EXTENSIONS OF THE BRANCHING-TIME LOGIC PROGRAMMING LANGUAGE CACTUS

MANOLIS GERGATSOULIS
Institute of Informatics and Telecommunications,
National Centre for Scientific Research (NCSR) “Demokritos”,
153 10 Aghia Paraskevi Attikis, GREECE
E-mail: manolis@iit.demokritos.gr

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 first moment in time and the temporal operator next; which refers to the i-th child of the current moment. Actually by nexti, we denote a family \{nexti | i \in N\} of next operators, each one referring to a different next moment that immediately follows the present one. In this paper we propose the extension of Cactus with new temporal operators. More specifically, we investigate the use of two variants of the operator next namely the operators \(O_F\) and \(O_G\) referring to “some next moment” and “all next moments” respectively. We also investigate the use of branching variants of the well known temporal operators \(\Box\) (always) and \(\Diamond\) (sometime).

1 Introduction

Temporal logic programming languages\(^1,^2\) provide a powerful means for the description and implementation of dynamic systems. Most temporal logic programming languages\(^1,^3,^4,^5,^6,^7\) are based on linear time temporal logics. Recently some research work has been done in the direction of developing temporal logic programming languages based on the branching notion of time. In\(^8,^9,^10\) 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 temporal operators: the operator first which refers to the beginning of time (or alternatively to the root of the tree) and the operator nexti which 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 \{nexti | i \in N\} of next operators, each one of them referring to a different next moments that immediately follows the present one.

As an example consider the non-deterministic finite automaton shown in figure 1 which accepts the regular language \(L = (01 \cup 010)^*\). The behaviour
of this automaton can be described by the following Cactus program:

\[\begin{align*}
\text{first } \text{state}(q0). \\
\text{next}_0 \text{state}(q1) &\leftarrow \text{state}(q0). \\
\text{next}_1 \text{state}(q0) &\leftarrow \text{state}(q1). \\
\text{next}_1 \text{state}(q2) &\leftarrow \text{state}(q1). \\
\text{next}_0 \text{state}(q0) &\leftarrow \text{state}(q2).
\end{align*}\]

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

\[\leftarrow \text{first } \text{next}_0 \text{next}_1 \text{next}_0 \text{state}(q0).\]

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

In\textsuperscript{10}, 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\textsuperscript{11}, a fragment of Cactus can be used as a target language for the 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 (as similar techniques have been successfully used in functional programming\textsuperscript{12}) that they can be implemented efficiently.

In this paper, we propose the extension of Cactus with more expressive temporal operators. More specifically, we investigate the extension of Cactus with two variants of the operator \texttt{next} namely the operators \(\bigcirc_F\) and \(\bigcirc_G\) referring to "some next moment" and "all next moments" respectively. We
also investigate the use of branching variants of the well known temporal operators $\square$ (always) and $\Diamond$ (sometime). In particular, we use the operators $\Diamond_F$ and $\Box_G$ whose intuitive meaning is “some moment in some future of the first moment in time” and “all moments in all futures of the first moment in time” respectively.

We show how we can extend the syntax of Cactus by allowing the use of $\bigcirc_G$, and $\Box_G$ in the clause heads and the use of $\bigcirc_F$, and $\Diamond_F$ in the clause bodies, while retaining the SLD-resolution style of the proof procedure.

The rest of the paper is organized as follows: The temporal logic on which the extended Cactus is based is presented in section 2. The syntax of the extended Cactus programs is presented in section 3. The declarative semantics of the extended Cactus programs is given in section 4. In section 5, a SLD-resolution like roof procedure for extended Cactus programs is outlined. Finally, section 6 gives the concluding remarks.

2 The branching-time temporal logic

2.1 Temporal operators

In $^8,^9,^10$ it is presented a simple 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 is modeled by the set $List(\mathcal{N})$ of lists of natural numbers $\mathcal{N}$. 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 $t$. In BTL there are two temporal operators namely the operators $\textit{first}$ and $\textit{next}_i$, $i \in \mathcal{N}$. The operator $\textit{first}$ refers to the first moment in time, while the operator $\textit{next}_i$ refers to the $i$-th child of the current moment in time.

Branching-time logics with richer sets of temporal operators than those of BTL have been proposed in the literature$^{13,14}$. In this section, we consider the extension of BTL with the temporal operators $\bigcirc_G$, $\bigcirc_F$, $\Diamond_F$ and $\Box_G$. The intuitive meaning of these temporal operators is as follows:

$\bigcirc_G A$: holds at $t$ iff the formula $A$ is true at every immediate successor of $t$.

$\bigcirc_F A$: 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 nodes of the subtree rooted at $t$. 

The syntax of the formulae of the extended BTL extends the syntax of first order logic with four new formation rules: If $A$ is a formula, so are $\text{first} \ A$, $\text{next} \ A$, $\bigcirc_F A$, $\bigcirc_G A$, $\Diamond_F A$, and $\Box_G A$.

In the following we will also use the operators $\Diamond_F$ and $\Box_G$, as shorthands for the sequences of operators “$\text{first} \Diamond_F$” and “$\text{first} \Box_G$” respectively.

Notice that the above operators are similar to the operators of the branching-time logic proposed by Ben-Ari, Pnueli and Manna in\textsuperscript{13}. In particular, the operators $\Box_G$, $\bigcirc_G$, and $\Diamond_F$ are denoted in\textsuperscript{13} as $\forall G$, $\forall X$, and $\exists F$ respectively. In\textsuperscript{13} two more operators are considered namely the operators $\forall F$ and $\exists G$ (in our notation $\Box_F$ and $\Diamond_G$ respectively). The intuitive meaning of these operators is:

$\Diamond_G A$: holds at $t$ iff for all paths departing from $t$ there is a time point (node) in which $A$ is true.

$\Box_F A$: holds at $t$ iff there is a path departing from $t$ such that $A$ is true at all time points (nodes) of this path.

We do not include these operators in our extended BTL.

2.2 Semantics of the formulas of the extended BTL

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

**Definition 2.1.** A branching temporal interpretation (or simply 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 constant; for each $n$-ary function symbol, an element of $[D^n \rightarrow D]$; and for each $n$-ary predicate symbol, an element of $[\text{List}(N) \rightarrow 2^D]$.

In the following definition, the satisfaction relation $|=\exists$ is defined in terms of temporal interpretations. $|=I,t\ A$ denotes that the formula $A$ is true at the moment $t$ in the temporal interpretation $I$. Finally, by $s \circ t$ we denote the list obtained by concatenating the lists $s$ and $t$.

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

1. If $\mathbf{e}(e_0, \ldots, e_{n-1})$ is a term, then $I(\mathbf{e}(e_0, \ldots, e_{n-1})) = I(\mathbf{e})(I(e_0), \ldots, I(e_{n-1}))$.

2. For any $n$-ary predicate symbol $p$ and terms $e_0, \ldots, e_{n-1},$

   $|=I,t\ p(e_0, \ldots, e_{n-1})$ iff $\langle I(e_0), \ldots, I(e_{n-1}) \rangle \in I(p)(t)$
3. $\models_{I,t} \neg A$ iff it is not the case that $\models_{I,t} A$

4. $\models_{I,t} A \lor B$ iff $\models_{I,t} A$ or $\models_{I,t} B$

5. $\models_{I,t} (\forall x) A$ iff $\models_{I[d/x],t} A$ for all $d \in D$ where the interpretation $I[d/x]$ is the same as $I$ except that the variable $x$ is assigned the value $d$.

6. $\models_{I,t} \text{first} A$ iff $\models_{I,[1]} A$

7. $\models_{I,t} \text{next}_i A$ iff $\models_{I,[i,t]} A$

8. $\models_{I,t} O_G A$ iff for all $i \in \mathcal{N}, \models_{I,[i,t]} A$

9. $\models_{I,t} O_F A$ iff for some $i \in \mathcal{N}, \models_{I,[i,t]} A$

10. $\models_{I,t} \Diamond_F A$ iff for some $s \in \text{List}(\mathcal{N}), \models_{I,s,ot} A$

11. $\models_{I,t} \Box_G A$ iff for all $s \in \text{List}(\mathcal{N}), \models_{I,s,ot} A$

Since $\Diamond_F$ and $\Box_G$ are shorthands of the sequences "first $\Diamond_F$" and "first $\Box_G$" respectively, it is easy to see that:

1. $\models_{I,t} \Diamond_F A$ iff for some $s \in \text{List}(\mathcal{N}), \models_{I,s} A$

2. $\models_{I,t} \Box_G A$ iff for all $s \in \text{List}(\mathcal{N}), \models_{I,s} A$

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

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

2.3 Axioms

In this section we present some axioms of the extended BTL, which will be used in this paper. In the following, the symbol $\nabla$ stands for either of $\text{first}$, $\text{next}_i$, $O_G$, $O_F$, $\Box_G$, and $\Diamond_F$. We do not present axioms for the operators $\Diamond_F$ and $\Box_G$ since these axioms immediately follow from the axioms of the operators composing the operators $\Diamond_F$ and $\Box_G$.

Temporal operator cancellation rules:

$$\nabla(\text{first} A) \leftrightarrow (\text{first} A) \quad (1)$$
Temporal operator distribution rules:

\[
\begin{align*}
\text{first}(A \land B) & \leftrightarrow (\text{first}A) \land (\text{first} B) \\
\text{next}_1(A \land B) & \leftrightarrow (\text{next}_1A) \land (\text{next}_1 B) \\
\nabla(A \land \text{first} B) & \leftrightarrow \nabla A \land \text{first} B \\
\text{first}(\neg A) & \leftrightarrow \neg(\text{first} A) \\
\text{next}_1(\neg A) & \leftrightarrow \neg(\text{next}_1 A) \\
\Box_F(\neg A) & \leftrightarrow \neg(\Box_G A) \\
\Diamond_F(\neg A) & \leftrightarrow \neg(\Box_G A)
\end{align*}
\]

Notice that although the following formulas are valid:

\[
\begin{align*}
\Diamond_F(A \land B) & \rightarrow \Diamond_F A \land \Diamond_F B \\
\Box_F(A \land B) & \rightarrow \Box_F A \land \Box_F B
\end{align*}
\]
the inverse implications are not valid.

**Other useful axioms** The following formulas are valid:

\[
\begin{align*}
\Box_G A & \rightarrow \Box_G \Box_G A \\
\Diamond_F \Box_F A & \rightarrow \Diamond_F A
\end{align*}
\]

Notice also that although the extended BTL formula:

\[
\Box_F \Diamond_F A \leftrightarrow \Diamond_F \Box_F A
\]

is a tautology, this is not true for the formula.

\[
\text{next}_1 \Diamond_F A \leftrightarrow \Diamond_F \text{next}_1 A
\]

**Rigidity of variables:** The following tautologies state that the temporal operators first, next\(_1\) and \(\Box_G\) can “pass inside” \(\forall\):

\[
\begin{align*}
\text{first}(\forall X)(A) & \leftrightarrow (\forall X)(\text{first} A) \\
\text{next}_1(\forall X)(A) & \leftrightarrow (\forall X)(\text{next}_1 A) \\
\Box_G(\forall X)(A) & \leftrightarrow (\forall X)(\Box_G A)
\end{align*}
\]

However, the operator \(\Diamond_F\) cannot pass inside \(\forall\) since although the implication

\[
\Diamond_F(\forall X)(A) \rightarrow (\forall X)(\Diamond_F A)
\]
is valid, the inverse implication is not.

The validity of formulas 15, 16 and 17 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\textsubscript{i} next\textsubscript{j} A and next\textsubscript{j} next\textsubscript{i} A are not equivalent in general when \( i \neq j \).

3 Syntax of extended Cactus programs

Programs in extended Cactus extend classical Horn clause programs by allowing the temporal operators first, next\textsubscript{i}, \( \bigcirc\textsubscript{G} \), and \( \square\textsubscript{G} \) to be used in the heads of the clauses and the temporal operators first, next\textsubscript{i}, \( \bigcirc\textsubscript{F} \), and \( \diamond\textsubscript{F} \) to be used 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 \( \bigcirc\textsubscript{F} G \) and next\textsubscript{i} \( G \) are also open goals.
- If \( G \) is an open goal then first \( G \), and \( \diamond\textsubscript{F} G \), are fixed goals.
- An open goal or a fixed goal is a goal.
- If \( G_1 \) and \( G_2 \) are goals then \( G_1 \land G_2 \) is a goal. \( G_1 \land G_2 \) is a fixed goal if both \( G_1 \) and \( G_2 \) are fixed goals otherwise \( G_1 \land G_2 \) is an open goal.

Definition 3.2. A head is defined as follows:
- An atom \( A \) is an open head.
- If \( H \) is an open head then \( \bigcirc\textsubscript{G} H \) and next\textsubscript{i} \( H \) are also open heads.
- If \( H \) is an open head then first \( H \) and \( \square\textsubscript{G} H \) 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 "grandparent".

\begin{enumerate}
\item parent\textsubscript{(X,Y)} \leftarrow node\textsubscript{(X)}, \bigcirc\textsubscript{F} node\textsubscript{(Y)}.
\item sibling\textsubscript{(X,Y)} \leftarrow \bigcirc\textsubscript{F} node\textsubscript{(X)}, \bigcirc\textsubscript{F} node\textsubscript{(Y)}.
\item uncle\textsubscript{(X,Z)} \leftarrow sibling\textsubscript{(X,Y)}, \bigcirc\textsubscript{F} parent\textsubscript{(Y,Z)}.
\item grandparent\textsubscript{(X,Y)} \leftarrow node\textsubscript{(X)}, \bigcirc\textsubscript{F} \bigcirc\textsubscript{F} node\textsubscript{(Y)}.
\end{enumerate}
(5) first node(john).
(6) first next\(_0\) node(nick).
(7) first next\(_1\) node(steve).
(8) first next\(_0\) next\(_0\) node(edward).
(9) first next\(_0\) next\(_1\) node(peter).
(10) first next\(_1\) next\(_0\) node(bill).
(11) first next\(_1\) next\(_1\) node(mike).

Example 3.2. Consider the following clauses redefining the relations "parent", "sibling" and "grandparent" of example 3.1:

\[
(1') \quad \text{parent}(X, Y) \leftarrow \diamond_F (\text{node}(X), \bigcirc_F \text{node}(Y)).
\]

\[
(2') \quad \text{sibling}(X, Y) \leftarrow \diamond_F (\bigcirc_F \text{node}(X), \bigcirc_F \text{node}(Y)).
\]

\[
(4') \quad \text{grandparent}(X, Y) \leftarrow \diamond_F (\text{node}(X), \bigcirc_F \bigcirc_F \text{node}(Y)).
\]

The difference between these definitions and the corresponding definitions of example 3.1 is that in example 3.1 an instance of one of these relations may be true in a specific time point and false in some other time points of the time tree. On the other hand, the relations "parent", "sibling" and "grandparent" defined by the clauses 1', 2', 4' are "time-independent" in the sense that if an instance of one of these relations is true in a specific time point then it is true in all time points of the time tree.

Definition 3.4. A head \( H \) is said to be in normal form if there are no occurrences of the operators first and \( \Box_G \) in \( H \) in the scope of any other operator. A goal is in normal form if there are no occurrences of the operators first or \( \diamond_F \) 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.

In the rest of this paper we suppose that all clauses of extended Cactus programs 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 context/atom/clause, initially introduced in the context of the linear time temporal logic programming language Chronolog\(^7,^3\).
Definition 4.1. The sequence of the temporal operators which have an atom in their scope is said to be the temporal context of that atom. The length of a temporal context $T$ (denoted by $\text{length}(T)$) is the number of temporal operators in $T$. A temporal atom is a classical atom preceded by its temporal context. A canonical temporal context is a temporal context of the form $\text{first next}_{i_1} \ldots \text{next}_{i_n}$, where $i_1, \ldots, i_n \in \mathcal{N}$ and $n \geq 0$. A canonical temporal atom is a temporal atom whose temporal context 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 $\Diamond_G$ in the head of $C$ by $\text{next}_i$ for some $i \in \mathcal{N}$.
- Replace each occurrence of $\Box_F$ in the body of $C$ by $\text{next}_j$ for some $j \in \mathcal{N}$.
- Replace each occurrence of $\Diamond_F$ in the body of $C$ by $\text{first next}_{i_1} \ldots \text{next}_{i_n}$ where $n \geq 0$ and $i_1, \ldots, i_n \in \mathcal{N}$.
- Replace the occurrence of $\Box_G$ in the head of $C$ (if any) by $\text{first next}_{i_1} \ldots \text{next}_{i_m}$ where $m \geq 0$ and $i_1, \ldots, i_m \in \mathcal{N}$.
- Apply the same canonical temporal context 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 all canonical temporal instances of all clauses in $P$.

Intuitively, a canonical temporal instance of a temporal clause $C$ is an instance in time of $C$. Using the definition 4.2 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:

$$\text{grandparent}(X, Y) \leftarrow \text{node}(X), \bigcirc_F \bigcirc_F \text{node}(Y).$$

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

$$\{\text{first next}_{i_1} \ldots \text{next}_{i_n} \text{grandparent}(X, Y) \leftarrow \text{first next}_{i_1} \ldots \text{next}_{i_n} \text{node}(X), \quad \text{first next}_{i_1} \ldots \text{next}_{i_n} \text{next}_j \text{next}_k \text{node}(Y) \mid i_1, \ldots, i_n \in \mathcal{N}, n \geq 0, j \in \mathcal{N}, k \in \mathcal{N}\}$$

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 \( \models_C C_t \) for all canonical instances \( C_t \) of \( C \).

The domain of the temporal Herbrand interpretations of a 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 predicates of \( P \) 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 proof procedure for extended Cactus programs, called ECSLD-resolution (Extended Cactus SLD-resolution). In the following by \( tc(A) \) we denote the temporal context of an occurrence of an atom \( A \) in a temporal clause.

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

\[
\text{first next}_0 A \leftarrow \Box_F B, \Diamond_F (\Box_F (C, \text{next}_1 D), E).
\]

are:

\[
\begin{align*}
tc(A) &= \text{first next}_0 \\
tc(B) &= \Box_F \\
tc(C) &= \Diamond_F \Box_F \\
tc(D) &= \Diamond_F \Box_F \text{next}_1 \\
tc(E) &= \Diamond_F
\end{align*}
\]

\[\Box \]

Definition 5.1. If a temporal context \( T \) is fixed and \( Op \) is the leftmost operator in \( T \) then we say that \( T \) is fixed by \( Op \).

Notice that, since all program clauses are supposed to be in normal form, the operator \( Op \) in the above definition is one of \( \text{first}, \Diamond_F \) and \( \Box_G \).

In order to facilitate the definition of the proof procedure we map the 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 \( \Box_G \) in \( T \) by \( \Box_F \) and each occurrence of \( \Box_G \) by \( \Diamond_F \). Moreover, by \( \text{open}(T) \) we denote the temporal context obtained as follows: If \( T \) is open then \( \text{open}(T) = T \), other-
wise \( \text{open}(T) \) is obtained by removing the leftmost operator of \( T \). Finally, by \( \text{open}\hat{\Diamond}_F(T) \) we denote the temporal context obtained as follows: If \( T \) is fixed by \( \hat{\Diamond}_F \), then \( \text{open}\hat{\Diamond}_F(T) \) is obtained by removing the leftmost operator of \( T \), otherwise \( \text{open}\hat{\Diamond}_F(T) = T \).

**Definition 5.2.** Two temporal body contexts \( T_1 \) and \( T_2 \) are said to be non-unifiable if when we traverse in parallel \( \text{open}\hat{\Diamond}_F(T_1) \) and \( \text{open}\hat{\Diamond}_F(T_2) \) 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 operators is \textbf{first} and the other is either \textbf{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 5.3.** Let \( T_1 \) and \( T_2 \) be two unifiable temporal body contexts. We say that \( T_1' \) is obtained by \textit{instantiating} \( T_1 \) \textit{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 \textbf{next}_i in \( T_2 \), by \textbf{next}_i.

**Definition 5.4.** Let \( T_1 \) and \( T_2 \) be two unifiable temporal body contexts, \( T_1' = \text{open}(T_1) \) and \( T_2' = \text{open}(T_2) \). Let \( m \) be the minimum of \text{length}(T_1') and \text{length}(T_2'). We obtain a pair of temporal body contexts \((R_b, R_h)\) which we call a \textit{prefix pair} of \( T_1 \) and \( T_2 \), as follows: We discurd the \( m \) rightmost operators of each one of \( T_1' \) and \( T_2' \) obtaining \( T_1'' \) and \( T_2'' \) respectively. Then \((R_b, R_h) = (T_2'', T_1'')\).

Notice that at least one of \( R_b, R_h \) in the above definition is the empty temporal context \( \epsilon \).

**Definition 5.5.** Let \( P \) be a program in extended Cactus and \( G \) be a (canonical) goal clause. An ECSLD-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 \textit{input clauses}), a sequence \( \theta_1, \ldots, \theta_n \) of most general unifiers, and a sequence of prefix pairs \((S_1^B, S_1^C), \ldots, (S_n^B, S_n^C)\), 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 \textit{selected} atom)

2. \( T_H \) \( B' \leftarrow \text{Body}C \) is the input clause \( C_{i+1} \) (standardized apart from \( G_i \))

3. \( \theta_{i+1} = \text{mgu}(B, B') \) and \((S_{i+1}^B, S_{i+1}^C)\) is the prefix pair of \( T_B \) and \textit{b}(T_H).

4. The new goal \( G_{i+1} \) is obtained as follows:
(a) If \( T_B \) is fixed by \textit{first} and \( T_H \) is open then the new goal \( G_{i+1} \) is \( \left< (G', \textit{first } S_{i+1}^C \textit{ BodyC}) \theta_{i+1} \right> \), 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 \textit{first} and \( T_H \) is fixed then \( G_{i+1} \) is \( \left< (G', \Diamond_F \textit{ BodyC}) \theta_{i+1} \right> \) 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 \( S_{i+1}^B \) is \( \epsilon \) then \( G_{i+1} \) is \( \left< G' \theta_{i+1} \right> \) where \( G' \) is obtained from \( G_i \) by removing \( B \), putting the \textit{BodyC} in the scope of the prefix \( S_{i+1}^C \) of \( T_B \), and instantiating \( T_B \) with respect to \( b(T_H) \). Otherwise \( G_{i+1} \) is \( \left< (G', \textit{ BodyC}) \theta_{i+1} \right> \), where \( G' \) is obtained from \( G_i \) by removing \( B \), instantiating \( T_B \) with respect to \( b(T_H) \) and putting \( S_{i+1}^B \) before the leftmost operator of \( T_B \).

(d) If \( T_B \) is fixed by \( \Diamond_F \) and \( T_H \) is fixed then \( G_{i+1} \) is \( \left< (G', \Diamond_F \textit{ BodyC}) \theta_{i+1} \right> \), where \( G' \) is obtained from \( G_i \) by removing \( B \), instantiating \( T_B \) with respect to \( b(T_H) \), and replacing the \( \Diamond_F \) in \( T_B \) by \( O_p \ S_{i+1}^B \), where \( O_p \) is the leftmost operator of \( b(T_H) \) (i.e. \textit{first} or \( \Diamond_F \)).

(e) If \( T_B \) is fixed by \( \Diamond_F \) and \( T_H \) is open then \( G_{i+1} \) is \( \left< (G') \theta_{i+1} \right> \), where \( G' \) is obtained from \( G_i \) by removing \( B \), instantiating \( T_B \) with respect to \( b(T_H) \), replacing \( \Diamond_F \) in \( T_B \) by \( ' \Diamond_F S_{i+1}^B ' \) and putting \( \textit{BodyC} \) in the scope of the prefix \( \Diamond_F S_{i+1}^C \) of \( T_B \).

(f) If \( T_B \) is open and \( T_H \) is fixed by \( \Diamond_F \) then \( G_{i+1} \) is \( \left< (G', \Diamond_F \textit{ BodyC}) \theta_{i+1} \right> \), where \( G' \) is obtained from \( G_i \) by removing \( B \), instantiating \( T_B \) with respect to \( b(T_H) \) and putting \( ' \Diamond_F S_{i+1}^B ' \) before the leftmost operator of \( T_B \).

(g) If \( T_B \) is open and \( T_H \) is fixed by \textit{first} then \( G_{i+1} \) is \( \left< (G', \Diamond_F \textit{ BodyC}) \theta_{i+1} \right> \), where \( G' \) is obtained from \( G_i \) by removing \( B \), instantiating \( T_B \) with respect to \( b(T_H) \) and putting \( ' \textit{first } S_{i+1}^B ' \) before the leftmost operator of \( T_B \).

**Definition 5.6.** Let \( P \) be a program in extended Cactus and \( G \) be a (canonical) goal clause. An \textit{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 5.7.** Let \( P \) be a program and \( G \) be a canonical temporal goal. A \textit{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):

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

using clause (3)

\[ \leftarrow \text{first sibling}(X,Y), \text{first } \bigcirc_F \text{ parent}(Y,Z) \]

using clause (2)

\[ \leftarrow \text{first } \bigcirc_F \text{ node}(X), \text{first } \bigcirc_F \text{ node}(Y), \]

\[ \text{first } \bigcirc_F \text{ parent}(Y,Z) \]

($Y = \text{nick}$) using clause (6)

\[ \leftarrow \text{first } \bigcirc_F \text{ node}(X), \text{first } \bigcirc_F \text{ parent}(\text{nick},Z) \]

using clause (1)

\[ \leftarrow \text{first } \bigcirc_F \text{ node}(X), \text{first } \bigcirc_F (\text{node}(\text{nick}), \bigcirc_F \text{ node}(Z)) \]

($Z = \text{peter}$) using clause (9)

\[ \leftarrow \text{first } \bigcirc_F \text{ node}(X), \text{first } \bignext_0 \text{ node}(\text{nick}) \]

using clause (6)

\[ \leftarrow \text{first } \bigcirc_F \text{ node}(X) \]

($X = \text{steve}$) using clause (7)

\[ \boxdot \]

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

**Example 5.3.** Consider the program 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 ECSLD-refutation of the goal:
\begin{itemize}
\item \texttt{first sibling(edward,X)}.
\end{itemize}

is given below:

\begin{itemize}
\item \texttt{first sibling(edward,X).}
\end{itemize}

using clause (2')

\begin{itemize}
\item \(\Diamond_F (O_F \text{ node(edward)}, \bigcirc_F \text{ node(X)})\).
\end{itemize}

using clause (8)

\begin{itemize}
\item \(\Diamond_F \text{ next}_0 (O_F \text{ node(X)})\).
\end{itemize}

\(X = \text{peter}\) using clause (9)

In the presentation of the proof procedure we have supposed that the top-
level goal is canonical. However, the proof procedure can be extended for the
case of non-canonical top level goals. In this case, besides the substitutions of
the variables of the goal, we may also find a sequence of temporal operators
which should be applied to the goal in order to be logical consequence of the
program.

\section{Discussion}

In this paper, we investigate the extension of the branching-time logic pro-
gramming language Cactus\textsuperscript{10} with new, more expressive, temporal operators.
In particular, we are interested in extensions of Cactus for which we can de-
dine SLD-resolution style refutation proof procedures. We show how we can
extend Cactus to allow also the use of the operators \(O_F\), and \(\Diamond_F\) in the bodies
of the clauses and the operators \(O_G\), and \(\Diamond_G\) in the heads of the clauses.

Following the work concerning the linear time logic programming language
TEMPLOG\textsuperscript{4}, in which the linear time temporal operator \(\Diamond\) is allowed in the
bodies of the clauses, we examined the possibility to allow the unlimited use
of the operator \(\Diamond_F\) in the clause bodies. However, this seems to be a difficult
task for the case of branching-time logic programming. The reason is that
while in the linear case the formula:

\[ \text{next } \Diamond A \leftrightarrow \Diamond \text{ next } A \]

is valid and thus it is always possible to move \(\Diamond\) in front of a temporal context,
in banching time logic BTL the formula:
\text{next}_i \Diamond_F A \leftrightarrow \Diamond_F \text{next}_i A

is not valid and thus $\Diamond_F$ cannot always be pulled to the front of a temporal context. The existence of $\Diamond_F$ in the context of a selected atom, in places other than the leftmost one, introduces nondeterminism when attempting to unify temporal contexts that contain $\Diamond_F$. Because of this difficulty, we only allow the use of a restricted form of this operator (i.e. the operator $\Diamond_F$ which is equivalent to the operator $\Diamond$ preceded by the operator first).

An interesting topic for future work is to investigate the possibility to allow also the use of the operators $\Diamond_G$ and $\Box_F$ in the syntax of the extended Cactus.

Acknowledgements

I would like to thank C. Nomikos for his helpful comments.

References


