bigcirc(T_2)$ from right to left, we find a pair of corresponding operators which is either $(\text{next}_i, \text{next}_j)$, with $i \neq j$, or one of the operator of the pair is first and the other is either of $\text{next}_i$, or $\bigcirc_F$. Two temporal body contexts $T_1$ and $T_2$ are said to be unifiable if they are not non-unifiable.
Definition 5.4.  Let $T_1$ and $T_2$ be two temporal body contexts. We say that $T'_1$ is obtained by **instantiating** $T_1$ with respect to $T_2$ if $T'_1$ is obtained from $T_1$ by replacing each occurrence of $\bigcirc_F$ in $T_1$ which correspond to an operator $\text{next}_i$ in $T_2$, by $\text{next}_i$.

Definition 5.5.  Let $T_1$ and $T_2$ be two unifiable temporal body contexts. We obtain a pair of temporal body contexts $(R_b, R_h)$ which we call a **prefix pair of $T_1$ and $T_2$**, as follows: Let $T'_1 = \text{open}(T_1)$ and $T'_2 = \text{open}(T_2)$. If $T'_1$ is empty then $^2 (R_b, R_h) = (T'_2, \epsilon)$. Otherwise if $T'_2$ is empty then $(R_b, R_h) = (\epsilon, T'_1)$. Finally, if both $T'_1$ and $T'_2$ are non-empty, the required pair is obtained by removing the rightmost operator from each one of $T'_1$ and $T'_2$ and repeating the same process.

Definition 5.6.  Let $P$ be a program in extended Cactus and $G$ be a goal clause. An $\epsilon$CSLD-derivation of $P \cup \{G\}$ consists of a (possibly infinite) sequence of temporal goals $G_0 = G, G_1, \ldots, G_n, \ldots$ a sequence $C_1, \ldots, C_n, \ldots$ of clauses of $P$ (called the input clauses), a sequence $\theta_1, \ldots, \theta_n$ of most general unifiers, and a sequence of prefix pairs $(S^{B_1}, S^{C_1}), \ldots, (S^{B_n}, S^{C_n})$ such that for all $i$, the goal $G_{i+1}$ is obtained from the goal $G_i$ as follows:

1. $T_B$ $B$ is a temporal atom ($B$ is the classical atom and $T_B$ its temporal context) in $G_i$ (called the selected atom)
2. $T_H$ $B'$ ← $\text{Body}C$ is the input clause $C_{i+1}$ (standardized apart from $G_i$)
3. $\theta_{i+1} = \text{mgu}(B, B')$ and $(S^{B}_{i+1}, S^{C}_{i+1})$ is the prefix pair of $T_B$ and $b(T_H)$.
4. The new goal $G_{i+1}$ is obtained as follows:
   
   (a) If the context $T_B$ is fixed by $\text{first}$ and the context $T_H$ is open then the new goal $G_{i+1}$ is ← $(G', \text{first} \ S^{C}_{i+1} \ \text{Body}C)\theta_{i+1}$, where $G'$ is obtained from $G_i$ by removing $B$, and instantiating $T_B$ with respect to $b(T_H)$.
   
   (b) If $T_B$ is fixed by $\text{first}$ and $T_H$ is fixed then $G_{i+1}$ is ← $(G', \Diamond \text{Body}C)\theta_{i+1}$ where $G'$ is obtained from $G_i$ by removing $B$ and instantiating $T_B$ with respect to $b(T_H)$.

   (c) If both $T_B$ and $T_H$ are open then: if $S^{B}_{i+1}$ is $\epsilon$ then $G_{i+1}$ is ← $G'\theta_{i+1}$ where $G'$ is obtained from $G_i$ by removing $B$, putting the $\text{Body}C$ in the scope of the prefix $S^{C}_{i+1}$ of $T_B$, and instantiating $T_B$ with respect to $b(T_H)$. Otherwise $G_{i+1}$ is ← $(G', \text{Body}C)\theta_{i+1}$, where $G'$ is obtained from $G_i$ by removing $B$, instantiating $T_B$ with respect to $b(T_H)$ and putting $S^{B}_{i+1}$ before the leftmost operator of $T_B$.

   (d) If $T_B$ is fixed by $\Diamond$ and $T_H$ is fixed then $G_{i+1}$ is ← $(G', \Diamond \text{Body}C)\theta_{i+1}$, where $G'$ is obtained from $G_i$ by removing $B$, instantiating $T_B$ with respect to $b(T_H)$, and replacing the $\Diamond$ in $T_B$ by $O_p \ S^{B}_{i+1}$, where $O_p$ is the leftmost operator of $b(T_H)$ (i.e. first or $\Diamond$).

---

2By $\epsilon$ we denote the empty temporal reference.
(e) If $T_B$ is fixed by $\Diamond$ and $T_H$ is open then $G_{i+1}$ is $\leftarrow (G')\theta_{i+1}$, where $G'$ is obtained from $G_i$ by removing $B$, instantiating $T_B$ with respect to $b(T_H)$, replacing $\Diamond$ in $T_B$ by $\Diamond S^B_{i+1}$ and putting $BodyC$ in the scope of the prefix $\Diamond S^C_{i+1}$ of $T_B$.

(f) If $T_B$ is open and $T_H$ is fixed by $\Diamond$ then $G_{i+1}$ is $\leftarrow (G', \Diamond BodyC)\theta_{i+1}$, where $G'$ is obtained from $G_i$ by removing $B$, instantiating $T_B$ with respect to $b(T_H)$ and putting $\Diamond S^B_{i+1}$ before the leftmost operator of $T_B$.

(g) If $T_B$ is open and $T_H$ is fixed by first then $G_{i+1}$ is $\leftarrow (G', \Diamond BodyC)\theta_{i+1}$, where $G'$ is obtained from $G_i$ by removing $B$, instantiating $T_B$ with respect to $b(T_H)$ and putting first $S^B_{i+1}$ before the leftmost operator of $T_B$.

**Definition 5.7.** A eCSLD-refutation of $P \cup \{G\}$ is a finite eCSLD-derivation of $P \cup \{G\}$ which has the empty goal clause $\Box$ as the last clause of the derivation.

**Definition 5.8.** Let $P$ be a program in Cactus and $G$ be a canonical temporal goal. A computed answer for $P \cup \{G\}$ is the substitution obtained by restricting the composition $\theta_1, \theta_2, \ldots, \theta_n$ to the variables of $G$, where $\theta_1, \theta_2, \ldots, \theta_n$, is the sequence of the most general unifiers used in a eCSLD-refutation of $P \cup \{G\}$.

**Example 5.2.** Consider the program in example 3.1. An eCSLD-refutation of the canonical temporal goal:

\[ \leftarrow \text{first uncle}(X,Z). \]

is given below (in every derivation step the selected temporal atom is the underlined one):

\[ \leftarrow \text{first uncle}(X,Z). \]

\begin{align*}
\quad & \text{using clause (3)} \\
\leftarrow & \text{first sibling}(X,Y), \text{first} \bigcirc_F \text{parent}(Y,Z) \quad \text{using clause (2)} \\
\quad & \leftarrow \text{first} \bigcirc_F \text{node}(X), \text{first} \bigcirc_F \text{node}(Y), \text{first} \bigcirc_F \text{parent}(Y,Z) \\
& (Y = \text{nick}) \quad \text{using clause (6)} \\
\leftarrow & \text{first} \bigcirc_F \text{node}(X), \text{first} \bigcirc_F \text{parent}(\text{nick},Z) \\
& \quad \text{using clause (1)} \\
\quad & \leftarrow \text{first} \bigcirc_F \text{node}(X), \text{first} \bigcirc_F \text{node}(\text{nick}), \bigcirc_F \text{node}(Z) \\
& (Z = \text{peter}) \quad \text{using clause (9)} \\
\leftarrow & \text{first} \bigcirc_F \text{node}(X), \text{first next}_0 \text{node}(\text{nick})
\end{align*}

242
using clause (6)

\[ \leftarrow \text{first } O_F \text{ node}(X) \]
\[ (X = \text{steve}) \quad \text{using clause (7)} \]

\[ \square \]

From the derivation above we conclude that first uncle(steve,peter) is a logical consequence of the program. \[ \square \]

**Example 5.3.** The program used in this example is obtained by replacing the clauses \{1, 2, 4\} in the program of example 3.1 by the clauses \{1', 2', 4'\} of example 3.2. An eCSDL-refutation of the temporal goal:

\[ \leftarrow \text{first sibling}(\text{edward},X). \]

is given below:

\[ \leftarrow \text{first sibling}(\text{edward},X). \]

\[ \quad \text{using clause (2')} \]

\[ \leftarrow \Diamond \ (O_F \text{ node}(\text{edward}), O_F \text{ node}(X)). \]

\[ \quad \text{using clause (8)} \]

\[ \leftarrow \Diamond \ \text{next}_0 O_F \text{ node}(X)). \]
\[ (X = \text{peter}) \quad \text{using clause (9)} \]

\[ \square \]

\[ \square \]

6 Discussion

In this paper, we investigate the extension of the branching-time logic programming language Cactus [RGP98] with new temporal operators. Our attempt was to extend Cactus with new temporal operators in such a way so as to retaining the SLD-resolution style of the proof system.

We show how the operators $O_F$, and $\Diamond$ can be used in the bodies of the clauses and the operators $O_G$, and $\Box$ in the heads of the clauses.

Following the work concerning the linear time logic programming language TEM- PLOG [Bau93], in which the linear time temporal operator $\Diamond$ is permitted in the bodies of the clauses, we tried to permit the use of the operator $O_F$ in the clause bodies. However this proved to be not so easy for the case of branching-time logic programming. For this reason we permit only the restricted use of this operator (i.e. the operator $\Diamond$ which is equivalent to the operator $\Diamond$ preceded by the operator first). The difficulties concerning the use of $O_F$ relate to the nondeterminism introduced when attempting to unify temporal contexts that contain $O_F$.

Finally, we believe that it would be interesting to investigate the extension of Cactus with other expressive temporal operators such as the operators proposed in the branching-time logic in [BAPM83]:

243
\(\Box_{F} A\): holds at \(t\) iff there is a path departing from \(t\) such that \(A\) is true at all time points (nodes) of this path.

\(\diamond_{\mathcal{G}} A\): holds at \(t\) iff for all paths departing from \(t\) there is some time point (node) in which \(A\) is true.

References


