TEMPORAL REASONING WITH TRL

THEMIS PANAYIOTOPoulos

Knowledge Engineering Lab., Dep. of Informatics, University of Piraeus,
80 Karaoli & Dimitriou Str., 185 34 Piraeus, Greece
E-mail: themisp@uni.pi.gr

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.

1 Introduction

Temporal systems and programming languages, [12, 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 [1,5,6,9], which have found application in many domains, such as temporal planning [10,11,16], temporal databases [2], verification of concurrent systems, VLSI design, etc. The proposed approaches may be divided into numerical and symbolic approaches. The symbolic approaches may be further divided in classic logic, modal logic and reified logic approaches.

Quite recently, Disjunctive Chronolog [7], a disjunctive temporal logic programming language which extends the Chronolog language, [21], 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, [8,20], 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 [4].

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 specifics of time in 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 The specifics of time in TRL

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

The TRL Temporal Reference Language, [13], 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. Each TRL sentence was a classic first order logic sentence labelled with a temporal reference that was defining the temporal aspect of the sentence.

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 HTL [14]. Along with HTL, a method for transforming an HTL program to an equivalent constraint logic program was also developed. At the same time a TRL meta-interpreter was implemented and it was applied to some interesting examples from the areas of intelligent problem solving, simulation as well as temporal databases [15].

It was already clear that the developed system carried great resemblance to other constraint based approaches 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:

1. Time is discrete.
2. Time points are totally ordered.
3. An interval is a convex set of time points.
4. Time is unbounded in both directions (in the past and in the future).
5. 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). As we are also interested to represent
cases of incomplete temporal information, 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.

**Definition 2. (Temporal Interval).**
A Temporal interval is an expression of the form $<T_1,T_2>$, 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 $<[T_1,T_2],[T_3,T_4]>$, $<T_1,[T_2,T_3]>$, $<[T_1,T_2],T_3>$, where $T_1$, $T_2$, $T_3$, $T_4$ are temporal points.

**Definition 5. (Temporal Reference).**
A Temporal Reference $T_{ref}$ 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:
- $T_{ref}$ is a temporal point.
- $T_{ref}$ is of the form $<[T_1,T_2],[T_3,T_4]>$ and it holds $T_1 \leq T_2$, $T_1 \leq T_3$, $T_3 \leq T_4$, $T_2 \leq T_4$.
- $T_{ref}$ is of the form $<T_1,[T_3,T_4]>$ and it holds $T_1 \leq T_3$, $T_3 \leq T_4$.
- $T_{ref}$ is of the form $<[T_1,T_2],T_3>$ and it holds $T_1 \leq T_2$, $T_2 \leq T_3$.
- $T_{ref}$ is of the form $<T_1,T_2>$ and it holds $T_1 \leq T_2$
- $T_{ref}$ 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 Property 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 $<1918,1987>:_\text{lives}(<text>peter>)$ 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 $8:00 \text{ am} : _\text{leaving}(<text>bus>)$ 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:

$<T_1,T_2>:_\text{laughs}(<text>stella>) :- <T_1,T_2>:_\text{watches}(<text>stella,funny\_film>)$.

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 t1 and t2, at which the property is true.

‘Nikos arrives between 8:00 and 10:00’ is written as $[8:00 \text{ am},10:00 \text{ am}]: _\text{arrives}(<text>nikos>)$. 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 $[8:00 \text{ am},10:00 \text{ am}]: _\text{cooks}(<text>maria>)$. 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 t1 between s1 and s2 and a time point t2 between s3 and s4, 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 ⟨[9.00 am, 11.00 am], 5.00 pm⟩ : 

\text{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.

3.2 Event atoms

Events are relations which do not necessarily preserve their characteristics inside smaller temporal intervals. For example, in the statement ⟨1980, 2000⟩:<

\text{published}\_\text{papers}(\text{george},\text{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 ⟨T1, T2⟩, 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(joice, $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 [t1, t2], then there is at least one temporal interval ⟨s1, s2⟩, between t1 and t2, 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] : \text{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 ⟨[s1, s2], [s3, s4]⟩ then there exists a time point t1 between s1 and s2 and a time point t2 between s3 and s4, such that the relation is true during the temporal interval ⟨t1, t2⟩. 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 ⟨[9.00 am, 9.30 am], 10.00 am⟩ : 

\text{eat}(john, 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 $[\mathbb{D}^n \to \mathbb{D}]$; for each property predicate symbol, an element of $[\mathbb{N} \to 2^{\mathbb{D}^n}]$; for each event predicate symbol, an element of $[\mathbb{S} \to 2^{\mathbb{D}^n}]$, where S is the subset of $\mathbb{N} \times \mathbb{N}$ 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 $\models$ is defined in terms of temporal interpretations; $\models_{I, t} A$ denotes that a formula A is true at a moment t given some temporal interpretation I. On the other hand, $\models_{I, [t1, t2]} A$ denotes that a formula A is true at an interval $<t1, t2>$ 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 $\{t1, ..., t2\}$ to denote the set of consecutive integers starting from t1 and ending to t2, i.e. $\{t1, ..., t2\} = \{t1, t1+1, ..., t2-1, t2\}$.

**Definition 10. Basic Model Theoretic Semantics**
1. If $f(a1, a2, ..., an)$ is a term then $I(f(a1, a2, ..., an)) = I(f)(I(a1), I(a2), ..., I(an))$
   For any n-ary predicate symbol p with terms a1, a2, ..., an,
2. Classical predicates are not temporally interpreted:
   $\models_{I, t} c(a1, a2, ..., an)$ iff $\langle I(a1), I(a2), ..., I(an) \rangle \in I(c)$
3. Property predicates are interpreted over temporal points:
   $\models_{I, t} p(a1, a2, ..., an)$ iff $\langle I(a1), I(a2), ..., I(an) \rangle \in I(\{t\})$
4. Event predicates are interpreted over temporal intervals:
   $\models_{I, [t1, t2]} e(a1, a2, ..., an)$ iff $\langle I(a1), I(a2), ..., I(an) \rangle \in I(\{t1, ..., t2\})$
5. Temporal predicates are temporal constraints
6. $\models_{I, t} \neg A$ iff it is not the case that $\models_{I, t} A$
7. $\models_{I, t} A \wedge B$ iff $\models_{I, t} A$ and $\models_{I, t} B$
8. $\models_{I, t} A \vee B$ iff $\models_{I, t} A$ or $\models_{I, t} B$
9. $\models_{I, t} (\forall X) A$ iff $\models_{[d/X], t} A$ for all $d \in \mathbb{D}$ where the interpretation $I[d/X]$ is identical to I except that the variable X is assigned the value d.
10. $\models_{I, t} (\exists X) A$ iff $\models_{[d/X], t} A$ for some $d \in \mathbb{D}$ 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 ∈ N then (|= 1,t, S:p) iff (|= 1,S p)
12. |= 1,t (\forall S) S:p iff for all S ∈ N, |= 1,S p
13. |= 1,t (\exists S) S:p iff for some S ∈ N, |= 1,S p
14. |= 1,t <S1,S2>:p iff for all S ∈ {S1,...,S2}, |= 1,S p
15. |= 1,t [S1,S2]:p iff for some S ∈ {S1,...,S2}, |= 1,S p
16. |= 1,t <[S1,S2],[S3,S4]>:p iff for some T1 ∈ {S1,...,S2}, for some T2 ∈ {S3,...,S4}, |= 1,t <T1,T2>: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 : if a |= b then a \lor b |= b, 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).

Definition 12. Model theoretic semantics for events
If e is an event, and S=<S1,S2> ∈ NxN, S1≤S2, then
17. |= 1,t S:e iff |= 1,S e, in the case of <t,t> for simplicity we write t.
18. |= 1,t (\forall S) S:e iff for all S |= 1,S e
19. |= 1,t (\exists S) S:e iff for some S |= 1,S e
20. |= 1,t [S1,S2]:e iff for some {S3,...,S4}⊆{S1,...,S2}, |= 1,S3:S4 e
21. |= 1,t <[S1,S2],[S3,S4]>:e iff for some T1 ∈ {S1,...,S2}, for some T2 ∈ {S3,...,S4}, |= 1,t <T1,T2>: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) cannot be further decomposed.
(b) [3,5]:e(a) |= <3,3>:p(a) \lor <3,4>:p(a) \lor <3,5>:p(a) \lor <4,4>:p(a) \lor <4,5>:p(a) \lor <5,5>:p(a)
(c) <[3,5],[8,9]>:e(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)
\[ <[3,5],[4,5]>: p(a) \vdash <3,4>: p(a) \lor <3,5>: p(a) \lor <4,4>: p(a) \lor <4,5>: p(a) \lor <5,5>: p(a) \]

### 3.4 Tautologies and Implications concerning property atoms

We are going to use the letters Di for temporal elements which are part of the source of information (Data) and Gi for temporal elements which are part of the derived information (Goal). Proofs of these are included in [19].

1. If \( S_1 \leq S_2 \leq S_3 \leq S_4 \) then \( <[S_1,S_2],[S_3,S_4]>: A \vdash <S_2,S_3>: A \)
2. If \( S_1 \leq S_3 \leq S_2 \leq S_4 \) then \( <[S_1,S_2],[S_3,S_4]>: A \vdash [S_2,S_3]: A \)
3. If \( \{D_1,\ldots,D_2\} \subseteq \{G_1,\ldots,G_2\} \) and \( \{D_3,\ldots,D_4\} \subseteq \{G_3,\ldots,G_4\} \), 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 \( <[D_1,D_2],[D_3,D_4]>: A \vdash <[G_1,G_2],[G_3,G_4]>: A \)
4. If \( \{D_1,\ldots,D_2\} \supseteq \{G_1,\ldots,G_2\} \) i.e. \( D_1 \leq G_1 \land G_2 \leq D_2 \) then \( <D_1,D_2>: A \vdash <G_1,G_2>: A \)
5. If \( \{D_1,\ldots,D_2\} \cup \{D_3,\ldots,D_4\} \supseteq \{G_1,\ldots,G_2\} \) i.e. \( D_3 \leq D_2 \land D_1 \leq D_4 \land \min(D_1,D_3) \leq G_1 \land G_2 \leq \max(D_2,D_4) \) then \( <D_1,D_2>: A \land <D_3,D_4>: A \vdash <G_1,G_2>: A \)
6. If \( \{R_1,\ldots,R_2\} \cap \{G_1,\ldots,G_2\} \neq \emptyset \), i.e. \( R_1 \leq G_2 \land G_1 \leq R_2 \) then \( <R_1,R_2>: A \vdash [G_1,G_2]: A \)
7. If \( \{D_1,\ldots,D_2\} \subseteq \{G_1,\ldots,G_2\} \), i.e. \( G_1 \leq D_1 \land D_2 \leq G_2 \) then \( [D_1,D_2]: A \vdash [G_1,G_2]: A \)

### 3.5 Tautologies and Implications concerning event atoms

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 \( S_1 = S_2 = S_3 = S_4 \) then \( <[S_1,S_2],[S_3,S_4]>: A \vdash <S_2,S_3>: A \)
9. If \( S_1 = S_3 \leq S_2 = S_4 \) then \( <[S_1,S_2],[S_3,S_4]>: A \vdash [S_2,S_3]: A \)
10. If \( \{D_1,\ldots,D_2\} \subseteq \{G_1,\ldots,G_2\} \) and \( \{D_3,\ldots,D_4\} \subseteq \{G_3,\ldots,G_4\} \), then \( <[D_1,D_2],[D_3,D_4]>: A \vdash <[G_1,G_2],[G_3,G_4]>: A \)
11. If \( \{D_1,\ldots,D_2\} = \{G_1,\ldots,G_2\} \), i.e. \( D_1 = G_1 \land G_2 = D_2 \) then \( <D_1,D_2>: A \vdash <G_1,G_2>: A \)
12. If \( \{D_1,\ldots,D_2\} \subseteq \{G_1,\ldots,G_2\} \), i.e. \( G_1 \leq D_1 \land D_2 \leq G_2 \) then \( <D_1,D_2>: A \vdash [G_1,G_2]: A \)
13. If \( \{D_1,\ldots,D_2\} \subseteq \{G_1,\ldots,G_2\} \), i.e. \( G_1 \leq D_1 \land D_2 \leq G_2 \) then \( [D_1,D_2]: A \vdash [G_1,G_2]: A \)
3.6 First Order Translation

A first order translation is very helpful as it provides an alternative first order interpretation to a sentence’s 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>
<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;S1,S2&gt;:p</td>
<td>(∀T) (S1 ≤ T ≤ S2 ⇒ T:p)</td>
</tr>
<tr>
<td>Temporal instance</td>
<td>[S1,S2]:p</td>
<td>(∃T) (S1 ≤ T ≤ S2 ∧ T:p)</td>
</tr>
<tr>
<td>Uncertain Temporal</td>
<td>{[S1,S2],[S3,S4]}:p</td>
<td>(∃T1)(∃T2) (S1 ≤ T1 ≤ S2 ∧ S3 ≤ T2 ≤ S4 ∧ T1,T2:p)</td>
</tr>
</tbody>
</table>

<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>(∃T1)(∃T2) (S1 ≤ T1 ≤ T2 ≤ S2 ∧ T1,T2:e)</td>
</tr>
<tr>
<td>Uncertain Temporal</td>
<td>{[S1,S2],[S3,S4]}:e</td>
<td>(∃T1)(∃T2) (S1 ≤ T1 ≤ S2 ∧ S3 ≤ T2 ≤ S4 ∧ T1,T2:e)</td>
</tr>
</tbody>
</table>

Using these two translations we can easily prove, [19], the following theorems:

**Properties**
- If S1=S2=S3=S4=S, then {[S,S],[S,S]}:p ↔ S:p
- If S1=S2=S and S3=S4=E, then {[S,S],[E,E]}:p ↔ <S,E>:p
- If S1=S3=S and S2=S4=E, then {[S,E],[S,E]}:p ↔ [S,E]:p

**Events**
- If S1=S2=S3=S4=S, then {[S,S],[S,S]}:e ↔ S:e
- If S1=S2=S and S3=S4=E, then {[S,S],[E,E]}:p ↔ <S,E>:p
- If S1=S3=S and S2=S4=E, then {[S,E],[S,E]}:e ↔ [S,E]:e

4 Deduction and Inference rules

For classical atoms, we retain SLD-resolution, i.e. TRLi 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.

\[ P1 \] \{ \langle d1, d2 \rangle, \langle d3, d4 \rangle \vdash p, d2 \leq g2, g3 \leq d3 \} \quad \text{inference} \quad \langle g1, g2 \rangle, \langle g3, g4 \rangle \vdash p \\
\[ P2 \] \{ \langle a1, a2 \rangle, \langle a3, a4 \rangle \vdash p, \langle b1, b2 \rangle, \langle b3, b4 \rangle \vdash p, a2 \leq a3, b2 \leq b3, a2 \leq b3 + 1 \} \\
\quad \text{inference} \quad \langle c1, c2 \rangle, \langle c3, c4 \rangle \vdash p \quad \text{where} \quad c2 = \min(a2, b2), \ c3 = \max(a3, b3) \\

Examples of special cases

\[ P1.1 \] \{ \langle d2, d3 \rangle \vdash p, d2 \leq g2, g3 \leq d3 \} \quad \text{inference} \quad \langle g2, g3 \rangle \vdash p \\
\text{Example:} \langle 8, 14 \rangle \vdash \text{works(manolis)} \quad \text{inference} \quad \langle 10, 12 \rangle \vdash \text{works(manolis)} \\
\[ P1.2 \] \{ \langle d3, d2 \rangle \vdash p, g3 \leq d3 \leq d2 \leq g2 \} \quad \text{inference} \quad \langle g3, g2 \rangle \vdash p \\
\text{Example:} \langle 8, 9 \rangle \vdash \text{eat_breakfast} \quad \text{inference} \quad \langle 7, 10 \rangle \vdash \text{eat_breakfast} \\
\[ P2.1 \] \{ \langle a2, a3 \rangle \vdash p, \langle b2, b3 \rangle \vdash p, a2 \leq a3 + 1, \ b2 \leq b3 + 1 \} \quad \text{inference} \quad \langle c2, c3 \rangle \vdash p, \quad \text{where} \\
\quad c2 = \min(a2, b2), \ c3 = \max(a3, b3). \\
\text{Example: From} \langle 1, 5 \rangle \vdash \text{studies}, \langle 3, 8 \rangle \vdash \text{studies}, \text{infer} \langle 1, 8 \rangle \vdash \text{studies} \\

4.2 Inference rules of events.

\[ E1 \] \{ \langle d1, d2 \rangle, \langle d3, d4 \rangle \vdash p, \ g1 \leq d1, \ d2 \leq g2, \ g3 \leq d3, \ d4 \leq g4 \} \quad \text{inference} \quad \langle g1, g2 \rangle, \langle g3, g4 \rangle \vdash p \\

Examples of special cases

\[ E1.1 \] \{ \langle d2, d3 \rangle \vdash p, \ d2 = g2 \leq g3 = d3 \} \quad \text{inference} \quad \langle g2, g3 \rangle \vdash p, \\
\text{(event intervals do not retain the truth values inside them)} \\
\text{Example:} \langle 8, 9 \rangle \vdash \text{speed_up(car)} \quad \text{inference} \quad \langle 8, 9 \rangle \vdash \text{speed_up(car)} \\
\text{(From the 8th second to the 9th second the car was accelerating)} \\
\[ E1.2 \] \{ \langle d3, d2 \rangle \vdash p, \ g3 \leq d3 \leq d2 \leq g2 \} \quad \text{inference} \quad \langle g3, g2 \rangle \vdash p, \\
\text{Example:} \langle 8, 9 \rangle \vdash \text{eat(5, hamburger)} \quad \text{inference} \quad \langle 7, 10 \rangle \vdash \text{eat(5, hamburger)} \\

5 TRLi meta-interpreter and its applications

5.1 Syntax of TRL programs

Clearly, the TRL temporal logic programming language is a ‘Horn subset’ of the TRL Logic. 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 A0 ← A1,...,An where Ai, (i≥0) are TRL atoms. A TRL program is a set of TRL clauses. A TRL query is a formula of the form ← Q1,...,Qn where Qi, (i≥1) are TRL atoms.

### 5.2 TRLi Meta-interpreter

The TRL Meta-interpreter follows the classic scheme.

\[

trl\_Solve(G:P) \leftarrow G:P.

trl\_Solve(G:E) \leftarrow G:E.

trl\_Solve(Constraint) \leftarrow \{Constraint\}.

trl\_Solve(Classic) \leftarrow call(Classic).

trl\_solve\_body([I]) \leftarrow !.

trl\_solve\_body([QA|QB]) \leftarrow trl\_solve(QA), trl\_solve\_body(QB).
\]

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.

### 5.3 Inference Rules for Properties

(a) A G interval subset of D interval or a union of D intervals

\[
<\{G1,G2\},\{G3,G4\}> : P \leftarrow
\]

- Insert\_Consistency\_Constraints(<\{G1,G2\},\{G3,G4\}>),
- Insert\_Case\_Constraints(\{G2≤G3\}), /* G is an interval */
- Database(<\{D1,D2\},\{D3,D4\}> : P, Body)
- Insert\_Consistency\_Constraints(<\{D1,D2\},\{D3,D4\}>),
- Insert\_Case\_Constraints(\{D2≤D3\}), /* D is an interval */
- \{D2≤G3, G2≤D3\}, /* at least Intersection */
- leftinterval(P, D2, G2), /* examine a possible missing left interval */
- rightinterval(P, D3, G3), /* examine a possible missing right interval */
- trl\_solve\_body(Body).

(b) A G instance superset of D instance

\[
<\{G1,G2\},\{G3,G4\}> : P \leftarrow
\]

- Insert\_Consistency\_Constraints(<\{G1,G2\},\{G3,G4\}>),
- Insert\_Case\_Constraints(\{G3≤G2-1\}), /* G is an instance */
- Database(<\{D1,D2\},\{D3,D4\}> : P, Body)
- Insert\_Consistency\_Constraints(<\{D1,D2\},\{D3,D4\}>),
Insert_Case_Constraints({D3≤D2-1}), /* D is an instance */
{ G3≤D3, D2≤G2 }, /* {G3,...,G2} superset of {D3,...,D2} */
trl_solve_body(Body).

(c) A G instance intersects with D interval
Insert_Consistency_Constraints(<[G1,G2],[G3,G4]>),
Insert_Case_Constraints({G3≤G2-1}), /* G is an instance */
Database(<[D1,D2],[D3,D4]>: P, Body)
Insert_Consistency_Constraints(<[D1,D2],[D3,D4]>),
Insert_Case_Constraints({D2≤D3}), /* D is an instance */
{ G3≤D3, D2≤G2 }, /* {G3,...,G2} intersects with {D2,...,D3} */
trl_solve_body(Body).

5.4 Inference Rules for Events

(a) A G interval equal to a D interval
Insert_Consistency_Constraints(<[G1,G2],[G3,G4]>),
Insert_Case_Constraints({G1=G2, G2≤G3, G3=G4}), /* G is an interval */
Database(<[D1,D2],[D3,D4]>: E, Body)
Insert_Consistency_Constraints(<[D1,D2],[D3,D4]>),
Insert_Case_Constraints({D1=D2, D2≤D3, D3=D4}), /* D is an interval */
{D2=G2, G3=D3}, /* equal intervals */
trl_solve_body(Body).

(b) A G instance superset of D instance
Insert_Consistency_Constraints(<[G1,G2],[G3,G4]>),
Insert_Case_Constraints({G1=G3, G3≤G2-1, G2=G4}), /* G instance */
Database(<[D1,D2],[D3,D4]>: E, Body)
Insert_Consistency_Constraints(<[D1,D2],[D3,D4]>),
Insert_Case_Constraints({D1=D3, D3≤D2-1, D2=D4}), /* D instance */
{ G3≤D3, D2≤G2 }, /* {G3,...,G2} superset of {D3,...,D2} */
trl_solve_body(Body).

(c) A G instance superset of D interval
Insert_Consistency_Constraints(<[G1,G2],[G3,G4]>),
Insert_Case_Constraints({G1=G3, G3≤G2-1, G2=G4}), /* G instance */
Database(<[D1,D2],[D3,D4]>: P, Body)
Insert_Consistency_Constraints(<[D1,D2],[D3,D4]>),
Insert_Case_Constraints({D1=D2, D2≤D3, D3=D4}), /* D interval */
\[ \{ G3 \leq D2, D3 \leq G2 \}./* \{ G3, ..., G2 \} superset of \{ D2, ..., D3 \} */ \]

5.5 Applications of TRL

The TRL interpreter has been used as a kernel for building an expressive temporal backward planner, (TRL-Planner, [10]), solving complex temporal planning applications such as the planning of cargo handling operations for chemical carriers, (Advisor, [11]), as well as building a forward temporal planner for the production of interactive tutoring dialogues, (TRL-Tutor, [16]). 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 action 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 [2].

5.6 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 [8], 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 [9], 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 [1], 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 [6]. The predicates HOLDS(p,T), IN(t,T), OCCURING(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 [17,18], complex simulation applications, planning/scheduling applications, etc.
References


