TRL: A Temporal Reasoning System and its applications

Themis Panayiotopoulos

Knowledge Engineering Laboratory,
Department of Informatics, University of Piraeus,
80 Karaoli & Dimitriou Str.,
185 34 Piraeus, Greece,
e-mail: themisp@unipi.gr,

Abstract

This work presents the TRL temporal reasoning system and investigates its ability to satisfy the requirements posed by complex application domains. Issues concerning the syntax, the semantics and the inference rules of the system are presented in depth. It is also investigated how complex temporal requirements can be handled by TRL. The concepts of temporal points, temporal instances and temporal intervals, extended predicates (properties or events) are introduced. TRL has a syntax and semantics very similar to that of Prolog, and it can therefore be easily used as the basic temporal deductive component of Intelligent Information Systems.

Keywords: Temporal reasoning, Temporal Logic Programming, Constraint Logic Programming

1. Introduction

Temporal reasoning has attracted many researchers during the last two decades. Temporal systems and programming languages seem to be very expressive and have been widely used for describing inherently dynamic systems. Intense research has resulted to the development of Temporal Logics [All84, Gal87a, Ost89, Jix94, Fru94, Org94, Vil94, Ger96, Ron98] which have found application in many domains, such as temporal planning, temporal databases, verification of concurrent systems, VLSI design, etc. Most of the implemented temporal logics are implementations of modal temporal logics [Org94].

The proposed approaches may be divided into two main categories: numerical [Bar93, Bod93, Ger93, Sti93] and symbolic approaches. The symbolic approaches may be further divided in classic logic [Kow86], modal logic [Gal87a, Gal87b, Ost89] and reified logic approaches [All84, Rei87]. Reified logic approaches tend to keep temporal and non-temporal components separate, and combine the naturalness of expression of modal temporal logics with the efficiency of first order temporal logics [Rei87].

In most systems however, the concept of uncertainty is not considered. An exception is the Tachyon system, [Sti93], where uncertainty regards the exact time and duration of events. Fuzzy temporal reasoning is examined in [Dub89].

Quite recently, Disjunctive Chronolog [Ger96], a disjunctive temporal logic programming language which extends the Chronolog language, [Wad88], has appeared. Disjunctive Chronolog expresses dynamic behaviour as well as uncertainty through temporal disjunctions.

Branching time logic programming languages have also been proposed. A representative is the Cactus language, [Ger97, Ron98], which expresses in a natural way algorithms that involve the manipulation of tree data structures.
The development of the theory of constraint logic programming has brought new ideas and tools which seem to face the problem of temporal reasoning, both theoretically and practically, in a more efficient and expressive manner [Fru94].

In this paper we present the TRL temporal reasoning system as an alternative approach to representing and reasoning about time. The TRL system is a result of recent research on temporal reasoning and has been applied on various areas of computer science but mainly on temporal planning and temporal databases.

The paper is structured as follows: in the next section we discuss the motivation and previous research concerning TRL. We consequently introduce the syntax of TRL sentences, its model-theoretic semantics as well as its first order translation. Section 4 presents deduction and the inference rules of the system. In section 5 we present the TRLi meta-interpreter, an example of its use, some of its applications and discuss its expressive power. Finally, we conclude and present future research work.

2. Motivation and Previous Research concerning TRL

2.1. A short review on ‘TRL systems’ and their applications

The TRL Temporal Reference Language, [Pan94], was proposed as a temporal logic which tried to unify the notions of temporal points and temporal intervals using a single temporal element called a temporal reference. A formal logic-based temporal language was needed which would fill in the gap, i.e. provide a common framework for representing both intervals and points with a single notation, consider the notion of temporal uncertainty, and preserve the important properties of point and interval based logics.

Each TRL sentence was a classic first order logic sentence labelled with a temporal reference that was defining the temporal aspect of the sentence. TRL proved to be very complex, an almost second order temporal logic. It was second order as far as temporal elements were concerned but first order in all other aspects. It was disjunctive from some points of view and an adequate inference system had not been found for it.

Later on, some of TRL’s requirements were reduced, and a ‘Horn’ subset of the language was isolated retaining and strengthening the most interesting of its temporal properties. This subset was called HTRL [Pan95a]. Along with HTRL, a method for transforming an HTRL program to an equivalent constraint logic program was also developed. At the same time, following the ideas proposed in [Pan95a], a TRL meta-interpreter was implemented and it was applied to some simple but nevertheless interesting examples from the areas of intelligent problem solving, simulation as well as temporal databases [Pan95b].

It was already clear that the developed system carried great resemblance to other constraint based approaches [Fru94] and should therefore be dependent on constraint logic programming. A more adaptive scheme was therefore adopted, and the developed constraint solver was replaced by calls to the constraint solver of the host language using a CLP language as a basis (i.e. Sicstus Prolog). In this way the TRL interpreter was much simplified and we had the opportunity to concentrate on the temporal aspects of the system.

2.2. Temporal References

In TRL, a dynamically changing but still linear world is assumed. Time in TRL has the following properties:

• Time is discrete.
• Time points are totally ordered.
• An interval is a convex set of time points.
• Time is unbounded in both directions (in the past and in the future).
• Time is considered to be certain or uncertain.

Some assertions hold at a specific time point (temporal point), some other hold during a whole interval (temporal interval). There are cases however, in which there is lack of complete information due to the vagueness of natural language or lack of precise information about the exact time of the occurrence of a fact. As we are also interested to represent such cases, we must provide the means to express a fact that occurred or is going to occur at some moment between two temporal points. Two such temporal points form a temporal instance, i.e. an uncertain temporal point. Moreover, in order to represent a fact occurring over a temporal interval with uncertain start and end points, we must use an uncertain temporal interval, i.e. an interval with temporal instances for its start or/and end points. A temporal reference is constructed by using temporal constants (integers), temporal variables and the temporal constructors $<$, $>$, $[\,\,]$. 

**Definition 1. (Temporal Point).** A Temporal point is defined as follows:

• A temporal variable is a temporal point.
• A temporal constant is a temporal point.
• If $T$ is a temporal variable and $n$ is an integer then, $T+n$, $T-n$ are temporal points.

Temporal points of the form $T+n$, $T-n$ are compound temporal points which are evaluable terms when $T$ becomes instantiated. Such terms can be parts of temporal constraints and the temporal constraint solver takes care of their evaluation.

**Definition 2. (Temporal Interval).** A Temporal interval is an expression of the form $\langle T_1,T_2 \rangle$, where $T_1$, $T_2$ are temporal points.

**Definition 3. (Temporal Instance).** A Temporal instance is an expression of the form $[T_1,T_2]$, where $T_1$, $T_2$ are temporal points.

**Definition 4. (Uncertain Temporal Interval).** An uncertain temporal interval is an expression of the form $\langle [T_1,T_2],[T_3,T_4] \rangle$, $\langle T_1,[T_2,T_3] \rangle$, $\langle [T_1,T_2],T_3 \rangle$, where $T_1$, $T_2$, $T_3$, $T_4$ are temporal points.

**Definition 5. (Temporal Reference).** A Temporal Reference $Tref$, is either a temporal point, or a temporal interval, or a temporal instance or an uncertain temporal interval.

**Definition 6. (Consistent Temporal Reference).** A temporal reference is Consistent if the following holds:

• $Tref$ is a temporal point.
• $Tref$ is of the form $\langle [T_1,T_2],[T_3,T_4] \rangle$ and it holds $T_1 \leq T_2$, $T_1 \leq T_3$, $T_3 \leq T_4$, $T_2 \leq T_4$.
• $Tref$ is of the form $\langle T_1,[T_3,T_4] \rangle$ and it holds $T_1 \leq T_3$, $T_3 \leq T_4$.
• $Tref$ is of the form $\langle [T_1,T_2],T_3 \rangle$ and it holds $T_1 \leq T_2$, $T_2 \leq T_3$.
• $Tref$ is of the form $\langle T_1,T_2 \rangle$ and it holds $T_1 \leq T_2$
• $Tref$ is of the form $[T_1,T_2]$ and it holds, $T_1 \leq T_2$

The constraints that ensure the consistency of a temporal reference are called consistency constraints.
Definition 7. (Temporal Reference Canonical Form). Every simpler form can be expressed as \(<[T_1,T_2],[T_3,T_4]>\) and we call this form a Temporal Reference Canonical Form.
- \(<[T,T],[T,T]>\) is a temporal point T.
- \(<[T_1,T_1],[T_2,T_2]>\) is a temporal interval \(<T_1,T_2>\).
- \(<[T_1,T_2],[T_1,T_2]>\) is a temporal instance \([T_1,T_2]\).
- \(<[T_1,T_1],[T_3,T_4]>\) is an uncertain temporal interval \(<T_1,[T_3,T_4]>\).
- \(<[T_1,T_2],[T_3,T_3]>\) is an uncertain temporal interval \(<[T_1,T_2],T_3]>\).

3. Syntax and Semantics of TRL

3.1. Syntax of TRL programs

Temporal information is expressed through temporal references which are labels of atoms (predicates). In any other respect, TRL uses the well known syntax of Prolog. Atoms referenced in such a way are called extended atoms (predicates). There are three categories of atoms:
- classic atoms, which bear no temporal label on them
- extended atoms, which are further subcategorised into properties and events and have the form \(Tref:A\) (e.g. 25/08/1998:arrives(costas, corfu)), and
- temporal atoms., which are temporal constraints.

Definition 8. (TRL clause, program and query).
A TRL clause is an expression of the form \(A_0 \leftarrow A_1, \ldots, A_n\) where \(A_i, (i \geq 0)\) are TRL atoms. A TRL program is a set of TRL clauses. A TRL query is a formula of the form \(\leftarrow Q_1, \ldots, Q_n\) where \(Q_i, (i \geq 1)\) are TRL atoms.

3.2. Property and event atoms

Properties and events have been long ago introduced by Allen, but they are considered here as basic entities of TRL’s logic. Although their syntax is identical, their semantics is different.

Properties are relations which preserve their characteristics in subintervals, i.e. when a property is true during a temporal interval it is also true during all subintervals of this interval. The relation ‘lives’ in the statement \(<1918,1987>:lives(peter)>\ should be considered as a property as the object ‘peter’ had the property of being a ‘living creature’ not only from the moment it was born to the moment it died but also for all the intermediate time points.

Properties use temporal points as their basic temporal element. Therefore we will use temporal points to define their behaviour over the other temporal references. When a property is true at a temporal point \(t\), then it is only known to be true at the moment \(t\). As an example consider the assertion \(8:00 \text{ am} :leaving(bus)\) and its meaning ‘The bus is leaving at 8:00 am’.

When a property is true over an interval it is necessarily true at every time point belonging in this interval: \(<T_1,T_2>:laughs(stella) \leftarrow <T_1,T_2>:watches(stella, funny_film)\). Meaning ‘Stella didn’t stop laughing while watching the funny film’.

When a property is true at some time during a temporal instance \([t_1,t_2]\), then there is at least one time point \(t\) between \(t_1\) and \(t_2\), at which the property is true. ‘Nikos arrives between 8:00 and 10:00’ is written as \([8:00 \text{ am},10:00 \text{ am}]:\text{arrives(nikos)}\). Nikos will arrive at some moment between 8:00 and 10:00. ‘Maria will be cooking between 8:00 and 10:00’ is written as \([8:00 \text{ am},10:00 \text{ am}]:\text{cooks(maria)}\). Maria will be cooking for half an hour at some time between 8:00 and 10:00.
If a property is true during an uncertain temporal interval \([s_1,s_2],[s_3,s_4]\) then there exists a time point \(t_1\) between \(s_1\) and \(s_2\) and a time point \(t_2\) between \(s_3\) and \(s_4\), such that the relation is true during the temporal interval \([t_1,t_2]\). For example, the statement ‘John stayed at the beach from some time between 9.00 am to 11.00 am till 5.00 pm’ can be expressed in TRL as \([\langle 9.00\, am, 11.00\, am \rangle,\, 5.00\, pm\rangle : stay(john,\text{beach})\). The meaning of such a statement is that there is some time point, say 10:30 am, at which John arrived at the beach and he stayed there till late in the afternoon (5:00 pm).

Properties can also be used when we want to describe relations, activities, phenomena which change over time, but we are not interested to relate them to some changing quantity of a resource: eating, running, working, dancing, etc.

Events are relations which do not necessarily preserve their characteristics inside smaller temporal intervals. For example, in the statement \([1980,2000] : published_papers(george,50)\) the relation ‘published_papers’ does not necessarily preserve its truth value during the temporal interval \([1980,1990]\) as George may have had only 20 papers published from 1980 to 1990.

Events use temporal intervals as their basic temporal element. Therefore we will use temporal intervals to define their behaviour over the other temporal references. Temporal points, however also exist for events as they are special cases of temporal intervals.

When an event is true during a temporal interval \([t_1,t_2]\), then it is only known to be true at this temporal interval, i.e. events handle temporal intervals as properties handle temporal points. The rationale behind events is that an interval over which an event holds can be considered as a ‘wide temporal point’ which cannot be further divided. When an event is true over an interval, then it is not necessarily true over all subintervals of this interval: ‘Joice has accumulated \$10000 during the last two years’, encoded in TRL as \([1/1/1993,1/1/1995] : accumulates(\text{joice},\text{\$10000})\). It is obvious that this does not hold over temporal intervals contained in the given interval, e.g. ‘Joice has not accumulated \$10000 during the last five weeks’.

When an event is true at some time during a temporal instance \([t_1,t_2]\), then there is at least one temporal interval \([s_1,s_2]\), between \(t_1\) and \(t_2\), at which the event is true. ‘At some time during the last hour he has run 10 times around the square’, \([8:00,9:00] : times\_around(\text{square},10)\), is true because ‘he run 10 times around the square from 8:30 to 8:45’.

If an event is true during an uncertain temporal interval \([s_1,s_2],[s_3,s_4]\) then there exists a time point \(t_1\) between \(s_1\) and \(s_2\) and a time point \(t_2\) between \(s_3\) and \(s_4\), such that the relation is true during the temporal interval \([t_1,t_2]\). For example, the statement ‘John ate 5 sandwiches on the beach from some time between 9.00 am to 9.30 am till 10.00 pm’ can be encoded in TRL as \([\langle 9.00\, am, 9.30\, am \rangle,\, 10.00\, pm\rangle : eat(john,\text{5},\text{sandwich})\). The meaning of such a statement is that there is some time point, say 9:15 am, at which John started eating the sandwiches and he finished eating them at 10:00 am.

Events are used when we want to relate the described ability to a changing quantity of some kind of resource and this quantity is included as a parameter of the relation: accumulating money, covering distances, eating a specific amount of food, etc.

3.3. Model Theoretic semantics of TRL Logic

**Definition 9.** A temporal interpretation \(I\) of the temporal logic TRL comprises of 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 \([D^n \to D]\); for each property predicate symbol, an element of \([N \to 2^D]\); for each event predicate symbol, an element of \([S \to 2^D]\), where \(S\) is the subset of \(NxN\) with pairs \(<S_1,S_2>\) such that \(S_1\leq S_2\).
D contains the standard constant symbols, integers and pairs of integers. The satisfaction relation \(|=\) is defined in terms of temporal interpretations; \(|=_{t}\). A denotes that a formula A is true at a moment t given some temporal interpretation I. On the other hand, \(|=_{<t_{1},t_{2}>}\) A denotes that a formula A is true at an interval \(<t_{1},t_{2}>\) given some temporal interpretation I. This latter notation is used for event predicates which handle intervals as 'wide temporal points' (they cannot look further into them). Sometimes we use the notation \(\{t_{1},...,t_{n}\}\) to denote the set of consecutive integers starting from \(t_{1}\) and ending to \(t_{n}\), i.e. \(\{t_{1},...,t_{n}\} = \{t_{1}, t_{1}+1, ..., t_{n} - 1, t_{n}\}\).

**Definition 10. Basic Model Theoretic Semantics**

1. If \(f(a_{1},a_{2},...,a_{n})\) is a term then \(I(f(a_{1},a_{2},...,a_{n})) = I(f)(I(a_{1}),I(a_{2}),...,I(a_{n}))\)
2. For any n-ary predicate symbol \(p\) with terms \(a_{1},a_{2},...,a_{n}\),
   - Classical predicates are not temporally interpreted:
     \(|=_{t_{1}} c(a_{1},a_{2},...,a_{n})\) iff \(<t_{1},I(a_{1}),I(a_{2}),...,I(a_{n})>\) \(\in I(c)\)
   - Property predicates are interpreted over temporal points:
     \(|=_{t_{1}} p(a_{1},a_{2},...,a_{n})\) iff \(<t_{1},I(a_{1}),I(a_{2}),...,I(a_{n})>\) \(\in I(\{t\})\)
   - Event predicates are interpreted over temporal intervals:
     \(|=_{<t_{1},t_{2}>} e(a_{1},a_{2},...,a_{n})\) iff \(<t_{1},I(a_{1}),I(a_{2}),...,I(a_{n})>\) \(\in I(\{t_{1},...,t_{2}\})\)
   - Temporal predicates are temporal constraints
     \(|=_{t_{1}} \neg A\) iff it is not the case that \(|=_{t_{1}} A\)
     \(|=_{t_{1}} A \land B\) iff \(|=_{t_{1}} A\) and \(|=_{t_{1}} B\)
     \(|=_{t_{1}} A \lor B\) iff \(|=_{t_{1}} A\) or \(|=_{t_{1}} B\)
     \(|=_{t_{1}} (\forall X) A\) iff \(|=_{[d,X]} A\) for all \(d \in D\) where the interpretation \(I[d/X]\) is identical to I except that the variable \(X\) is assigned the value \(d\).
     \(|=_{t_{1}} (\exists X) A\) iff \(|=_{[d,X]} A\) for some \(d \in D\) where the interpretation \(I[d/X]\) is identical to I except that the variable \(X\) is assigned the value \(d\).

**Definition 11. Model theoretic semantics for properties**

If \(p\) is a property, then

11. If \(S \in N\) then \(|=_{t_{1}} S:p\) iff \(|=_{t_{1}} S\)
12. \(|=_{t_{1}} (\forall S) S:p\) iff for all \(S \in N\), \(|=_{t_{1}} S:p\)
13. \(|=_{t_{1}} (\exists S) S:p\) iff for some \(S \in N\), \(|=_{t_{1}} S:p\)
14. \(|=_{t_{1}} <S_{1},S_{2}>:p\) iff for all \(S \in \{S_{1},S_{2}\}\), \(|=_{t_{1}} S:p\)
15. \(|=_{t_{1}} [S_{1},S_{2}]:p\) iff for some \(S \in \{S_{1},...,S_{2}\}\), \(|=_{t_{1}} S:p\)
16. \(|=_{t_{1}} <S_{1},S_{2}],[S_{3},S_{4}]:p\) iff for some \(T_{1} \in \{S_{1},S_{2}\}\), for some \(T_{2} \in \{S_{3},...,S_{4}\}\), \(|=_{t_{1}} <T_{1},T_{2}>:p\)

**Example 1. Assume p(a) is a property predicate. It is interesting to demonstrate a property of these semantics through examples:**

(a) \(<3,5>:p(a)\) \(|= 3:p(a) \land 4:p(a) \land 5:p(a)\)
(b) \([3,5]:p(a)\) \(|= 3:p(a) \lor 4:p(a) \lor 5:p(a)\)
(c) \(<[3,5],[8,9]:p(a)\) \(|= <3,8>:p(a) \lor <3,9>:p(a) \lor <4,8>:p(a) \lor <4,9>:p(a) \lor <5,8>:p(a) \lor <5,9>:p(a)\). Notice that due to the nature of the terms of the formula it is equivalent to \(<5,8>:p(a)\). (Remember the property \(F \lor (F \land Q)\) \(|= F\), and the fact that each term of the formula logically entails \(<5,8>:p(a)\).
(d) Also, \(<[3,5],[4,5]:p(a)\) \(|= <3,4>:p(a) \lor <3,5>:p(a) \lor <4,4>:p(a) \lor <4,5>:p(a) \lor <5,5>:p(a)\). For similar reasons the formula it is equivalent to \(<4,5>:p(a)\).

The above properties can be easily generalised to:
If $S_1 \leq S_2 \leq S_3 \leq S_4$ then $[S_1, S_2], [S_3, S_4] \models A \models [S_2, S_3] : A$

We are going to use the letters $D_i$ for temporal elements which are part of the source of information (Data) and $G_i$ for temporal elements which are part of the derived information (Goal).

If $[D_1, ..., D_2] \subseteq [G_1, ..., G_2]$ and $[D_3, ..., D_4] \subseteq [G_3, ..., G_4]$,

i.e. $G_1 \leq D_1 \land D_2 \leq G_2 \land G_3 \leq D_3 \land D_4 \leq G_4$

then $[D_1, D_2], [D_3, D_4] : A \models [G_1, G_2], [G_3, G_4] : A$

Proof

$[D_1, D_2], [D_3, D_4] : A$

$\models [D_1, D_3] : A \lor [D_1, D_3+1] : A \lor \ldots \lor [D_1, D_4] : A \lor \ldots \lor [D_2, D_4] : A$

$\models [G_1, G_2], [G_3, G_4] : A$

Given that $A \models A \lor B$, and the fact that $[G_1, G_2], [G_3, G_4] : A$ contains the above terms as well as (possibly) other terms.

This is also a very interesting property which could be used as a general inference rule. Using similar techniques it is easy to prove the following properties:

If $[D_1, ..., D_2] \subseteq [G_1, ..., G_2]$, i.e. $D_1 \leq G_1 \land G_2 \leq D_2$ then $[D_1, D_2] : A \models [G_1, G_2] : A$

Rationale: Given that $A$ is true throughout $[D_1, D_2]$ it will also be true in all subintervals of it.

If $[D_1, ..., D_2] \cup [D_3, ..., D_4] \supseteq [G_1, ..., G_2]$

then $[D_1, D_2] : A \land [D_3, D_4] : A \models [G_1, G_2] : A$

Rationale: Given that $A$ is true on two overlapping intervals, then it will also be true on their union, and subsequently on any subinterval of the union.

If $[R_1, ..., R_2] \cap [G_1, ..., G_2] \neq \emptyset$, i.e. $R_1 \leq G_2 \land G_1 \leq R_2$ then $[R_1, R_2] : A \models [G_1, G_2] : A$

Rationale: Given that $A$ is true over an interval $[D_1, D_2]$, then it is true at every temporal point. Therefore, at any temporal instance $[G_1, G_2]$ intersecting with the interval $A$ is true (for the temporal instance).

If $[D_1, ..., D_2] \subseteq [G_1, ..., G_2]$, i.e. $G_1 \leq D_1 \land D_2 \leq G_2$ then $[D_1, D_2] : A \models [G_1, G_2] : A$

Rationale: Given that $A$ is true at some time between $D_1$ and $D_2$, then there must be some temporal points or interval in it which make it true. Therefore it will be also true at some time between $G_1$ and $G_2$ which form a superset of $\{D_1, ..., D_2\}$.

Definition 12. Model theoretic semantics for events

If $e$ is an event, and $S = \langle S_1, S_2 \rangle \in N \times N$, $S_1 \leq S_2$, then

$\models_{i_1} S: e$ if $\models_{i_5} e$, in the case of $\langle t, t \rangle$ for simplicity we write $t$.

$\models_{i_1} (VS) S: e$ if $e$ for all $S \models_{i_5} e$

$\models_{i_1} (3S) S: e$ if $e$ for some $S \models_{i_5} e$

$\models_{i_1} \langle S_1, S_2 \rangle: e$ if $e$ for some $\langle S_3, ..., S_4 \rangle \subseteq \langle S_1, ..., S_2 \rangle$, $\models_{i_5, S_3} e$

$\models_{i_1} \langle S_1, S_2 \rangle, \langle S_3, S_4 \rangle: e$ if $e$ for some $T_1 \in \{S_1, ..., S_2\}$, for some $T_2 \in \{S_3, ..., S_4\}$, $\models_{i_5, T_1, T_2} e$
Example 2. Assume e(a) is an event. Again, we will demonstrate the semantics through examples. Notice that formulas with events cannot be further decomposed or simplified.

(a) \[3,5]e(a) \models <3,5>:p(a) \lor <3,5>:p(a) \lor <4,5>:p(a) \lor <5,5>:p(a)\]

(b) \([3,5],[8,9]e(a) \models <3,8>:p(a) \lor <3,9>:p(a) \lor <4,8>:p(a) \lor <4,9>:p(a) \lor <5,8>:p(a) \lor <5,9>:p(a)\]

(c) \([3,5],[4,5]p(a) \models <3,4>:p(a) \lor <3,5>:p(a) \lor <4,4>:p(a) \lor <4,5>:p(a) \lor <5,5>:p(a)\]

Notice that formulas (1) and (2) are not valid in the case of event predicates. It is clear however that there are two special cases of (1) and (2) which are valid for events:

(8) If \(S_1=S_2 \leq S_3=S_4\) then \([S_1,S_2],[S_3,S_4]:A \models <S_2,S_3>:A\)

(9) If \(S_1=S_3 \leq S_2=S_4\) then \([S_1,S_2],[S_3,S_4]:A \models [S_2,S_3]:A\)

Similarly to (3) we can also prove for events that:

(10) If \([D_1,...,D_2] \subseteq [G_1,...,G_2]\) and \([D_3,...,D_4] \subseteq [G_3,...,G_4]\), i.e. \(G_1 \leq D_1 \land D_2 \leq G_2 \land G_3 \leq D_3 \land D_4 \leq G_4\), then \([D_1,D_2],[D_3,D_4]:A \models [G_1,G_2],[G_3,G_4]:A\)

This is a property which could be used as a general inference rule.

(11) If \([D_1,...,D_2] = [G_1,...,G_2]\), i.e. \(D_1 = G_1 \land D_2 = G_2\) then \((\forall T) (S_1 \leq T \leq S_2 \Rightarrow T:p)\)

Rationale : Given that A is true throughout \([D_1,D_2]\) it will also be true only in equal intervals.

(12) If \([D_1,...,D_2] \subseteq [G_1,...,G_2]\), i.e. \(G_1 \leq D_1 \land D_2 \leq G_2\) then \((\forall T) (S_1 \leq T \leq S_2 \Rightarrow T:p)\)

Rationale : Given that A is true over an interval \([D_1,D_2]\), then at any temporal instance \([G_1,G_2]\) containing the interval A is also true (for the temporal instance).

(13) If \([D_1,...,D_2] \subseteq [G_1,...,G_2]\), i.e. \(G_1 \leq D_1 \land D_2 \leq G_2\) then \((\exists T) (S_1 \leq T \leq S_2 \land T:p)\)

Rationale : Similar to (7)

Clearly, the TRL temporal logic programming language as defined in section 2 is a ‘Horn subset’ of the TRL Logic.

3.4. First Order Translation

First order translation is very helpful as it provides an alternative meaning. In the sequel we provide such a first order translation and prove through it some interesting formulas both for property and event atoms.

Table 1 : First order translation of Property assertions

<table>
<thead>
<tr>
<th>Temporal Reference</th>
<th>Property Predicate</th>
<th>First Order Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temporal point</td>
<td>(S:p)</td>
<td>(p) is true at (S)</td>
</tr>
<tr>
<td>Temporal interval</td>
<td>(&lt;S_1,S_2&gt;:p)</td>
<td>((\forall T) (S_1 \leq T \leq S_2 \Rightarrow T:p))</td>
</tr>
<tr>
<td>Temporal instance</td>
<td>([S_1,S_2]:p)</td>
<td>((\exists T) (S_1 \leq T \leq S_2 \land T:p))</td>
</tr>
<tr>
<td>Uncertain Temporal interval</td>
<td>(&lt;[S_1,S_2],[S_3,S_4]:p)</td>
<td>((\exists T_1) (\exists T_2) (S_1 \leq T_1 \leq S_2 \land S_3 \leq T_2 \leq S_4 \land &lt;T_1,T_2&gt;:p))</td>
</tr>
</tbody>
</table>
Table 2: First order translation of Event assertions

<table>
<thead>
<tr>
<th>Temporal Reference</th>
<th>Event Predicate</th>
<th>First order Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temporal point</td>
<td>S:e</td>
<td>e is true at S, same as &lt;S,S&gt;</td>
</tr>
<tr>
<td>Temporal interval</td>
<td>&lt;S1,S2&gt;:e</td>
<td>e is true at &lt;S1,S2&gt;</td>
</tr>
<tr>
<td>Temporal instance</td>
<td>[S1,S2]:e</td>
<td>(3T1)(3T2) (S1≤T1≤T2≤S2 ∧ &lt;T1,T2&gt;:e)</td>
</tr>
<tr>
<td>Uncertain Temporal interval</td>
<td>[S1,S2],[S3,S4]:e</td>
<td>(3T1)(3T2) (S1≤T1≤S2 ∧ S3≤T2≤S4 ∧ T1≤T2 ∧ &lt;T1,T2&gt;:e)</td>
</tr>
</tbody>
</table>

Using these two translations we will prove that every temporal reference has an equivalent canonical temporal reference form (Definition 7). Assume p to be a property, e to be an event.

Properties

If S1=S2=S3=S4=S, then <[S,S],[S,S]>:p ↔ S:p

(3T1) (3T2) (S≤T1≤S ∧ S≤T2≤S ∧ <T1,T2>:p), therefore T1=T2=S ↔ <S,S>:p

If S1=S2=S and S3=S4=E, then <[S,S],[E,E]>:p ↔ <S,E>:p

(3T1) (3T2) (S≤T1≤S ∧ E≤T2≤E ∧ <T1,T2>:p), i.e. T1=S, T2=E, ↔ <S,E>:p

If S1=S=S and S2=S4=E, then <[S,E],[S,E]>::p ↔ [S,E]:p

(3T1) (3T2) (S≤T1≤S ∧ S≤T2≤E ∧ <T1,T2>:p), and without loss of generality. T1=T2=K, ↔ (3K) (S≤K≤E ∧ <K,K>:p), which leads to (3T) (S≤T≤E ∧ T:p) ↔ [S,E]:p

Events

If S1=S2=S3=S4=S, then <[S,S],[S,S]>:e ↔ S:e

(3T1)(3T2) (S≤T1≤S ∧ S≤T2≤S ∧ T1≤T2 ∧ <T1,T2>:e), i.e. T1=T2=S and produces <S,S>:e

If S1=S2=S and S3=S4=E, then <[S,S],[E,E]>:p ↔ <S,E>:p

(3T1)(3T2) (S≤T1≤S ∧ E≤T2≤E ∧ T1≤T2 ∧ <T1,T2>:e), i.e. T1=S, T2=E producing <S,E>:e

If S1=S=S and S2=S4=E, then <[S,E],[S,E]>:e ↔ [S,E]:e

(3T1)(3T2) (S≤T1≤S ∧ S≤T2≤E ∧ T1≤T2 ∧ <S,E>:e) ↔ (3T1)(3T2) (S≤T1≤T2 ≤ S ∧ <S,E>:e) ↔ [S,E]:e

4. Deduction and Inference rules

For classical atoms, we retain SLD-resolution, i.e. TRL behaves like Prolog. For extended atoms, however, we have defined additional inference rules. These rules are sound as they are straightforward applications of the properties discussed in a previous section.

4.1. Inference rules of properties.

[P1] {<d1,d2>,[d3,d4]>:p, d2≤g2, g3≤d3} infers <[g1,g2],[g3,g4]>:p

Proof of Soundness Using the (1)-(7) relations we can prove

a) If d2≤d3, g2≤g3 then we have the ordering d2≤g2≤g3≤d3 and therefore

<[d1,d2],[d3,d4]>:p =<[d2,d3]>:p =<[g2,g3]>:p =<[g1,g2],[g3,g4]>:p

(the ‘goal interval’ g2,g3 being a subset of the ‘source interval’ d2,d3)

b) If d2≤d3, g3≤g2 then we have

1) the ordering d2≤g2≤g3≤d3 and therefore

<[d1,d2],[d3,d4]>:p =<[d2,d3]>:p =<[g3,g2]>:p =<[g1,g2],[g3,g4]>:p

254
(the ‘goal instance’ \([g_3,g_2]\) being a subset of the ‘source interval’ \([d_2,d_3]\))

b2) the ordering \(d_2\leq g_3\leq d_3\leq g_2\) or \(g_3\leq d_2\leq g_2\leq d_3\) and therefore
\(\langle[d_1,d_2],[d_3,d_4]\rangle:p \models \langle d_2,d_3\rangle:p \models [g_3,g_2]:p \models [g_1,g_2],[g_3,g_4]:p\)
(the ‘goal instance’ \([g_3,g_2]\) intersecting with the ‘source interval’ \([d_2,d_3]\))

b3) the ordering \(g_3\leq d_2\leq d_3\leq g_2\) and therefore
\(\langle [d_1,d_2],[d_3,d_4]\rangle:p \models \langle d_2,d_3\rangle:p \models [g_3,g_2]:p \models [g_1,g_2],[g_3,g_4]:p\)
(the ‘goal instance’ \([g_3,g_2]\) being a superset of the ‘source interval’ \([d_2,d_3]\))

c) If \(d_3\leq d_2\leq g_3\) then \(d_3\leq d_2\leq g_2\leq g_3\leq d_3\), valid only when all equal.

d) If \(d_3\leq d_2\leq g_3\leq g_2\) then we have the ordering \(g_3\leq d_3\leq d_2\leq g_2\) and therefore
\(\langle [d_1,d_2],[d_3,d_4]\rangle:p \models [d_3,d_2]:p \models [g_3,g_2]:p \models [g_1,g_2],[g_3,g_4]:p\)
(the ‘goal instance’ \([g_3,g_2]\) being a superset of the ‘source instance’ \([d_3,d_2]\)).

\([P2]\) \(\langle [a_1,a_2],[a_3,a_4]\rangle:p \models \langle [b_1,b_2],[b_3,b_4]\rangle:p, a_2\leq a_3, b_2\leq b_2, b_2\leq a_3+1, a_2\leq b_3+1\)
\(\textbf{infers} \ \langle [c_1,c_2],[c_3,c_4]\rangle:p \text{ where } c_2=\min(a_2,b_2), c_3=\max(a_3,b_3)\)

**Proof of Soundness**

Clearly, given that \(a_2\leq a_3\) then \(\langle [a_1,a_2],[a_3,a_4]\rangle:p \models \langle a_2,a_3\rangle:p\).

Similarly given that \(b_2\leq b_3\) then \(\langle [b_1,b_2],[b_3,b_4]\rangle:p \models \langle b_2,b_3\rangle:p\).

The constraints \(a_2\leq a_3+1\) and \(a_2\leq b_3+1\) ensure that \([a_2,\ldots,a_3]\) and \([b_2,\ldots,b_3]\) are consecutive.

The result is obvious given the property (5).

**Examples of special cases**

\([P1.1]\) \(\langle d_2,d_3\rangle:p, d_2\leq g_2\leq g_3\leq d_3\) \textbf{infers} \(g_2,g_3\rangle:p, \)
\(\text{Example} : <8,14> : \text{works(manolis)} \textbf{infers} <10,12> : \text{works(manolis)}\)

\([P1.2]\) \(\langle d_3,d_2\rangle:p, g_3\leq d_3\leq d_2\leq g_2\) \textbf{infers} \(g_3,g_2\rangle:p, \)
\(\text{Example} : <8,9> : \text{eat_breakfast} \textbf{infers} <7,10> : \text{eat_breakfast}\)

\([P2.1]\) \(\langle a_2,a_3\rangle:p, \langle b_2,b_3\rangle:p, b_2\leq a_3+1, a_2\leq b_3+1\) \textbf{infer} \(\langle c_2,c_3\rangle:p, \text{ where } c_2=\min(a_2,b_2), \)
\(c_3=\max(a_3,b_3). \)

\(\text{Example} : \text{From} <1,5> : \text{studies}, <3,8> : \text{studies}, \text{infer} <1,8> : \text{studies}\)

**4.2. Inference rules of events.**

\([E1]\) \(\langle d_1,d_2\rangle,p, d_3\leq d_1, d_2\leq g_2, g_3\leq d_3, d_4\leq g_4\) \textbf{infers} \(\langle g_1,g_2\rangle,p\)

**Proof of Soundness** Immediately from (8)

**Examples of special cases**

\([E1.1]\) \(\langle d_2,d_3\rangle:p, d_2=g_2\leq g_3=d_3\) \textbf{infers} \(g_2,g_3\rangle:p, \)
\(\text{(event intervals do not retain the truth values inside them)}\)
\(\text{Example} : <8,9> : \text{speed_up(car)} \textbf{infers} <8,9> : \text{speed_up(car)}\)
\(\text{(From the 8th second to the 9th second the car was accelerating)}\)

\([E1.2]\) \(\langle d_3,d_2\rangle:p, g_3\leq d_3\leq d_2\leq g_2\) \textbf{infers} \(g_3,g_2\rangle:p, \)
\(\text{Example} : <8,9> : \text{eat}(5,\text{hamburger}) \textbf{infers} <7,10> : \text{eat}(5, \text{hamburger})\)

**4.3. Concerning Completeness**

TRL is inherently disjunctive, as the assertion \([1,5]:p\) is equivalent to \(1:p \lor 2:p \lor \ldots \lor 5:p\). Given this fact, there is at least a case that the meta-interpreter does not answer correctly. If we happen to know \([1,5]:p\) and we provide the query \(T:p\), then the meta-interpreter should answer
However, the inference rules in the way they are implemented do not provide us with such facilities. To answer such questions we should provide the system with additional inference rules which give as a result disjunctions of extended atoms. Clearly, the system is not complete. But still it has been proved very expressive and efficient on various applications.

5. TRLi meta-interpreter and its applications

5.1. TRLi Meta-interpreter

The TRL Meta-interpreter follows the classic scheme.

\[
\begin{align*}
trl\_Solve(G:P) & \leftarrow G:P. \\
trl\_Solve(G:E) & \leftarrow E:P. \\
trl\_Solve(\text{Constraint}) & \leftarrow \{\text{Constraint}\}. \\
trl\_Solve(\text{Classic}) & \leftarrow \text{call(Classic)}. \\
trl\_solve\_body([]) & \leftarrow !. \\
trl\_solve\_body([QA,QB]) & \leftarrow ttrl\_solve(QA), ttrl\_solve\_body(QB).
\end{align*}
\]

It is easy to implement the general inference rules. However, as we want to take meaningful explanations about the results, we were forced to decompose the general inference rule into several cases. When these cases do not cover the general case, the general inference rule is triggered.

5.2. Inference Rules for Properties

(a) A G interval subset of D interval or a union of D intervals
\[
<\{G1,G2\},\{G3,G4\}> : P \leftarrow
trl\_Consistency\_Constraints(\{G1,G2\},\{G3,G4\}),
trl\_Case\_Constraints(\{G2 \leq G3\}), /* G is an interval */
trl\_Database(\{D1,D2\},\{D3,D4\} : P, Body),
trl\_Consistency\_Constraints(\{D1,D2\},\{D3,D4\}),
trl\_Case\_Constraints(\{D2 \leq D3\}), /* D is an interval */
trl\_left\_interval(P, D2, G2), /* at least intersection */
trl\_right\_interval(P, D3, G3). /* examine a possible missing right interval */
trl\_solve\_body(Body).
\]

(b) A G instance superset of D instance
\[
<\{G1,G2\},\{G3,G4\}> : P \leftarrow
trl\_Consistency\_Constraints(\{G1,G2\},\{G3,G4\}),
trl\_Case\_Constraints(\{G3 \leq G2\}), /* G is an instance */
trl\_Database(\{D1,D2\},\{D3,D4\} : P, Body),
trl\_Consistency\_Constraints(\{D1,D2\},\{D3,D4\}),
trl\_Case\_Constraints(\{D3 \leq D2\}), /* D is an instance */
\{ G3 \leq D3, D2 \leq G2 \}. /* \{G3, ..., G2\} superset of \{D3, ..., D2\} */
trl\_solve\_body(Body).
\]

(c) A G instance intersects with D interval
5.3. Inference Rules for Events

(a) A G interval equal to a D interval

\[((G1,G2),[G3,G4]) : E \leftarrow \]
Insert_Constistency_Constraints((G1,G2), [G3,G4]),
Insert_Case_Constraints((G1 \leq G2, G2 \leq G3, G3 \leq G4)), /* G is an interval */
Database((D1,D2), [D3,D4] : E, Body)
Insert_Constistency_Constraints((D1,D2), [D3,D4]),
Insert_Case_Constraints((D1 \leq D2, D2 \leq D3, D3 \leq D4)), /* D is an interval */
\{(D2<G2, G3=D3) \}
trl_solve_body(Body).

(b) A G instance superset of D instance

\[((G1,G2),[G3,G4]) : E \leftarrow \]
Insert_Constistency_Constraints((G1,G2), [G3,G4]),
Insert_Case_Constraints((G1=G3, G3 \leq G2-1, G2=G4)), /* G is an instance */
Database((D1,D2), [D3,D4] : E, Body)
Insert_Constistency_Constraints((D1,D2), [D3,D4]),
Insert_Case_Constraints((D1=D3, D3 \leq D2-1, D2=G4)), /* D is an instance */
\{(G3 \leq D3, D2 \leq G2) \}
/* (G3,...,G2 superset of D3,...,D2) */
trl_solve_body(Body).

(c) A G instance superset of D interval

\[((G1,G2),[G3,G4]) : P \leftarrow \]
Insert_Constistency_Constraints((G1,G2), [G3,G4]),
Insert_Case_Constraints((G1=G3, G3 \leq G2-1, G2=G4)), /* G is an instance */
Database((D1,D2), [D3,D4] : P, Body)
Insert_Constistency_Constraints((D1,D2), [D3,D4]),
Insert_Case_Constraints((D1=D2, D2 \leq D3, D3 \leq D4)), /* D is an interval */
\{(G3 \leq D2, D3 \leq G2) \}
/* (G3,...,G2 superset of D2,...,D3) */
trl_solve_body(Body).

5.4. Examples of TRL programming

Example 3. Simulation

A simple example which is often used is the simulation of a computer's cpu state. This example is also presented here in the context of TRLi.
- properties([cpu2,job_queue/1,initial_queue/1,job/1]).
- events([jobs/1]).
  l:initial_queue([([job1,1],[job2,2],[job3,2])].

257
\[ T: \text{cpu}(\text{idle}, 0) \leftarrow T: \text{job\_queue}([]). \]
\[ T: \text{cpu}(X, N) \leftarrow T: \text{job\_queue}([X,N]\text{\textbackslash Rest}). \]
\[ T: \text{job\_queue}(X) \leftarrow \]
\[ T1: \text{initial\_queue}(\text{Jobs}), \]
\[ \text{total\_time}(T1, \text{Jobs}, \text{Total}), \]
\[ \text{between}(T1, \text{Total}, T), \]
\[ \{T2 = T-T1\}, \]
\[ \text{pending\_jobs}(T2, \text{Jobs}, X). \]
\[ T: \text{job}(X) \leftarrow T: \text{job\_queue}([X,N]\text{\textbackslash Rest}). \]
\[ T: \text{jobs}([L]) \leftarrow T: \text{job}(L). \]
\[ <S,E>: \text{jobs}([L,Ls]) \leftarrow S: \text{job}(L), <S+1,E>: \text{jobs}(Ls). \]

The predicate \( T: \text{initial\_queue}(\text{Jobs}) \) defines the list of jobs (along with their duration) waiting for execution at the starting point \( T \). The predicate \( T: \text{cpu}(X,N) \) returns the job \( X \) executed in the cpu at time \( T \), and the remaining cpu time \( N \) for the job to be finished. The predicate \( T: \text{job\_queue}(\text{Jobs}) \) returns the remaining job queue (\( \text{Jobs} \)) at \( T \). The predicate \( T: \text{job}(X) \) returns in \( X \) the first job in the queue at \( T \). The predicate \( <B,E>: \text{jobs}(\text{Jobs}) \) returns in \( \text{Jobs} \) the list of jobs executed during the interval \( <B,E> \).

The classic predicates \( \text{total\_time}, \text{between}, \text{pending\_jobs} \) are defined as follows: The predicate \( \text{total\_time}(T1, \text{Jobs}, \text{Total}) \) computes the total time (\( \text{Total} \)) needed for all jobs to complete their execution. The predicate \( \text{between}(T1, \text{Total}, T) \) returns in \( T \) the time points between \( T1 \) and \( \text{Total} \). The predicate \( \text{pending\_jobs}(T, \text{Jobs}, X) \) returns in \( X \) the jobs which are still pending at \( T \), given that all the jobs are stored in the \( \text{Jobs} \) variable. Given the above program we may ask several questions:

**Query 1.** List the jobs \( X \) which are executed during the interval \( <1,3> \):

\[ \text{Query} : <1,3>: \text{jobs}(X). \]

\[ \text{Ans} : X=[\text{job1},\text{job2},\text{job2}] \]

**Query 2.** List the jobs \( X \) which are executed at some time between 1 and 3 (temporal instance \([1,3])\):

\[ \text{Query} : [1,3]: \text{jobs}(X). \]

\[ \text{Ans} : X=[\text{job1}], X=[\text{job2}], X=[\text{job2}], X=[\text{job1,job2}], X=[\text{job1,job2,job2}], X=[\text{job2,job2}]. \]

**Query 3.** List the jobs \( X \) which are executed during the interval \( <4,E> \):

\[ \text{Query} : <4,E>: \text{jobs}(X). \]

\[ \text{Ans} : E=4; X=[\text{job3}], E=5; X=[\text{job3,job3}]. \]

### 5.5. Applications of TRL

The TRL interpreter has been used as a kernel for building an expressive temporal backward planner, (TRL-Planner, [Mar95]), solving complex temporal planning applications such as the planning of cargo handling operations for chemical carriers, (Advisor, [Mar96]), as well as building a forward temporal planner for the production of interactive tutoring dialogues, (TRL-Tutor, [Pan98a]). All these planners have used TRL-like action representation schemata, i.e. actions which utilise the notion of temporal references. Preconditions and effects of actions are extended atoms which are also stored in a World temporal knowledge base. The role of TRL in these systems is to keep track of the changes in the World, asserting temporal effects and verifying the truth value of preconditions when a new actions is to be selected.

Other applications have also been supported especially from the area of temporal databases. In these cases TRL has been used as an intelligent temporal supervisor which manages and analyses
temporal queries in such a way so that it can extend the expressive power of classic database query languages [Bax98].

5.5. Expressive power of TRL

The P,F operators (possibility operator of Modal Logics) can both be expressed in TRL by a temporal instance. In fact, the possibility is better expressed through a temporal instance [T1,T2], as one can be more specific about the starting and ending temporal bounds than just referring to the possibility in the future (F operator) or the possibility in the past (P operator). Moreover, a temporal instance may start at sometime in the past and finish at sometime in the future. Similarly, the H,G operators (necessity operator of Modal Logics) can also be both expressed in TRL by a temporal interval <T1,T2>. Again, a temporal interval is more expressive than the H,G operators. Moreover, information concerning temporal points, can still be expressed. It is very important however, that all temporal references are special cases of the uncertain temporal interval. Therefore, a single notation captures the meaning of the operators P,F,G,H.

Assume the sentence q="John is happy". In Tense Logic [Gal87a], the sentences "John has been happy", "John will be happy", "John has always been happy", "John will always be happy" are expressed by Pq, Fq, Hq, Gq, respectively. In TRL such information can be represented much better: (first,now):q, (now,last):q, (first,now):<q, (now,last):<q, assuming that first, last and now have been defined to be the first moment, the last moment and the current moment of the conceptualisation. In addition TRL can express even more complex sentences.

Assume the sentence q="John eats an apple".in the Logic of Occurrence [Gal87b], the sentences "John has been eating an apple", "John is eating an apple", "John will be eating an apple", "John has eaten an apple", "John will eat an apple" can be represented by P Prog q, Prog p, F Prog p, Perf q, Pros p. In TRL such information may also be represented: <now-t,now-s>:q, where t>s, <now-t,now+s>:q, <now+t,now+s>:q, where t<s, (now-t):q, (now+t):q. However, such sentences are just special cases of even more complex sentences : "John has been eating an apple at some time between 8:00 a.m. and 10:00 a.m.: <[8,8+t],[10-s,10]>:q, where t,s are some positive temporal points, such that 8+t<10-s.

Allen's Logic [All84], can represent sentences involving the relations after, before, starts, overlaps, meets, etc. In TRL it is very easy to develop semantics for representing Allen's relations and functions. Moreover, we can extend and generalise these relations to the level of temporal instances and uncertain temporal intervals [Fra94]. The predicates HOLDS(p,t), IN(t,T), OCCUR(p,T) as well as all the properties which appear in Allen's Logic can also be represented and generalised within the TRL framework.

6. Conclusions and Future work

TRL is a logic based temporal language, which extends a first-order logic to a logic incorporating temporal references. It deals with the notion of temporal uncertainty which is deeply embedded in its semantics, represents both intervals and temporal points (certain and uncertain) with a single notation and generalises notions appearing in many other temporal logics. We have presented the specifications, syntax and semantics of TRL for both property and event atoms. We have also provided a meta-interpreter and discussed examples and applications of its use.

A TRL meta-interpreter has been developed using Sicstus Prolog. Quite recently we integrated the meta-interpreter in a full TRL development package for Windows 95/NT with which we will be able to experiment on other application areas.

Much work must be still be done. The TRL framework must be completed: a complete proof system is needed, semantics concerning negation must be investigated, a fully disjunctive version
could be developed, etc. Moreover, it seems that we can extend TRL to handle other kinds of data such as spatio-temporal references. Moreover, we intend to incorporate TRL as the kernel logic for virtual intelligent agent architectures [Pan98b, Pan99], complex simulation applications, planning/scheduling applications, etc.

References


