Temporal Disjunctive Logic Programming

Manolis GERGATSOULIS

Institute of Informatics & Telecommunications,
N.C.S.R. 'Demokritos', 153 10 A. Paraskevi Attikis, Greece
manolis@iit.demokritos.gr

Panos RONDOGIANNIS

Department of Computer Science, University of Ioannina,
P.O. Box 1186, 45110 Ioannina, Greece
prondo@cs.uoi.gr

Themis PANAYIOTOPOULOS

Department of Informatics, University of Piraeus
80 Karaoli & Dimitriou Str., 18534 Piraeus, Greece
themisp@unipi.gr

Received 10 March 2000

Abstract In this paper we introduce the logic programming language *Disjunctive Chronolog* which combines the programming paradigms of temporal and disjunctive logic programming. 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 and the fixpoint semantics for the new programming language and demonstrate their equivalence. We also show how proof procedures developed for disjunctive logic programs can be easily extended to apply to Disjunctive Chronolog programs.

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

§1 Introduction

Temporal logic programming [25] has been widely used as a means for describing systems that are inherently *dynamic*. For example, consider the following Chronolog [34] 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 [22, 17, 16] 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. Our starting point is the temporal language Chronolog proposed by W. W. Wadge [34] whose semantics have been systematically developed by M. Orgun [23, 26, 28]. The new language that we propose, called *Disjunctive Chronolog* [6], 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 register in the Algorithms or Data Structures course in the term after they have taken Discrete Mathematics. 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 enrol 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 [23] and Disjunctive Logic programming [16, 17]. More specifically, we define minimal model semantics and fixpoint semantics for Disjunctive Chronolog programs and show their equivalence. Moreover we show that proof procedures developed for disjunctive logic programs [15, 16, 17, 22], can be extended to apply to Disjunctive Chronolog programs.

This paper is organized as follows. Section 2, describes the temporal logic (TL) of Disjunctive Chronolog. Section 3 introduces the syntax and section 4 the declarative semantics of Disjunctive Chronolog programs. In section 5, we present a proof procedure for Disjunctive Chronolog programs. In section 6 we present related work. Finally, section 7 concludes the paper and suggests some topics for future work.

§2 The Temporal Logic of Disjunctive Chronolog

The temporal logic (TL) of Disjunctive Chronolog is the logic of the logic programming language Chronolog($\mathcal{Z}$) developed by W. W. Wadge and M. Orgun [26, 27, 28, 34]. In this section we give some useful background definitions regarding temporal logic which are adopted from [23, 28] and present some useful tautologies that are valid formulas of the logic TL.

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 $[D^n \rightarrow D]$; and for each n-ary predicate symbol, an element of $[\mathcal{Z} \rightarrow 2^{D^n}]$.

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* $\langle I(e_0), \ldots, I(e_{n-1})\rangle \in I(\mathtt{p})(t)$
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 $\models A$).

Now we present some useful tautologies that are valid formulas of the logic TL. The symbol ∇ stands for any of `first`, `next`, and `prev`. The proof of correctness of these tautologies is straightforward [23].

1. **Temporal operator cancellation rules:**

   a. $\nabla(\texttt{first}\ A) \leftrightarrow (\texttt{first}\ A)$
   b. $\texttt{next}\ \texttt{prev}\ A \leftrightarrow A$
   c. $\texttt{prev}\ \texttt{next}\ A \leftrightarrow A$

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

§3 Syntax of Disjunctive Chronolog Programs

The syntax of Disjunctive Chronolog extends the syntax of disjunctive logic programs [16] 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 disjunctive clause* is a clause of the form:

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

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

In this section we introduce the *minimal temporal model semantics* and the *fixpoint semantics* of Disjunctive Chronolog programs. For this we will use the notion of *canonical temporal atoms / clauses / programs* [23]. 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.

As in Chronolog [23, 28], 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 $n \geq 0$, 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. The same procedure is followed for `first prev`^{n}, where $n \geq 0$. For more details and examples on this transformation the interested reader may refer to [8].

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, originally stated in [27], 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.

4.1 Mapping Temporal Programs into Classical Programs

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 we show that the minimal model and the fixpoint semantics developed for disjunctive logic programs [16, 17], 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

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

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 [2] in order to develop semantics for the temporal logic programming language TEMPLOG, by extending the semantics of Horn clause logic programs [14].

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.

It is important to emphasize here that using the above mapping the proofs of all lemmas and theorems that follow are easy because they reduce to the proofs of the corresponding theorems in the theory of Disjunctive Logic Programming [16].

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, in order to prove unsatisfiability of a set of TL clauses it suffices to consider only temporal Herbrand interpretations.

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 [16], 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 iff 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 immediate extensions of the proofs of the corresponding theorems and lemmas in the theory of disjunctive logic programs [16] if we take into account the correspondence between the canonical instance of a Disjunctive Chronolog program and its classical counterpart.

4.3 Fixpoint Semantics

The fixpoint semantics developed for disjunctive logic programs by J. Lobo, A. Rajasekar and J. Minker [16, 21, 22] can also be easily extended to Disjunctive Chronolog programs. In order to define the fixpoint semantics we will introduce the following definitions:

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:

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

The definition of the mapping T_P for Chronolog programs is given as follows.

Definition 4.3 (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 $\forall i, with\ 1 \leq i \leq n,\ C_i$ can be null and C is the positive temporal clause obtained from $C' \vee C_1 \vee \ldots \vee C_n$ after eliminating the multiple occurrences of temporal atoms$\}$.

The power set of $TDHB_P$ for 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. In is known that monotonic mappings over complete lattices have fixpoints [14].

The following theorem shows the equivalence between minimal temporal model and the fixpoint semantics of Disjunctive Chronolog programs.

Theorem 4.2
Let P be a Disjunctive Chronolog program and $C \in TDHB_P$. Then the following are equivalent:

1. C is true in all minimal temporal Herbrand models of P.
2. C is in $exp(lfp(T_P))$.
3. C is a logical consequence of P.

§5 Procedural Semantics of Disjunctive Chronolog

A *Disjunctive Chronolog goal clause* is 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 '$\wedge$'. 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 the program.

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. In the following sections all goal clauses are considered to be canonical.

5.1 TSLO-resolution

By taking into account the correspondence between a canonical instance of a Disjunctive Chronolog program and its classical counterpart, one can easily adapt the proof procedures developed for disjunctive logic programs so as that they apply to Disjunctive Chronolog programs. In this section we illustrate how SLO-resolution, developed by J. Lobo, J. Minker and A. Rajasekar [15, 22, 17] for disjunctive logic programs, extends to apply to Disjunctive Chronolog programs. The new proof procedure, called *TSLO-resolution*, applies to canonical program and goal clauses.

Definition 5.1
Given two canonical positive temporal clauses C and D, where $C = A_1 \vee \ldots \vee A_n$, 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. $CB \leftarrow B_1, \ldots, B_r$ is a canonical instance of a program clause,
3. CB 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 (i.e. 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 from P with top goal G.

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^* as well as the one-to-one correspondence between a goal clause and its classical counterpart which is obtained in the same way, we can easily prove soundness and completeness of TSLO-resolution by adapting the corresponding theorems for the soundness and completeness of SLO-resolution for disjunctive logic programs [17).

§6 Related Work

Introducing uncertainty in programming languages appears to be useful in a number of interesting applications. Our first attempt in this direction was the design of the temporal logic programming language TRL [29, 30), which has been successfully used for a number of real world applications [18). However, although the proof system designed for TRL was sound, it was not complete mainly due to the model of uncertainty supported by the language. The language Disjunctive Chronolog proposed in this paper has been developed in order to provide a manageable form of time uncertainty as well as a sound and complete proof system.

In [24) Chronolog clauses without function symbols are considered as temporal deductive databases, specifying temporal relations among data and providing base relations to an algebraic front-end called TRA (Temporal Relational Algebra). Recently, considerable attention has been given to *Disjunctive Datalog* [4), as the basis for *disjunctive deductive databases.* We believe that the Disjunctive Chronolog language introduced in this paper, without function symbols, could be used as the basis for *temporal disjunctive deductive databases.*

In temporal reasoning and temporal databases, it is common to manipulate the time parameter (or more generally the *context*) as an extra argument in classical logic relations. This approach has been used in Datalog$_{1S}$ [3). It is easy to see that Datalog$_{1S}$ is closely related to Chronolog without function symbols. To our knowledge however, Datalog$_{1S}$ has not been extended to allow disjunctions in the heads of clauses. An amalgamation of Disjunctive Datalog [4) and Datalog$_{1S}$ would lead to a formalism which is similar in nature to Disjunctive Chronolog. Another formalism that resembles to both Chronolog and Datalog$_{1S}$ is (flat) Statelog [10], a language designed for applications on active deductive databases. The syntax of (flat) Statelog is similar to that of Chronolog, but only a special kind of rules (the *progressive* ones) are considered. To our knowledge, there does not exist an extension of Statelog for disjunctive programming.

Other formalisms that are related to the present work come from the artificial intelligence domain. Such an example is McCarthy's *situation calculus* [20], a formalism for describing change in first-order logic. The situation calculus views the world as consisting of a sequence of situations, each of which is a "snapshot" of the state of the world. Situations are generated from previous situations by actions. In the Chronolog case, situations are integer time-points and moving from one point to another is accomplished with the use of temporal operators. Research on the situation calculus has given numerous results, many of them concerning extensions related to temporal reasoning [31, 32]. The situation calculus is not a logic programming language (in the usual sense) but a formal system for representing change in the world. In this respect, the two formalisms have different goals and different underlying theory. For example, the minimal model and the fixpoint semantics that characterize elegantly the meaning of Disjunctive Chronolog programs, do not apply in general to a Situation Calculus specification. It should be mentioned however that there has been research on designing logic programming languages based on the situation calculus. One such example is the GOLOG language [11, 12] which borrows features from both the situation calculus and classical logic programming.

Other time-related formalisms that come from the artificial intelligence domain, are action languages [1, 5, 13], causal logic [19] and the event calculus [9] (see also the special issue of the Journal of Logic Programming, Vol 31, No 1-3, on "Reasoning about action and Change"). Again, these are mainly declarative specification formalisms rather than logic programming languages. It would be interesting though to refer to [5], in which the action language $\mathcal{A}$ is translated into "extended logic programs"; the conclusions drawn by the authors of [5] is that "to make the translation complete ... can be achieved by using the more expressible language of disjunctive programs". This is also noticed in [1] where the use of disjunction is also suggested in order to achieve completeness of translation. In this reference however it is noticed that the usefulness of their approach "...depends on the development and availability of query answering systems for disjunctive and abductive programs".

A conclusion that can be drawn from the above discussion is that there exist common aspects between temporal logic programming and the formalisms proposed for representing action and change in Artificial Intelligence. The two above areas can certainly influence each other, but a further examination of such connections is outside the scope of this paper. The work proposed in this paper is the enhancement of temporal logic programming with disjunctions, an extension that leads to better capabilities for expressing uncertainty in a temporal framework.

§7 Conclusions

Temporal programming, either functional or logic, has been widely used as a means for describing systems that are inherently *dynamic*. On the other hand the need to express uncertainty has led researchers to introduce disjunctive logic programming [16)]. In this paper, we have introduced the paradigm of temporal disjunctive logic programming and proposed a new programming language, called Disjunctive Chronolog, which combines the virtues of Disjunctive and Temporal logic programming in a unified framework. We have presented the syntax and the declarative semantics of the language. In particular, we have developed the minimal model and the fixpoint semantics and have demonstrated their equivalence. It is worth noting here that the model state semantics of disjunctive logic programs [16)], can be easily extended to apply to Disjunctive Chronolog programs.

We have also shown that SLO-resolution, a proof procedures developed for disjunctive logic programs, can be easily extended to apply to Disjunctive Chronolog programs. Using the idea of mapping a canonical instance of a Disjunctive Chronolog program into a classical program, we can also adapt easily other resolution based proof procedures developed for disjunctive logic programs, such as *SLI-resolution* [16, 21)] and SLOP-resolution [15)] (SLO-resolution with program partitions) to apply to Disjunctive Chronolog programs. More details about this adaptation can be found in [8)].

The main tool we have used in our investigation is the notion of canonical temporal clauses. This concept was introduced by W. W. Wadge and M. Orgun [23, 28)], and is a common practice in dealing with temporal [23, 27, 28)] logic programming languages. However, we have recently developed a proof procedure [7)] for branching-time logic programs [33)], which does not require the restriction to canonical program and goal clauses. Since such proof procedures can not be obtained by directly extending proof procedures for disjunctive logic programs, it would be an interesting topic for further research to investigate whether this generalized procedure can be adapted to apply to Disjunctive Chronolog programs. It might also be interesting to consider a disjunctive extension of the temporal logic programming language TEMPLOG [2] which offers a more expessive set of temporal operators than Chronolog.

Acknowledgment This work has been partially supported by the Greek General Secretariat of Research and Technology under the project "Time-Logic" of ΠΕΝΕΔ′95, contract no 1134.

The authors would like to thank C. Nomikos for valuable comments and the anonymous reviewers for their insightful suggestions. In particular, one of the reviewers provided many useful pointers indicating the connections between temporal logic programming and the formalisms used in Artificial Intelligence for representing action and change.

References

1) Baral, C., Gelfond, M. and Provetti, A., "Representing actions: Laws, observations and hypotheses," *The Journal of Logic Programming*, 31(1–3):201–244, 1997.

2) Baudinet, M., "A simple proof of the completeness of temporal logic programming," in *Intensional Logics for Programming* (L. Farinas del Cerro and M. Penttonen, eds.), pages 51–83. Oxford University Press, 1993.

3) Chomicki, J., "Depth-bounded bottom-up evaluation of logic programs," *The Journal of Logic Programming*, 25(1):1–31, 1995.

4) Eiter, T., Gottlob, G. and Mannila, H., "Disjunctive datalog," *ACM Transactions on Database Systems*, 22(3):364–418, September 1997.

5) Gelfond, M. and Lifschitz, V., "Representing action and change by logic programs," *The Journal of Logic Programming*, 17(2,3 & 4):301–323, 1993.

6) Gergatsoulis, M., Rondogiannis, P. and Panayiotopoulos, T., "Disjunctive Chronolog," in *Proceedings of the JICSLP'96 Post-Conference Workshop "Multi-Paradigm Logic Programming"* (M. Chacravarty, Y. Guo, and T. Ida, eds.), pages 129–136, Bonn, 5-6 Sept. 1996.

7) Gergatsoulis, M., Rondogiannis, P. and Panayiotopoulos, T., "Proof procedures for branching-time logic programs," in *Proc. of the Tenth International Symposium on Languages for Intensional Programming (ISLIP'97)* (W. W. Wadge, ed.), May 15-17, Victoria BC, Canada, pages 12–26, 1997.

8) Gergatsoulis, M., Rondogiannis, P. and Panayiotopoulos, T., "Temporal disjunctive logic programming," Technical Report 15-97, Dept. of Computer Science, University of Ioannina, Greece, 1997.

9) Kowalski, R. and Sergot, M., "A logic-based calculus of events," *New Generation Computing*, 4:67–95, 1986.

10) Lausen, G., Ludäscher, B. and May, W., "On Active Deductive Databases: The Statelog Approach," in *Tansactions and Change in Logic Databases* (B. Freitag, H. Decker, M. Kifer, and A. Voronkov, eds.), Lecture Notes in Computer Science (LNCS) 1472, pages 69–106. Springer-Verlag, 1998.

11) Levesque, H., Pirri, F. and Reiter, R., "Foundations for the situation calculus," *Linköping Electronic Articles in Computer and Information Science*, 3(18), 1998.

12) Levesque, H., Reiter, R., Lespérance, Y., Lin, F. and Scherl, R., "GOLOG: A logic programming language for dymanic domains," *The Journal of Logic Programming*, 31(1–3):59–83, 1997.

13) Lifschitz, V., "Action languages, Answer sets and planning," in *The Logic Programming Paradigm: A 25-Year Perspective*, (K. R. Apt, V. W. Marek, M. Truszczynski, and D. S. Warren, eds.), pages 357–373, Springer-Verlag, 1999.

14) Lloyd, J. W., *Foundations of Logic Programming.* Springer-Verlag, 1987.

15) Lobo, J., Minker, J. and Rajasekar, A., "Extending the semantics of logic programs to disjunctive logic programs," in *Proc. of the Sixth International Conference on Logic Programming* (G. Levi and M. Martelli, eds.), pages 255–267. MIT Press, 1989.

16) Lobo, J., Minker, J. and Rajasekar, A., *Foundations of Disjunctive Logic Programming.* MIT Press, 1992.

17) Lobo, J. and Rajasekar, A., "Semantics of Horn and disjunctive logic programs," *Theoretical Computer Science*, 86(1):93–106, 1991.

18) Marinagi, C., Panayiotopoulos, T., Vouros, G. and Spyropoulos, C., "Advisor: A knowledge-based planning system," *International Journal of Expert Systems*, 9(3):319–355, 1997.

19) McCain N. and Turner, H., "A causal theory of ramifications and qualifications," in *Proc. of the Fourteenth International Conference on Artificial Intelligence (IJCAI' 95)*, pages 1978–1984, 1995.

20) McCarthy J. and Hayes, P., "Some philosophical problems from the standpoint of artificial intelligence," *Machine Intelligence*, 4:463–502, 1969.

21) Minker, J. and Rajasekar, A., "A fixpoint semantics for disjunctive logic programs," *The Journal of Logic Programming*, 9(1):45–74, July 1990.

22) Minker, J. and Rajasekar, A. and Lobo, J., "Theory of disjunctive logic programs," in *Computational Logic. Essays in the Honor of Alan Robinson*, (J. L. Lasser and G. Plotkin, eds.), pages 613–639. MIT Press, 1991.

23) Orgun, M. A., *Intensional logic programming.* PhD thesis, Dept. of Computer Science, University of Victoria, Canada, December 1991.

24) Orgun, M. A., "On temporal deductive databases," *Computational Intelligence*, 12(2):235–259, 1996.

25) Orgun, M. A. and Ma, W., "An overview of temporal and modal logic programming," in *Proc. of the First International Conference on Temporal Logics (ICTL'94)* (D. M. Gabbay and H. J. Ohlbach, eds.), Lecture Notes in Computer Science (LNCS) 827, pages 445–479. Springer-Verlag, 1994.

26) Orgun, M. A. and Wadge, W. W., "Towards a unified theory of intensional logic programming," *The Journal of Logic Programming*, 13(4):413–440, August 1992.

27) Orgun, M. A. and Wadge, W. W., "Chronolog admits a complete proof procedure," in *Proc. of the Sixth International Symposium on Lucid and Intensional Programming (ISLIP'93)*, pages 120–135, 1993.

28) Orgun, M. A., Wadge, W. W. and Du, W., "Chronolog($\mathcal{Z}$): Linear-time logic programming," in *Proc. of the Fifth International Conference on Computing and Information* (O. Abou-Rabia, C. K. Chang, and W. W. Koczkodaj, eds.), pages 545–549. IEEE Computer Society Press, 1993.

29) Panayiotopoulos, T. and Gergatsoulis, M., "Intelligent information processing using TRLi," in *6th International Conference and Workshop on Data Base and Expert Systems Applications (DEXA' 95), (Workshop Proceedings)* (A. Min Tjoa, and N. Revell, ed.), London, UK, 4-8 September, pages 494–501, 1995.

30) Panayiotopoulos, T. and Gergatsoulis, M., "A Prolog like temporal reasoning system," in *Proc. of 13th IASTED International Conference on APPLIED INFORMATICS*, (M. H. Hamza, ed.), ICLS(Innsbruck), Austria 21-23 Ferbuary, pages 123–126, 1995.

31) Pinto, J. and Reiter, R., "Temporal reasoning in logic programming: A case for the situation calculus," in *Proc. of the Tenth International Conference on Logic Programming*, (D. S. Warren, ed.), pages 203–221. MIT Press, 1993.

32) Pinto, J. and Reiter, R., "Reasoning about time in the situation calculus," *Annals of Mathematics and Artificial Intelligence*, 14(2–5):251–268, 1995.

33) Rondogiannis, P., Gergatsoulis, M. and Panayiotopoulos, T., "Branching-time logic programming: The language Cactus and its applications," *Computer Languages*, 24(3):155–178, October 1998.

34) Wadge, W. W., "Tense logic programming: A respectable alternative," in *Proc. of the 1988 International Symposium on Lucid and Intensional Programming*, pages 26–32, 1988.