TEMPORAL DISJUNCTIVE LOGIC PROGRAMMING

M Gergatsoulis, P. Rondogiannis, T. Panayiotopoulos

15-97

Preprint no. 15-97/1997

Department of Computer Science
University of Ioannina
451 10 Ioannina, Greece

Temporal Disjunctive Logic Programming

M. Gergatsoulis[1], **P. Rondogiannis**[2], **T. Panayiotopoulos**[3]

[1] Inst. of Informatics & Telecom., N.C.S.R. 'Demokritos',
153 10 A. Paraskevi Attikis, Greece
e_mail: manolis@iit.nrcps.ariadne-t.gr

[2] Dept. of Computer Science, University of Ioannina,
P.O. BOX 1186, 45110 Ioannina, Greece,
e_mail: prondo@zeus.cs.uoi.gr

[3] Dept. of Informatics, University of Piraeus
80 Karaoli & Dimitriou Str., 18534 Piraeus, Greece
e-mail : themisp@unipi.gr

Abstract

In this paper we introduce *temporal disjunctive logic programming* in order to combine the programming paradigms of temporal and disjunctive logic programming. For this, we define a simple temporal disjunctive logic programming language, called *Disjunctive Chronolog*. Disjunctive Chronolog is capable of expressing dynamic behaviour as well as uncertainty, two notions that are very common in a variety of real systems. We present the minimal temporal model semantics, temporal model state semantics and fixpoint semantics for the new programming language and demonstrate their equivalence. We also show how the proof procedures developed for disjunctive logic programs, can be easily extended to also apply to Disjunctive Chronolog programs.

Keywords: Logic Programming, Temporal Logic Programming, Disjunctive Logic Programming, Semantics, Proof Procedures.

1 Introduction

Temporal logic programming [OM94, FO95, Org91, Brz91, Hry93, Bau93, RGP97b, RGP97a] has been widely used as a means for describing systems that are inherently *dynamic*. For example, consider the following Chronolog [Wad88] program simulating the operation of the traffic lights:

```
first light(green).
next light(amber) ← light(green).
next light(red) ← light(amber).
next light(green) ← light(red).
```

On the other hand, disjunctive logic programming [MRL91, LR91, LMR92] was introduced as a formalism for expressing *uncertainty*:

```
plays(john,soccer) ∨ plays(john,basketball).
sportsman(X) ← plays(X,Y), sport(Y).
sport(soccer).
sport(basketball).
```

From this program, it can be easily extracted as a conclusion that `john` is a `sportsman`, even though there is no exact knowledge about the kind of sports he is participating in.

There are many systems however in which dynamic behaviour and uncertainty coexist. Real-time and reactive systems, expert systems, temporal or generally multidimensional databases are such examples.

It is therefore natural to ask whether there exists a single logic programming paradigm which amalgamates the above two notions in a semantically lucid way.

In this paper we introduce *temporal disjunctive logic programming* in order to combine the ideas of both temporal logic programming and disjunctive logic programming. For this, we introduce the temporal disjunctive logic programming language, called *Disjunctive Chronolog* [GRP96]. Our starting point is the temporal language Chronolog, proposed by W. W. Wadge [Wad88] whose semantics have been systematically developed by M. Orgun [Org91, OW92b, OWD93]. The new formalism that we propose, is capable of expressing time related uncertainty of different forms:

Event uncertainty. Consider for example the curriculum of a computer science department, which requires from students to have taken Discrete Mathematics before they register to either of the Data Structures or the Algorithms course. The above restriction could be expressed in Disjunctive Chronolog as:

```
first course(discrete_math).
next course(algorithms) ∨ next course(data_structures) ←
                course(discrete_math).
```

The event uncertainty results from the fact that after a student has taken Discrete Mathematics he can choose to enroll in the Algorithms course, in the Data Structures course, or in both. The particular event that will take place is not known in advance.

Time uncertainty. Consider for example the following program:

```
first visit(george,greece) ∨ first next visit(george,greece).
have_good_time(X) ← visit(X,greece).
```

The time uncertainty is expressed by the first clause which says that "George is either going to visit Greece this or next year (or both)".

The semantics of Disjunctive Chronolog extend the semantics of both Chronolog [Org91, OW92a] and Disjunctive Logic programming [LR91, LMR92]. More specifically, we define minimal model semantics, model state semantics and fixpoint semantics for Disjunctive Chronolog programs and show their equivalence. Moreover, we investigate proof procedures for Disjunctive Chronolog programs, and show that proof procedures developed for disjunctive logic programs [LMR89, MRL91, LR91, LMR92], can be easily extended to apply also to Disjunctive Chronolog programs. For the development of both the semantics and the proof procedures for Disjunctive Chronolog, we are based on the notion of *canonical program clauses* [Org91, OW92a].

The rest of this paper is organized as follows. Section 2, describes the underlying temporal logic (TL) of Disjunctive Chronolog. Section 3 introduces the syntax of Disjunctive Chronolog. In section 4, we present the declarative semantics of Disjunctive Chronolog programs. More specifically, after giving some background definitions in subsection 4.1, we present the minimal temporal Herbrand model semantics in subsection 4.2, the temporal model state semantics in subsection 4.3, and the fixpoint semantics in subsection 4.4. In section 5, we investigate proof procedures for Disjunctive Chronolog programs. Finally, in section 6, we conclude the paper and suggest some topics for future work.

2 Background

The temporal logic (TL) of Disjunctive Chronolog is that of the logic programming language Chronolog($\mathcal{Z}$) developed by W. W. Wadge and M. Orgun [Wad88, OW92a, OWD93, OW93, OW92b]. In this section we give some useful background definitions regarding temporal logic which are adopted from [Org91, OW92a, OWD93].

2.1 The temporal logic of Disjunctive Chronolog

The temporal logic (TL) of Disjunctive Chronolog is based on linear time with unbounded past and future. The set of moments in time is represented by the set $\mathcal{Z}$ of integers. TL has three temporal operators `first`, `next`, and `prev`. The operator `first` is used to express the first moment in time, while `next` refers to the next moment in time, and `prev` to the previous moment in time. The syntax of the formulas of temporal logic is an extension of the syntax of first-order logic with three formation rules: if A is a formula, then so are `first` A, `next` A and, `prev` A.

The semantics of formulas of TL are given through temporal interpretations.

Definition 2.1 (*Temporal interpretation*). A *temporal interpretation* I of the temporal logic TL 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 $[\mathcal{Z} \to 2^{D^n}]$.

Interpretations are extended to all elements of the language by a satisfaction relation $\models$ which is defined in terms of temporal interpretations. In the following definition $\models_{I,t} A$ denotes that a formula A is true at a moment t in some temporal interpretation I.

Definition 2.2 (*Semantics of TL*). The semantics of the elements of the temporal logic TL are given inductively as follows:

1. If $\mathtt{f}(e_0, \ldots, e_{n-1})$ is a term, then $I(\mathtt{f}(e_0, \ldots, e_{n-1})) = I(\mathtt{f})(I(e_0), \ldots, I(e_{n-1}))$.
2. For any n-ary predicate symbol $\mathtt{p}$ and terms $e_0, \ldots, e_{n-1}$,
   $\models_{I,t} \mathtt{p}(e_0, \ldots, e_{n-1})$ *iff* $< I(e_0), \ldots, I(e_{n-1}) > \in I(\mathtt{p})(t)$
3. $\models_{I,t} \neg A$ *iff it is not the case that* $\models_{I,t} A$
4. $\models_{I,t} A \wedge B$ *iff* $\models_{I,t} A$ *and* $\models_{I,t} B$
5. $\models_{I,t} A \vee B$ *iff* $\models_{I,t} A$ *or* $\models_{I,t} B$
6. $\models_{I,t} (\forall x) A$ *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.
7. $\models_{I,t}$ `first` A *iff* $\models_{I,0} A$
8. $\models_{I,t}$ `prev` A *iff* $\models_{I,t-1} A$
9. $\models_{I,t}$ `next` A *iff* $\models_{I,t+1} A$

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. If a formula A is true in all temporal interpretations at all moments in time, we say that A is *valid* (we write $\models A$).

2.2 Tautologies

In this section we present some useful tautologies that are valid formulas of the logic TL. The symbol ∇ stands for any of `first`, `next`, and `prev`.

1. **Temporal operator cancellation rules:**
   (a) $\nabla(\mathtt{first}\ A) \leftrightarrow (\mathtt{first}\ A)$
   (b) $\mathtt{next}\ \mathtt{prev}\ A \leftrightarrow A$
   (c) $\mathtt{prev}\ \mathtt{next}\ A \leftrightarrow A$

2. **Temporal operator distribution rules:**

   (a) $\nabla(\neg A) \leftrightarrow \neg(\nabla A)$

   (b) $\nabla(A \wedge B) \leftrightarrow (\nabla A) \wedge (\nabla B)$

   (c) $\nabla(A \vee B) \leftrightarrow (\nabla A) \vee (\nabla B)$

3. **Rigidness of variables:**

   (a) $\nabla(\forall X)(A) \leftrightarrow (\forall X)(\nabla A)$

The proof of correctness of the above tautologies is straightforward [Org91].

3 Syntax of Disjunctive Chronolog programs

The syntax of Disjunctive Chronolog extends the syntax of disjunctive logic programs [LMR92] by permitting temporal operators to be applied to the atomic formulas (atoms) of the clause. A *temporal atom* is an atomic formula with a number (possibly 0) of applications of temporal operators. The sequence of temporal operators applied to an atom is called the *temporal reference* of that atom. A *temporal literal* is a temporal atom or the negation of a temporal atom. A *temporal disjunctive clause* is a clause of the form:

$$H_1 \vee H_2 \vee \ldots \vee H_n \leftarrow B_1, \ldots, B_m$$

where $H_1, \ldots, H_n, B_1, \ldots, B_m$ are temporal atoms, $n \geq 1$ and $m \geq 0$. The left hand side is called the *head* of the clause while the right hand side is called the *body* of the clause. In the body, the comma stands for the conjunction operator '$\wedge$'. If $n = 1$ then, the clause is said to be a *definite temporal clause*. If $m = 0$ then the clause is said to be a *positive temporal disjunctive clause*. A *Disjunctive Chronolog program* is a finite set of *temporal disjunctive clauses*.

Clearly, Chronolog is a subset of Disjunctive Chronolog, obtained when all clauses are definite temporal clauses.

4 Declarative Semantics

4.1 Canonical Atom/Clause/Program

In order to define the model theoretic semantics of Disjunctive Chronolog programs, we will use the notion of *canonical temporal atoms/clauses/programs* [Org91]. A *canonical temporal atom* is a formula of the form[1] `first next`^{n} A or `first prev`^{n} A for some $n \geq 0$, where A is an atomic formula. A *canonical temporal disjunctive clause* is a temporal disjunctive clause whose temporal atoms are canonical temporal atoms. Finally, a *canonical temporal disjunctive program* is a set of canonical temporal disjunctive clauses.

[1] By `next`^{n} and `prev`^{n} we mean n applications of the operator `next` and `prev` respectively.

As in Chronolog [Org91, OWD93], every temporal disjunctive clause can be transformed into a (possibly infinite) set of canonical temporal disjunctive clauses. This can be done by applying `first next`^{n} where $n \geq 0$, as well as `first prev`^{n} where $n \geq 0$, to the clause, and then using the tautologies of TL to distribute the temporal reference so as to be applied to each individual temporal atom of the clause; finally any superfluous operator is eliminated by applying cancellation rules of TL.

Intuitively, a canonical temporal disjunctive clause is an instance in time of the corresponding temporal disjunctive clause.

The 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, taken from [OW93], shows:

Lemma 4.1. *Let C be a clause and I a temporal interpretation of TL. $\models_I C$ if and only if $\models_I C_t$ for all canonical instances C_t of C.*

Example 4.1. Consider the following (propositional) Disjunctive Chronolog program:

`first rains` $\vee$ `first snows.`
`next wet` $\leftarrow$ `rains.`
`next wet` $\leftarrow$ `snows.`

The set of canonical temporal disjunctive clauses corresponding to the program clauses is as follows:

The clause:

`first rains` $\vee$ `first snows.`

is the only canonical temporal clause corresponding to the first program clause (because of tautology 1a).

The sets of canonical clauses:

$\{$ `first next`$^{n+1}$ `wet` $\leftarrow$ `first next`^{n} `rains` $\mid n \geq 0$ $\}$

and

$\{$ `first prev`$^{n-1}$ `wet` $\leftarrow$ `first prev`^{n} `rains` $\mid n \geq 1$ $\}$

correspond to the second program clause. Finally the sets of canonical clauses:

$\{$ `first next`$^{n+1}$ `wet` $\leftarrow$ `first next`^{n} `snows` $\mid n \geq 0$ $\}$

and

$\{$ `first prev`$^{n-1}$ `wet` $\leftarrow$ `first prev`^{n} `snows` $\mid n \geq 1$ $\}$

correspond to the third program clause. □

Let P be a Disjunctive Chronolog program. The set of all canonical instances of the program clauses is itself a (possibly infinite) Disjunctive Chronolog program P_c which we call the *canonical instance of the program P*. In the following sections we show that the minimal model semantics, the model state semantics and the fixpoint semantics developed for disjunctive logic programs [LR91, LMR92], can be easily extended to apply to Disjunctive Chronolog programs. For the development of these semantics, we will use an interesting property of the canonical instance P_c of a Disjunctive Chronolog program P. That is, the clauses in P_c can be put into a one-to-one correspondence with the clauses of a classical disjunctive logic program. This idea was first used by M. Baudinet [Bau93]

in order to develop semantics for the temporal logic programming language TEMPLOG, by extending the semantics of Horn clause logic programs [Llo87].

The correspondence between P_c and the classical program P_c^* (which we will call *the classical counterpart of* P_c, following the terminology used by M. Baudinet) is established if we consider each `first next`^{n} `p` as well as each `first prev`^{n} `p`, i.e. each predicate symbol along with the temporal reference applied to it, as a single predicate symbol in a first-order language. More formally, the classical counterpart P_c^* of a canonical temporal program P_c is defined as follows: Let L^* be the language that contains the constant and function symbols of the language L of P_c and the predicates `first next`^{n} `p` and `first prev`^{n} `p`, for each predicate symbol p in L and each $n \geq 0$. Then P_c^* is the classical program obtained from P_c by replacing each predicate in P_c along with the temporal reference applied to it, by the classical predicate in L^* corresponding to it. It is easy to see that there is a one-to-one correspondence between the temporal interpretations for P_c (for L) and the classical interpretations of P_c^* (of L^*). Thus a temporal interpretation I satisfies P_c if and only if the corresponding classical interpretation I^* satisfies P_c^*. In this way, the results on the semantics of (possibly infinite) disjunctive logic programs apply to the classical counterpart of the canonical instance of the Disjunctive Chronolog program and then we can easily extend them to Disjunctive Chronolog programs.

4.2 Minimal Temporal Model Semantics

The minimal temporal model semantics of Disjunctive Chronolog are based on the notion of *Temporal Herbrand Models.*

The *Herbrand universe* U_P of a program P is the set of all ground terms that can be formed by the constant and function symbols that appear in P. The *temporal Herbrand base* THB_P is the set of all canonical ground temporal atoms whose predicate symbols appear in P and their arguments are in U_P. A *temporal Herbrand interpretation* I is a subset of THB_P. A temporal Herbrand interpretation which satisfies all clauses in P at all moments in time, is a *temporal Herbrand model* of P.

As in the case of the clausal form of first-order logic [CL73], in order to prove unsatisfiability of a set of TL clauses it suffices to consider only temporal Herbrand interpretations.

Example 4.2 (*Continued from example 4.1*). The temporal Herbrand base of the program P in example 4.1 is:

$B_P = \{$`first rains, first next rains, first next`^{2} `rains`$, \ldots,$
`first prev rains, first prev`^{2} `rains`$, \ldots,$
`first snows, first next snows, first next`^{2} `snows`$, \ldots,$
`first prev snows, first prev`^{2} `snows`$, \ldots,$
`first wet, first next wet, first next`^{2} `wet`$, \ldots,$
`first prev wet, first prev`^{2} `wet`$, \ldots \}$. □

Because of the correspondence between the canonical instance P_c of a program P and its classical counterpart P_c^*, the results concerning the minimal model semantics of disjunctive logic programs [LMR92], can be also applied to Disjunctive Chronolog

programs. Thus, a Disjunctive Chronolog program does not have in general a unique minimal temporal Herbrand model. Instead, its meaning can be captured by the set of its minimal temporal Herbrand models.

Theorem 4.1. *Let P be a Disjunctive Chronolog program. A canonical ground positive temporal clause C is a logical consequence of P if and only if C is true in all minimal temporal Herbrand models of P.*

The proof of this theorem as well as the proofs of the theorems and lemmas in the following sections, are trivial extensions of the proofs of the corresponding theorems and lemmas in the theory of disjunctive logic programs [LMR92] if we take into account the correspondence between the canonical instance of a Disjunctive Chronolog program and its classical counterpart.

Example 4.3 (*Continued from example 4.2*). It is easy to see that the program in example 4.1 has two minimal temporal Herbrand models:

$MM1(P) = \{$ `first rains, first next wet` $\}$

and

$MM2(P) = \{$ `first snows, first next wet`$\}$.

The positive ground clause `first next wet` is true in both minimal temporal Herbrand models and thus it is a logical consequence of the program. □

4.3 Temporal model state semantics

An alternative way which gives a least model characterization of the semantics of Disjunctive Chronolog programs. is obtained by extending the model state approach used in disjunctive logic programming [LMR92].

Definition 4.1 (*Temporal disjunctive Herbrand base*). Let P be a Disjunctive Chronolog program. Then, the *temporal disjunctive Herbrand base* ($TDHB_P$) of P is the set of all canonical ground positive temporal clauses formed using distinct elements from the temporal Herbrand Base of P.

Definition 4.2 (*Expansion*). Let P be a Disjunctive Chronolog program and S a set of canonical ground positive temporal clauses. The *expansion* $exp(S)$ of S is defined as follows:

$$exp(S) = \{C \in TDHB_P | C \in S \text{ or } \exists C' \in S \text{ such that } C' \text{ is a subclause of } C\}$$

Definition 4.3 (*Temporal disjunctive Herbrand state*). Let P be a Disjunctive Chronolog program. A *temporal disjunctive Herbrand state* of P is a subset of the temporal disjunctive Herbrand base $TDHB_P$ of P. An *expanded temporal disjunctive Herbrand state* TS of P is a temporal disjunctive Herbrand state of P such that $TS = exp(TS)$.

Definition 4.4 (*Temporal model state*). Let P be a Disjunctive Chronolog program. An expanded temporal disjunctive Herbrand state TS of P is said to be a *temporal model state* of P iff every minimal temporal Herbrand model of TS is a temporal Herbrand model of P. A temporal model state MS is *minimal* if no proper subset of MS is a temporal model state of P.

Lemma 4.2 (**Temporal model state intersection property**). *Let P be a Disjunctive Chronolog program and $\{M_i\}_{i\in\mathcal{N}}$ a non-empty set of temporal model states of P. Then, $\cap_{i\in\mathcal{N}} M_i$ is also a temporal model state of P.*

The intersection of all model states of a Disjunctive Chronolog program P, denoted by TMS_P, is called the *least model state of* P. The least model state of a Disjunctive Chronolog program characterizes the logical consequences of that program:

Theorem 4.2. *Let P be a Disjunctive Chronolog program. Then*

$$TMS_P = \{C \in TDHB_P | C \text{ is a logical consequence of } P\}.$$

The connection between minimal temporal model semantics and temporal model state semantics is shown in the theorem 4.4 in the next section.

4.4 Fixpoint semantics

The fixpoint semantics developed for disjunctive logic programs by J. Lobo, A. Rajasekar and J. Minker [LMR92, MR90, MRL91] can also be easily extended to Disjunctive Chronolog programs. The definition of the mapping T_P for Chronolog programs is given as follows.

Definition 4.5 (*Immediate consequence operator T_P*). Let P be a Disjunctive Chronolog program, and $TDHB_P$ be the temporal disjunctive Herbrand base of P. The *immediate consequence operator* $T_P : 2^{TDHB_P} \rightarrow 2^{TDHB_P}$ is defined as follows:

$T_P(I) = \{C \mid C' \leftarrow B_1, \ldots, B_n$ *is a canonical ground instance of a clause in* P,
and $\{B_1 \vee C_1, \ldots, B_n \vee C_n\} \subseteq I$ *where* $\forall i, 1 \leq i \leq n$ C_i *can be null*
and C *is* $C' \vee C_1 \vee \ldots \vee C_n$ *after eliminating the multiple occurrences*
of temporal atoms$\}$.

The power set of $TDHB_P$ of a program P is a complete lattice under the partial order of set inclusion ($\subseteq$). The bottom element of the lattice is the empty set ($\emptyset$), end the top element is the temporal disjunctive Herbrand base $TDHB_P$ of P.

of a disjunction of canonical temporal atoms. We call this form of goal clauses *simple goal clauses*. In general, a *goal clause* may be a conjunction of simple goal clauses (i.e. a conjunction of disjunction of canonical temporal atoms). Thus in general, Disjunctive Chronolog goal clauses are of the form:

$$\leftarrow G_1, \ldots, G_n$$

where each G_i is a canonical positive temporal clause. Comma stands for the conjunction operator '$\wedge$'.

Now, suppose that $X_1, \ldots, X_n$ are the free variables of the goal. Then, an answer to the goal might be a simple substitution or a set of substitutions. In the first case, we say that an answer substitution θ is a *correct answer* to the goal clause if

$$\forall(G_1 \wedge \ldots \wedge G_n)\theta$$

is a logical consequence of the program. Nevertheless, there are cases in which an answer is not a single substitution but a set of substitutions. For example, consider the program:

```
first course(data_structures) ∨ first course(algorithms).
```

Then, there is not a single substitution θ to answer the goal clause

$$\leftarrow \texttt{first course(X)}.$$

although $\exists$(`first course(X)`) is a logical consequence of the program. Thus, in general a *correct answer* is considered to be a set of substitutions $\{\theta_1, \theta_2, \ldots, \theta_k\}$ such that $\forall((G_1 \wedge \ldots \wedge G_n)\theta_1 \vee (G_1 \wedge \ldots \wedge G_n)\theta_2 \vee \ldots \vee (G_1 \wedge \ldots \wedge G_n)\theta_k)$ is a logical consequence of P.

5.2 Open-ended goal clauses

When not all temporal atoms included in a goal clause are canonical, we say that the goal clause is *open-ended*. An open-ended goal clause G represents the infinite set of all canonical goal clauses corresponding to G. An implementation strategy for executing open-ended goal clauses is by enumerating and evaluating (one by one) the set of all possible canonical instances of the goal clause. As in Chronolog [OW93] open-ended goal clauses are used to imitate non-terminating computations. In the following sections all goal clauses are considered to be canonical.

5.3 TSLO-resolution

SLO-resolution is a resolution-based proof procedure developed by J. Lobo, J. Minker and A. Rajasekar [LMR89, MRL91, LR91] so as to extract answers (logical consequences) from disjunctive logic programs. In this section we show how SLO-resolution can be directly extended to provide answers to Disjunctive Chronolog goal clauses. The new procedure is called *TSLO-resolution*. TSLO-resolution applies to canonical program and goal clauses.

Definition 5.1. Given a canonical positive temporal clause C, where $C = A_1 \vee \ldots \vee A_n$ and another canonical positive temporal clause D, we say that C *θ-subsumes* D iff $\theta = mgu((A_1, \ldots, A_n), (B_1, \ldots, B_n))$, where $B_1, \ldots, B_n$ are (not necessarily distinct) atoms in D.

Definition 5.2. Let P be a Disjunctive Chronolog program and G be a canonical temporal goal. A *TSLO-derivation* from P with top goal G consists of a (possibly infinite sequence) of canonical temporal goals $G_0 = G, G_1, \ldots, G_n, \ldots$ such that for all i the goal G_{i+1} is obtained from the goal:

$G_i = \leftarrow C_1, \ldots, C_{m-1}, C_m, C_{m+1}, \ldots, C_p$

as follows:

1. C_m is a canonical positive temporal clause in G_i (called the *selected* clause),
2. $C\,B \leftarrow B_1, \ldots, B_r$ is a canonical instance of a program clause,
3. $C\,B$ θ-subsumes C_m,
4. G_{i+1} is the goal:

   $G_{i+1} = \leftarrow (C_1, \ldots, C_{m-1}, (B_1 \vee C_m), \ldots, (B_r \vee C_m), C_{m+1}, \ldots, C_p)\theta$

   if $r > 0$, otherwise (if $r = 0$) the goal G_{i+1} is:

   $G_{i+1} = \leftarrow (C_1, \ldots, C_{m-1}, C_{m+1}, \ldots, C_p)\theta$.

Definition 5.3. Let P be a Disjunctive Chronolog program and G be a canonical temporal goal. A *TSLO-refutation* from P with top goal G is a finite TSLO-derivation of the null clause $\Box$ from P with top goal G.

Example 5.1. Let P the program:

```
(1)   first light(green) ∨ first light(red).
(2)   next light(amber) ← light(green).
(3)   next light(red) ← light(amber).
(4)   next light(green) ← light(red).
```

A TSLO-refutation of the canonical temporal goal:

```
← first next light(X) ∨ first next next light(X)
```

is given below (the underlined atoms of the goal clause are those which unify with the head of the canonical instance of the corresponding program clause i.e. those taking part

in the θ-subsumption):

$\leftarrow$ `first next light(X)` $\vee$ `first next next light(X)`
$\theta_1 = \{X/amber\}$ using clause (2)
$\leftarrow$ `first light(green)` $\vee$ `first next light(amber)`
$\vee$ `first next next light(amber)`
using clause (2)
$\leftarrow$ `first next light(green)` $\vee$ `first light(green)`
$\vee$ `first next light(amber)` $\vee$ `first next next light(amber)`
using clause (4)
$\leftarrow$ `first light(red)` $\vee$ `first next light(green)`
$\vee$ `first light(green)` $\vee$ `first next light(amber)`
$\vee$ `first next next light(amber)`
using clause (1)
$\square$

Thus by applying θ_1 to the initial goal we conclude that

`first next light(amber)` $\vee$ `first next next light(amber)`

is a logical consequence of the program. $\square$

By taking into account the one to one correspondence between the canonical instance P_c of a program P and its classical counterpart P_c^*, we can easily prove the following soundness and completeness theorems for TSLO-resolution which have been adapted from the corresponding theorems for the soundness and completeness of SLO-resolution for disjunctive logic programs [LR91].

Theorem 5.1. (Soundness) *Let P be a Disjunctive Chronolog program and $G = \leftarrow C_1 \wedge \ldots \wedge C_k$ be a goal. Suppose that there is a TSLO-refutation from P with top level goal G, and let $\theta_1, \ldots, \theta_n$ be the substitutions obtained from this refutation. Then, $\forall((C_1 \wedge \ldots \wedge C_k)\theta_1, \ldots, \theta_n)$ is a logical consequence of P.*

Theorem 5.2. (Completeness) *Let P be a Disjunctive Chronolog program and G a ground canonical positive temporal clause which is a logical consequence of P. Then, there is a TSLO-refutation from P with top level goal G.*

5.4 TSLI-Resolution

Another resolution-based proof procedure for disjunctive logic programs, developed in [MR90, LMR92], is the so-called *SLI-resolution*. In this section, we show how SLI-resolution can be directly extended to apply to Disjunctive Chronolog programs. We call the resulting procedure *TSLI-resolution*.

Following the formulation of SLI-resolution, TSLI-resolution uses trees to represent program clauses. Each node of the tree is a temporal literal either marked or unmarked. A nonterminal literal is always marked, while a terminal literal may be either marked or unmarked.

Definition 5.4 (*Temporal t-clause*). A *temporal t-clause* is an ordered pair $< T, m >$ where:

1. T is a labeled tree whose root is labeled with the symbol ϵ and whose nodes are labeled with temporal literals, and;

2. m is a marking (unary) relation on the nodes such that every non-terminal node in T is marked.

When all temporal literals of a temporal t-clause are canonical, the temporal t-clause is said to be a *canonical temporal t-clause.*

Program clauses as well as (simple[2]) goal clauses are represented as temporal t-clauses. A temporal t-clause can also be represented as a well parenthesized pre-order expression.

A TSLI-derivation starts with a canonical temporal t-clause (a canonical temporal goal t-clause) and successively derives further canonical temporal goal t-clauses by resolving with canonical instances of program t-clauses. During the derivation, an unmarked canonical temporal literal (either positive or negative) of the goal t-clause is selected and unified with a complementary canonical temporal literal of a canonical instance of a program t-clause (we say that this is a t-derivation step). The resolvent is attached as a subtree to the temporal literal in the goal clause. Besides t-derivation, also t-factoring, t-ancestry and t-truncation are used in TSLI-resolution. For the needs of the ancestry resolution and the factoring, we will use two sets γ_L and δ_L, defined as follows:

Definition 5.5. Let L be an unmarked temporal literal in a temporal t-clause.
$\delta_L = \{N : N$ is a marked temporal literal and an ancestor of $L\}$
$\gamma_L = \{M : M$ is an unmarked temporal literal and a sibling of an ancestor of $L\}$.

The set γ_L is used in order to perform factoring as well as to detect the derivation of a tautology. The set δ_L is used in ancestry resolution as well as to detect infinite derivations.

Definition 5.6 (*Admissibility condition*). A canonical temporal t-clause is said to satisfy the *admissibility condition* if for every occurrence of every unmarked canonical temporal literal L, the following conditions hold:

1. No two canonical temporal literals in γ_L and L have identical temporal atoms (modulo variable renaming).

2. No two canonical temporal literals in δ_L and L have identical temporal atoms (modulo variable renaming).

The satisfaction of the admissibility condition prevents tautologies and infinite loops from arising.

[2]If the goal clause is not simple, then it is transformed into a set of t-clauses.

Definition 5.7 (*Minimality condition*). A canonical temporal t-clause is said to satisfy the *minimality condition* if there is no marked temporal literal which is a terminal node.

The satisfaction of the minimality condition ensures that truncation is performed as soon as possible.

Definition 5.8. Let C_0 be a canonical temporal t-clause. The temporal t-clause C_n is a *trunfac-derivation* (truncation, ancestry and factoring) of C_0 when there is a sequence of canonical temporal t-clauses $C_0, C_1, \ldots, C_n$ and a sequence of substitutions $\theta_0, \theta_1, \ldots, \theta_{n-1}$ such that for all $i, 0 \leq i \leq n$, C_{i+1} is obtained from C_i by *t-factoring* iff:

1. L is an unmarked terminal literal in C_i;
2. M is an unmarked literal in γ_L and there is a substitution θ_i such that $M\theta_i = L\theta_i$;
3. C_{i+1} is $C'_i\theta_i$ where C'_i is the t-clause obtained by deleting the terminal node L from C_i.

C_{i+1} is obtained from C_i by *t-ancestry* iff

1. L is an unmarked terminal literal in C_i;
2. M^* is a marked literal in δ_L such that $L\theta_i = \neg M\theta_i$ where θ_i is a substitution (most general);
3. C_{i+1} is $C'_i\theta_i$ where C'_i is the tree obtained by deleting the terminal node L from C_i.

C_{i+1} is obtained from C_i by *t-truncation* with θ_i equal to the identity substitution iff C_{i+1} is a t-clause obtained from C_i by deleting a marked terminal node. (If C_i is (ϵ^*) then C_{i+1} is $\square$).

Definition 5.9. Let S be an input set of temporal t-clauses and C a canonical temporal t-clause in S. A *TSLI-derivation* of a t-clause E from S with top temporal t-clause C is a sequence of canonical temporal t-clauses $C_1, \ldots, C_n$ such that:

1. C_1 is either C or a tranfac-derivation of C, and C_n is E.
2. For all $i, 1 \leq i \leq n-1$, C_{i+1} is *t-derived* from C_i and a t-clause B_{i+1} in S. We say that C_{i+1} is t-derived from C_i and B_{i+1}, if there is an unmarked literal L in C_i and an unmarked literal M in B_{i+1} such that $L\theta'_i = \neg M\theta'_i$ where θ'_i is a most general unifier and if C'_{i+1} is the clause obtained by marking L in C_i and putting all siblings of M in B_{i+1} as child nodes of L^* and applying θ'_i, then C_{i+1} is either C'_{i+1} or a tranfac derivation of C'_{i+1}, such that C_{i+1} satisfy the admissibility and minimality conditions.

computed answer s.t. the correct answer is an instance of this computed answer) of TSLI-resolution are directly obtained from the soundness and completeness of SLI-resolution for disjunctive logic programs [LMR92]. The independence of the computation rule can also be proved.

6 Conclusions

Definition 5.10. Let S be an input set of temporal t-clauses and C a canonical temporal t-clause in S. A *TSLI-refutation* from S with top temporal t-clause C is a TSLI-derivation of the null clause $\Box$.

Example 5.2 (*Continued from example 4.1*). We will apply TSLI-resolution to answer the query:

? first next wet.

The proof is as follows:

(ϵ^*, ¬ first next wet)

t-derivation

(ϵ^*, (¬ first next wet*, ¬ first rains))

t-derivation

(ϵ^*, (¬ first next wet*, (¬ first rains*, first snows)))

t-derivation

(ϵ^*, (¬ first next wet*, (¬ first rains*, (first snows*, first next wet))))

t-ancestry

(ϵ^*, (¬ first next wet*, (¬ first rains*, (first snows*))))

t-truncation

(ϵ^*, (¬ first next wet*, (¬ first rains*)))

t-truncation

(ϵ^*,(¬ first next wet*))

t-truncation

ϵ^*

t-truncation

$\Box$.

$\Box$

Again by taking into account the one to one correspondence between the canonical instance P_c of a program P and its classical counterpart P_c^*, we can directly extend the soundness and completeness results of SLI-resolution [LR91] to TSLI-resolution.

Definition 5.11. Let P be a Disjunctive Chronolog program and G a temporal goal in temporal t-clause form. Suppose that G is the top level clause in a TSLI-refutation, in which G has been used n times with the corresponding renaming substitutions $\sigma_1, \ldots, \sigma_n$. Let θ be the composition of the substitutions computed for the variables in G during the refutation, and $\theta_1, \ldots, \theta_n$ be the substitutions such that each θ_i is obtained by restricting θ to the variables of σ_i. Then a *TSLI-computed answer* is given as $\{\theta_1\sigma_1, \ldots, \theta_n\sigma_n\}$.

As it is the case for TSLO-resolution, the soundness (i.e. that every computed answer is a correct answer) and completeness (i.e. that for every correct answer there is a