A Prolog like temporal reasoning system

T. Panayiotopoulos and M. Gergatsoulis
Institute of Informatics and Telecommunications
N.C.S.R Demokritos
15310 Aghia Paraskevi, Athens, Greece
e-mail : {themisp,manolis}@iit.nrcps.ariadne-t.gr

Abstract
An extension of the Horn clause logic programming language (PROLOG), called Horn Temporal Reference Language (HTRL), suitable for temporal reasoning is presented in this paper. The syntax of the HTRL language is given and its informal semantics is briefly presented. An implementation through the transformation of an HTRL program to an equivalent Constraint Logic Program is also briefly presented.

Keywords
Temporal reasoning, logic programming, constraint logic programming, uncertainty.

1 Introduction

The problem of representing time depended information and reasoning about time has become an issue of great concern for many researchers in Artificial Intelligence during the last few years [8, 10]. Temporal Logics[6] have found application in many domains, such as, planning, temporal deductive data bases, verification of concurrent systems, VLSI design, etc. Some work has also been reported in Constraint Logic Programming (CLP) and its application to Temporal Reasoning [2, 3, 5].

Many practical systems which implement Temporal Logics have also been reported[8]. However, most of them are implementations of Modal temporal logics [6, 8].

In this paper we propose an extension of the Horn clause logic programming language (PROLOG), called Horn Temporal Reference Language (HTRL), suitable for Temporal Reasoning. The HTRL system is based on the Temporal Reference Language (TRL), and handles temporal references [9, 4]. Temporal references are labels assigned to atoms, to express the time during which an atom is true. Certain or uncertain time information may be expressed through temporal references.

The rest of this paper is organized as follows. In section 2, the syntax, the informal semantics and the inference rules of the HTRL language are presented. In section 3, the implementation of the HTRL system is presented. Finally, in section 4, a conclusion is given and some thoughts for future work are discussed.

2 Syntax, Semantics and deduction in HTRL

TRL [9] is a temporal logic which expresses temporal information in the form of *temporal references*. Temporal references are labels of atoms and formulae. In HTRL we assume that temporal references are not used as labels of formulae, but only as labels of atoms. In the following, familiarity is assumed with the basic notions of logic programming[7] such as term, atom, clause, resolution etc.

2.1 Definitions

An *HTRL atom* is either a *classical atom* or an *extended atom*. An extended atom is of the form $T_{ref} : A$, where T_{ref} is a temporal reference and A is a classical atom. Temporal references [4], are constructed by using *temporal constants*, *temporal variables* and the *temporal constructors* '<','>','[',']'.

The most general temporal reference is the *uncertain temporal interval*. Uncertain temporal interval is an expression of the form $< [T_1, T_2], [T_3, T_4] >$, where each of T_i is either a temporal constant or a temporal variable. In other words, an uncertain temporal interval represents a temporal interval with uncertain start and end points. A temporal reference $< [T_1, T_2], [T_3, T_4] >$ is said to be *consistent* if $T_1 \leq T_2, T_3 \leq T_4, T_1 \leq T_3, T_2 \leq T_4$.

All other forms of temporal references are special cases of the uncertain temporal reference [9]:

- a *temporal point* t, is an abbreviation of $< [t,t], [t,t] >$,
- a (*certain*) *temporal interval*, $< t_1, t_2 >$, is an abbreviation of $< [t_1, t_1], [t_2, t_2] >$,
- a *temporal instance*, $[t_1, t_2]$, is an abbreviation of $< [t_1, t_2], [t_1, t_2] >$.

Therefore, an uncertain temporal interval is a temporal reference in *temporal reference canonical form*.

In HTRL, there are two different types of extended atoms: *events* and *properties*. The distinction and the corresponding semantics follow Allen's approach[1]. The difference between events and properties concerns their behaviour over temporal intervals and not their syntax.

When a property is true over an interval it is necessarily true over every subinterval of this interval, i.e. When $T_2 \leq T_3$ then $< [T_1, T_2], [T_3, T_4] >: A$ is true iff there is at least an interval, $< S_1, S_2 >$, such that $T_1 \leq S_1 \leq T_2$, and $T_3 \leq S_2 \leq T_4$, during which A is true.

When an event is true over an interval it is not necessarily true over any subinterval of this interval, i.e. When $T_2 \leq T_3$ then $< [T_1, T_2], [T_3, T_4] >: A$ is true iff A is true over the temporal interval $< T_2, T_3 >$.

An *HTRL clause* has the form:
$A_0 \leftarrow A_1, ..., A_n$
where A_i $(i = 1, ..., n)$ are HTRL atoms. An *HTRL canonical clause* is an HTRL clause in which all temporal references are in their canonical form. An *HTRL program* is a set of HTRL clauses.

Example 1. In this example we have a promotion problem, represented as an HTRL program. Notice that there are certain temporal references but also uncertain ones. Uncertain temporal references represent cases in which the knowledge about the occurrence of the corresponding atom is only known within some temporal bounds.

$\leftarrow declaration([hire/2, promote/3, leave/2, rank/2], property).$
$1980 : hire(mary, lecturer).$
$1985 : promote(mike, lecturer, professor).$
$[1983, 1984] : promote(mary, lecturer, professor).$
$1989 : leave(mary, professor).$
$[1988, 1989] : leave(mike, professor).$
$< [T1, T2], [T3, T4] >: rank(Name, Rank) \leftarrow$
$[T1, T2] : hire(Name, Rank),$
$[T3, T4] : leave(Name, Rank).$
$< [T1, T2], [T3, T4] >: rank(Name, Rank1) \leftarrow$
$[T1, T2] : hire(Name, Rank1),$
$[T3, T4] : promote(Name, Rank1, Rank2).$
$< [T1, T2], [T3, T4] >: rank(Name, Rank2) \leftarrow$
$[T1, T2] : promote(Name, Rank1, Rank2),$
$[T3, T4] : leave(Name, Rank2).$

2.2 Inference rules

The inference system of HTRL language is a resolution based inference system. For classical atoms, we retain SLD-resolution. For extended atoms we have defined additional inference rules which impose constraints concerning the temporal references. These rules are briefly described in the following lines:

2.2.1 Inference rules for property atom types

For a property there are two general inference rules, each of which has some special cases. These inference rules are sound in the semantics of HTRL, and their intuitive meaning is given in the special cases.

1. $< [u1, u2], [u3, u4] >: p, u2 \leq l2, l3 \leq u3 \vdash$
$< [l1, l2], [l3, l4] >: p$

2. $\{< [a1, a2], [a3, a4] >: p, a2 \leq a3\},$
$\{< [b1, b2], [b3, b4] >: p, b2 \leq b3\},$
$b2 \leq a3, a2 \leq b3, \vdash$
$\{< [l1, l2], [l3, l4] >: p, l2 = min(a2, b2),$
$l3 = max(a3, b3)\}$

Some Special cases of 1.[1]

1a. $< u2, u3 >: p,$
$u2 \leq u3, l2 \leq l3, u2 \leq l2, l3 \leq u3 \vdash$
$< l2, l3 >: p$

The meaning of this rule is that for every property which is true over a temporal interval $< u2, u3 >$, it is also true over every temporal interval $< l2, l3 >$ which is contained in $< u2, u3 >$ i.e.[2] {l2,...,l3}⊆{u2,...,u3}.

Example: From $< 0, 8 >: is\_working(mary)$ infer
$< 2, 6 >: is\_working(mary)$

1b. $< u2, u3 >: p,$
$u2 \leq u3, l3 < l2, u2 \leq l2, l3 \leq u3 \vdash$
$[l3, l2] : p$

This means that for every property which is true over a temporal interval $< u2, u3 >$, it is also true at every temporal instance $[l3, l2]$, which overlaps with this interval, i.e. {u2,...,u3} ∩ {l3,...,l2}≠ ∅

Example: From $< 0, 8 >: is\_working(mary)$ infer
$[4, 10] : is\_working(mary)$

1c. $[u3, u2] : p, u3 < u2, l2 \leq l3, l3 \leq u3, u2 \leq l2 \vdash$
$[l3, l2] : p$

This is an inconsistent set of constraints, and therefore it is not an inference rule.

1d. $[u3, u2] : p, u3 < u2, l3 < l2, l3 \leq u3, u2 \leq l2 \vdash$
$[l3, l2] : p$

This means that for every property which is true at a temporal instance $[u3, u2]$, it is also true at every temporal instance $[l3, l2]$, which contains the given temporal instance, i.e. {u3,...,u2}⊆{l3,...,l2}

Example: From $[4, 10] : is\_working(mary)$ infer
$[2, 11] : is\_working(mary)$

[1] The names u2, u3, l2, l3 are used in such a way so as to make clear the relation of special cases to the corresponding general case

[2] By {a,...,b} we represent an interval starting at a and finishing at b.

Some Special cases of 2.

2a. $\{< a2, a3 >: p, < b2, b3 >: p, b2 \leq a3, a2 \leq b3\} \vdash$
$< l2, l3 >: p, l2 = min(a2, b2),$
$l3 = max(a3, b3)$

The meaning of this rule is that for every property which is true over two overlapping intervals, it is also true over the concatenation of these intervals, i.e. {l2,...,l3}={a2,...,a3}∪{b2,...,b3}.

Example: From $< 0, 8 >: is\_working(mary)$ and
$< 6, 10 >: is\_working(mary)$ infer
$< 0, 10 >: is\_working(mary)$

2.2.2 Inference rules for event atom types

For an event there are two inference rules, each of which has also special cases (not presented here).

1. $< [u1, u2], [u3, u4] >: p, u2 \leq u3,$
$l3 \leq u2, u3 \leq l2 \vdash$
$< [l1, l2], [l3, l4] >: p$

i.e. for every event which is true over a temporal interval $< u2, u3 >$, it is also true at every temporal instance $[l3, l2]$, which contains the given temporal interval, i.e. {u2,...,u3} ⊆{l3,...,l2}.

2. $< [u1, u2], [u3, u4] >: p, u3 < u2,$
$l3 \leq u3, u2 \leq l2 \vdash$
$< [l1, l2], [l3, l4] >: p$

i.e. for every event which is true at a temporal instance $[u3, u2]$, it is also true at every temporal instance $[l3, l2]$, which contains the given temporal instance, i.e. {u3,...,u2}⊆{l3,...,l2}

3 Implementation Issues

An experimental implementation of the HTRL language and its inference system has been developed. The main idea of the implementation is to transform an HTRL program to a Constraint Logic Program (CLP). HTRL queries are also transformed to Constraint Logic Program queries.

3.1 Transforming an HTRL program to a CLP program

Each HTRL clause is transformed into one or more CLP clauses. In order to transform an HTRL clause we must first transform it to an HTRL canonical clause and consequently transform it into the corresponding a set of CLP clauses. The transformation of an HTRL clause into the corresponding CLP clause(s) is indepentent from the transformation of the other HTRL clauses of the same program. The main idea of this transformation is as follows:

Let C be an HTRL clause of the form:

$$T_0 : A_0(\overline{R_0}) \leftarrow T_1 : A_1(\overline{R_1}), ..., T_n : A_n(\overline{R_n})$$

where T_0,..., T_n are temporal references, and $\overline{R_0}$, ...$\overline{R_n}$ are tuples of arguments ($n \geq 0$). The canonical form of C is:

$$< [T_{01}, T_{02}], [T_{03}, T_{04}] >: A_0(\overline{R_0}) \leftarrow$$
$$< [T_{11}, T_{12}], [T_{13}, T_{14}] >: A_1(\overline{R_1}), ...,$$
$$< [T_{n1}, T_{n2}], [T_{n3}, T_{n4}] >: A_n(\overline{R_n})$$

This clause is transformed into one or more CLP clauses of the form:

$$A(W1, W2, W3, W4, \overline{R_0}) \leftarrow$$
$$consistency\_constraints\_for\_T_0,$$
$$consistency\_constraints\_for\_Ws,$$
$$resolution\_step\_constraints,$$
$$A_1(T_{11}, T_{12}, T_{13}, T_{14}, \overline{R_1}), ...,$$
$$A_n(T_{n1}, T_{n2}, T_{n3}, T_{n4}, \overline{R_n})$$

In each CLP clause, three types of constraints may be included: the *goal consistency constraints*, the *head consistency constraints* and the *resolution step constraints*.

The goal consistency constraints ensure the consistency of the instantiated temporal reference of the calling predicate. The head consistency constraints ensure the consistency of the instantiated temporal reference of the head predicate of the selected clause. The resolution step constraints implement the inference rules of HTRL, and therefore provide the means for making sound inferences. Resolution step constraints are produced from the constraints of the inference rules, after performing all possible compile time simplifications.

Example 2. In this example we transform two HTRL clauses of the HTRL program of example 1 into CLP clauses. The HTRL clause:

$$/ * 1980 : hire(mary, lecturer) * /$$

is transformed into the following CLP clause:

$$hire(W1, W2, W3, W4, mary, lecturer) \leftarrow$$
$$W1 \leq W2, W1 \leq W3, W3 \leq W4, W2 \leq W4,$$
$$1980 \leq W2, W3 \leq 1980.$$

while the HTRL clause:

$$< [T1, T2], [T3, T4] >: rank(Name, Rank) \leftarrow$$
$$[T1, T2] : hire(Name, Rank),$$
$$[T3, T4] : leave(Name, Rank).$$

is transformed into the following two clauses in the CLP program:

$$rank(W1, W2, W3, W4, Name, Rank) \leftarrow$$
$$T2 \leq T4, T3 \leq T4, T1 \leq T3, T1 \leq T2,$$
$$W1 \leq W2, W1 \leq W3, W3 \leq W4, W2 \leq W4,$$
$$T2 \leq W2, T3 < T2, W3 \leq T3,$$
$$hire(T1, T2, T1, T2, Name, Rank),$$
$$leave(T3, T4, T3, T4, Name, Rank).$$

$rank(W1, W2, W3, W4, Name, Rank) \leftarrow$
$T2 \leq T4, T3 \leq T4, T1 \leq T3, T1 \leq T2,$
$W1 \leq W2, W1 \leq W3, W3 \leq W4, W2 \leq W4,$
$W3 \leq T3, T2 \leq W2, T2 \leq T3,$
$hire(T1, T2, T1, T2, Name, Rank),$
$leave(T3, T4, T3, T4, Name, Rank).$

3.2 The Constraint Solver

An experimental Symbolic Constraint Solver (SCS) has been constructed for the manipulation of the constraints. SCS has been written in PROLOG and operates at the metalevel, i.e. the constraints are provided as data to the SCS program which solves or propagates them according to the instantiation of their arguments. Unresolved constraints are propagated to the next resolution step. Failure of the SCS to satisfy the constraints results to failure of the corresponding alternative clause of the HTRL program.

Example 3 (*continued from example 1*). In this example we provide queries to the transformed program of example 1, and receive the following answers.

$Query \quad :\leftarrow < T1, T2 >: rank(mary, R).$
$Answer :< 1980, 1983 >: rank(mary, lecturer).$
$Answer :< 1984, 1989 >: rank(mary, professor).$

$Query \quad :\leftarrow [T1, T2] : rank(N, professor).$
$Answer : [1985, 1988] : rank(mike, professor).$
$Answer : [1984, 1989] : rank(mary, professor).$

In general, the answer to a query consists of some values for the variables of the query together with a (possibly empty) simplified set S of output constraints. The constraints in S refer to the variables of the temporal references of the query. S is consistent (otherwise the query would have fail). Often, S can be expressed as a properly instantiated temporal reference (this is the case in the queries of the example 2), but sometimes this is not possible and S is returned as a simplified set of constraints.

The inference system that we have already implement for HTRL is sound but it is not (at present) complete. This means that not all HTRL atoms which are logical consequences of an HTRL program according to the semantics of the HTRL language, are provable by our system.

4 Conclusions

We have developed a temporal reasoning system, called HTRL, which is based on the semantics of a previously proposed temporal logic called TRL.

HTRL is a practical tool, as it handles time as a first order component, but it is also expressive enough to represent some kind of temporal uncertainty about the future and the past. Programming with HTRL resembles with Logic Programming, but it is possible to incorporate pure classical predicates (without any temporal references) into an HTRL program.

We are also working on practical and large examples concerning planning, temporal deductive databases, etc. We are planning to develop a complete inference system and also introduce measures of uncertainty.

References

[1] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23:123–154, 1984.

[2] C. Brzoska. Temporal logic programming and its relation to constraint logic programming. In V. Saraswat and K. Ueda, editors, *Logic Programming: Proc. of the 1991 International Symposium*, pages 661–677. MIT Press, 1991.

[3] C. Brzoska. Temporal logic programming with bounded universal modality goals. In D. S. Warren, editor, *Proc. of the Tenth International Conference on Logic Programming*, pages 239–256. MIT Press, 1993.

[4] K. T. Frantzi, T. Panayiotopoulos, and C. D. Spyropoulos. Extending Allen's relations for uncertain time points and uncertain intervals. In *7th Irish Conference on Artificial Intelligence and Cognitive Science (AICS'94)*, Sept. 1994.

[5] T. Fruehwirth. Temporal reasoning with constraint handling rules. Technical Report ECRC-94-05, ECRC, February 1994.

[6] D. M. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical foundations and computational aspects*. Clarendon Press-Oxford, 1994.

[7] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.

[8] M. A. Orgun and W. Ma. An overview of temporal and modal logic programming. In D. M. Gabbay and H. J. Ohlbach, editors, *Proc. of the First International Conference on Temporal Logics (ICTL'94)*, Lecture Notes in Artificial Intelligence (LNAI), Vol 827, pages 445–479. Springer-Verlag, 1994.

[9] T. Panayiotopoulos and C. D. Spyropoulos. Trl: A formal language for temporal references. In H. J. Olbach, editor, *Temporal Logic, Proceedings of the ICTL Workshop*, pages 99–109, 1994. MPI-I-94-230.

[10] L. Vila. A survey on temporal reasoning in artificial intelligence. *AI Communications*, 7(1):4–28, 1994.