

Disjunctive Chronolog^{*†}M. Gergatsoulis, P. Rondogiannis, T. Panayiotopoulos¹

Inst. of Informatics and Telecommunications, N.C.S.R. “Demokritos”
 15310 Aghia Paraskevi, Athens, Greece
 e-mail : {prondo,manolis}@iit.nrcps.ariadne-t.gr

¹Dept. of Informatics, University of Piraeus
 80 Karaoli & Dimitriou Str., 18534 Piraeus, Greece
 e-mail : themisp@unipi.gr

Abstract

A disjunctive temporal logic programming language, called *Disjunctive Chronolog* is presented in this paper. Disjunctive Chronolog combines the ideas of both temporal logic programming and disjunctive logic programming. The new language is capable of expressing dynamic behaviour as well as uncertainty, two notions that are very common in a variety of real systems. Minimal model semantics, model state semantics and fixpoint semantics are developed for the proposed language and their equivalence is shown.

Keywords: Temporal logic programming, disjunctive logic programming.

1 Introduction

Temporal logic programming[OM94, Org91] 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, 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.

*This work has been funded by the Greek General Secretariat of Research and Technology under the project “TimeLogic” of ΠΕΝΕΔ’95, contract no 1134.

†This paper appears in the Proceedings of the JICSLP’96 Post-Conference Workshop “MULTI-PARADIGM LOGIC PROGRAMMING” M. Chacravarty, Y. Guo, T. Ida (eds), Bonn 5-6 Sept, 1996, p.p. 129-136.

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 and present the semantics of Disjunctive Chronolog, a temporal disjunctive logic programming language. Our starting point is the temporal language Chronolog, introduced by W. W. Wadge[Wad88] whose semantics have been systematically developed by M. Orgun[Org91, OW92, OWD93]. The proposed formalism, Disjunctive Chronolog, 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 enrol in the Algorithms course, in the Data Structures course, or in both. The particular event that will take place is not known.

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 rest of this paper is organized as follows. Section 2 introduces the syntax of Disjunctive Chronolog. In section 3, after giving some background definitions, we present the minimal Herbrand model semantics, the model state semantics and the fixpoint semantics of the language. Finally, in section 4 a conclusion is given.

2 Syntax of Disjunctive Chronolog programs

Disjunctive Chronolog is a logic programming language based on linear time with unbounded past and future. The set of moments in time is represented by the set \mathcal{Z} of integers. The temporal logic (TL) of Disjunctive Chronolog 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. Some of the axioms of Disjunctive Chronolog’s underlying temporal logic are given below. The symbol ∇ stands for any of **first**, **next**, and **prev**.

Temporal operator cancellation rules:	Temporal operator distribution rules:
1) $\nabla(\mathbf{first} A) \leftrightarrow (\mathbf{first} A)$	4) $\nabla(\neg A) \leftrightarrow \neg(\nabla A)$
2) $\mathbf{next} \mathbf{prev} A \leftrightarrow A$	5) $\nabla(A \wedge B) \leftrightarrow (\nabla A) \wedge (\nabla B)$
3) $\mathbf{prev} \mathbf{next} A \leftrightarrow A$	6) $\nabla(A \vee B) \leftrightarrow (\nabla A) \vee (\nabla B)$
Rigidness of variables:	Temporal operator introduction rules:
7) $\nabla(\forall X)(A) \leftrightarrow (\forall X)(\nabla A)$	8) <i>if</i> $\vdash A$ <i>then</i> $\vdash \nabla A$

The proof of correctness of these axioms and inference rules is straightforward[Org91].

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 *disjunctive temporal clause* is a clause of the form:

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

where $H_1, \dots, H_n, B_1, \dots, B_m$ are temporal atoms, $n \geq 1$ and $m \geq 0$. 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 disjunctive temporal clause*. A *Disjunctive Chronolog program* is a finite set of *disjunctive temporal clauses*. Clearly, Chronolog is a subset of Disjunctive Chronolog, obtained when all clauses are definite temporal clauses.

3 Declarative Semantics

3.1 Background

In this subsection we give some useful background definitions regarding temporal logic which are adopted from[Org91].

Definition 3.1 (*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 $[Z \rightarrow 2^{D^n}]$. •

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

Definition 3.2 (*Semantics of TL*). The semantics of the elements of the temporal logic TL are given inductively as follows:

1. If $\mathbf{f}(e_0, \dots, e_{n-1})$ is a term, then $I(\mathbf{f}(e_0, \dots, e_{n-1})) = I(\mathbf{f})(I(e_0), \dots, I(e_{n-1}))$.
2. For any n -ary predicate symbol \mathbf{p} and terms e_0, \dots, e_{n-1} ,
 $\models_{I,t} \mathbf{p}(e_0, \dots, e_{n-1})$ iff $\langle I(e_0), \dots, I(e_{n-1}) \rangle \in I(\mathbf{p})(t)$
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. $\models_{I,t} \mathbf{first} A$ iff $\models_{I,0} A$
8. $\models_{I,t} \mathbf{prev} A$ iff $\models_{I,t-1} A$
9. $\models_{I,t} \mathbf{next} A$ 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 .

In order to simplify the examples, we consider in the following the subset of Disjunctive Chronolog which does not contain the temporal operator **prev**. However, from a theoretical point of view, the inclusion of **prev** is straightforward.

3.2 Minimal model semantics

The semantics of Disjunctive Chronolog are based on the notion of *Temporal Herbrand Models*. In order to define temporal Herbrand models, we need the notion of a *canonical temporal atom*[Org91]. A canonical temporal atom is a formula of the form¹ **first next** ^{n} A for some $n \geq 0$, where A is an atomic formula. A *canonical disjunctive temporal clause* is a disjunctive temporal clause whose temporal atoms are canonical temporal atoms.

As in Chronolog [Org91, OWD93], every disjunctive temporal clause can be transformed into a (possibly infinite) set of canonical disjunctive temporal clauses. This can be done by applying **first next** ^{n} where $n \geq 0$, to the clause, and then using the axioms 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 disjunctive temporal clause is an instance in time of the corresponding disjunctive temporal clause.

The value of a given formula 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 3.1. *Let A be a formula and I a temporal interpretation of *TL*. $\models_I A$ if and only if $\models_I A_t$ for all canonical instances A_t of A .*

Example 3.1 . Consider the following disjunctive Chronolog program:

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

The set of canonical disjunctive temporal 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 axiom 1).

The set of clauses:

```
{ first next $t+1$  wet  $\leftarrow$  first next $t$  rains |  $t \geq 0$ }
```

corresponds to the second program clause. Finally the set of clauses:

```
{ first next $t+1$  wet  $\leftarrow$  first next $t$  snows |  $t \geq 0$ }
```

corresponds to the third program clause. •

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* TB_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 TB_P . A temporal

¹By **next** ^{n} we mean n applications of the operator **next**.

Herbrand interpretation which satisfies all clauses in P at all moments in time, is a *temporal Herbrand model* of P .

In order to prove unsatisfiability of a set of TL clauses it suffices to look for temporal Herbrand models (as in the case of the clausal form of first order logic[CL73]).

Example 3.2 (*Continued from example 3.1*). The temporal Herbrand base of the program P in example 3.1 is:

$$B_P = \{ \text{first rains, first next rains, first next}^2 \text{ rains,.....,} \\ \text{first snows, first next snows, first next}^2 \text{ snows,.....,} \\ \text{first wet, first next wet, first next}^2 \text{ wet,.....} \}. \bullet$$

As in disjunctive logic programming[LMR92], a Disjunctive Chronolog program does not have in general a unique minimal temporal Herbrand model. Instead, the meaning of a Disjunctive Chronolog program is captured by the set of minimal temporal Herbrand models of the program.

Theorem 3.1. *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 .*

Example 3.3 (*Continued from example 3.2*). It is easy to see that the program in example 3.1 has two minimal temporal Herbrand models:

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

and

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

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

3.3 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 3.3 (*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 (TB_P) of P such that no two clauses contain exactly the same temporal atoms. \bullet

Definition 3.4 (*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:

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

\bullet

Definition 3.5 (*Temporal Herbrand state*). Let P be a Disjunctive Chronolog program. A *temporal Herbrand state* of P is a subset of the temporal disjunctive Herbrand base $TDHB_P$ of P . An *expanded temporal Herbrand state* TS of P is a temporal Herbrand state of P such that $TS = exp(TS)$. \bullet

Definition 3.6 (*Temporal Model state*). Let P be a Disjunctive Chronolog program. Then, a *temporal model state* TS of P is an expanded temporal Herbrand state of P such that:

1. Every minimal temporal Herbrand model of TS is a temporal Herbrand model of P .
2. Every minimal temporal Herbrand model of P is contained in a minimal temporal Herbrand model of TS .

•

A temporal model state MS of a program P is *minimal* if no temporal Herbrand state MS' of P which is a proper subset of MS is a temporal model state of P .

Lemma 3.2 (**Temporal Model state intersection property**). *Let P be a Disjunctive Chronolog program and $\{M_i\}_{i \in I}$ a non-empty set of temporal model states of P . Then, $\bigcap_{i \in I} M_i$ is also a temporal model state of P .*

Since for every disjunctive Chronolog program P , the temporal disjunctive Herbrand base $TDHBP$ is a temporal model state of P , the set of model states is non-empty. Therefore, the intersection of all model states of P , is also a model state of P , which is the *least model state* of P , denoted by $MTMS_P$.

Theorem 3.2. *Let P be a Disjunctive Chronolog program. Then*

$$MTMS_P = \{C \in TDHBP \mid C \text{ is a logical consequence of } P\}.$$

The connection between minimal model semantics and model state semantics is shown in the theorem 3.3 in the next section.

3.4 Fixpoint semantics

In this section, we provide a fixpoint characterization of the semantics of Disjunctive Chronolog programs using a closure operator that maps temporal Herbrand states to temporal Herbrand states of a program.

Definition 3.7 (*Immediate consequence operator*). Let P be a Disjunctive Chronolog program, and $TDHBP$ be the temporal disjunctive Herbrand base of P . The *closure operator* $T_P : 2^{TDHBP} \rightarrow 2^{TDHBP}$ is defined as follows :

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

The power set of $TDHBP$ of a program P is a complete lattice under the partial order of set inclusion. The bottom element of the lattice is the empty set, and the top element is $TDHBP$. The operator T_P is continuous and hence reaches a least fixpoint. The least fixpoint $lfp(T_P)$ of T_P characterizes the derivability from a Disjunctive Chronolog program. This is shown, in the following lemma.

Lemma 3.3. *Let P be a Disjunctive Chronolog program. Then $MTMS_P = exp(lfp(T_P))$.*

Example 3.4 (*Continued from example 3.2*). Applying the fixpoint operator to the program in example 3.1 we obtain:

$$S_1 = T_P(\emptyset) = \{\text{first rains} \vee \text{first snows}\}$$

$$S_2 = T_P(S_1) = \{\text{first rains} \vee \text{first snows}, \text{first rains} \vee \text{first next wet}, \text{first snows} \vee \text{first next wet}\}.$$

$$S_3 = T_P(S_2) = \{\text{first rains} \vee \text{first snows}, \text{first rains} \vee \text{first next wet}, \text{first snows} \vee \text{first next wet}, \text{first next wet}\}.$$

It is easy to see that S_3 is a fixpoint of T_P . •

The following theorem shows the equivalence between minimal model, temporal model state and fixpoint semantics.

Theorem 3.3. *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 $MTMS_P$.
3. C is in $exp(lfp(T_P))$.
4. C is a logical consequence of P .

4 Conclusions

Temporal programming, either functional [WA85, RW96] or logic [OM94, Org91, PG95], have 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 in introducing disjunctive logic programming [LMR92]. We have developed the syntax and declarative semantics of a new logic programming language, called Disjunctive Chronolog. The new formalism combines the virtues of Disjunctive and Temporal logic programming in a unified framework.

More specifically, we have developed the minimal model, temporal model state, and fixpoint semantics and have demonstrated their equivalence.

In this paper however, we have not considered the proof system of Disjunctive Chronolog which will be reported in a forthcoming paper.

References

- [CL73] C. L. Chang and R. C. T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Inc, 1973.
- [LMR92] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming*. MIT Press, 1992.
- [MRL91] J. Minker, A. Rajasekar, and J. Lobo. Theory of disjunctive logic programs. In J. L. Lasser and G. Plotkin, editors, *Computational Logic. Essays in the Honor of Alan Robinson*, pages 613–639. MIT Press, 1991.
- [OM94] M. A. Orgun and W. Ma. An overview of temporal and modal logic programming. In *Proc. of the First International Conference (ICTL'94)*, pages 445–479. Springer Verlag, 1994. LNCS No 827.

- [Org91] M. A. Orgun. *Intensional Logic Programming*. PhD thesis, Dept. of Compute Science, University of Victoria, Canada, December 1991.
- [OW92] M. A. Orgun and W. W. Wadge. Towards a unified theory of intensional logic programming. *The Journal of Logic Programming*, 13(4):113–145, August 1992.
- [OW93] M. A. Orgun and W. W. Wadge. Chronolog admits a complete proof procedure. In *Proc. of the Sixth International Symposium on Lucid and Intensional Programming (ISLIP'93)*, pages 120–135, 1993.
- [OWD93] M. A. Orgun, W. W. Wadge, and W. Du. Chronolog(\mathcal{Z}): Linear-time logic programming. In O. Abou-Rabia, C. K. Chang, and W. W. Koczkodaj, editors, *Proc. of the fifth International Conference on Computing and Information*, pages 545–549. IEEE Computer Society Press, 1993.
- [PG95] T. Panayiotopoulos and M. Gergatsoulis. Intelligent information processing using TRLi. In *6th International Conference and Workshop on Data Base and Expert Systems Applications (DEXA '95), (Workshop Proceedings) London, UK, 4th-8th September*, pages 494–501, 1995.
- [RW96] P. Rondogiannis and W. W. Wadge. First-order functional languages and intensional logic. *Journal of Functional Programming*, 1996. (to appear).
- [WA85] W. W. Wadge and E. A. Ashcroft. *Lucid, the dataflow Programming Language*. Academic Press, 1985.
- [Wad88] W. W. Wadge. Tense logic programming: A respectable alternative. In *Proc. of the 1988 International Symposium on Lucid and Intensional Programming*, pages 26–32, 1988.