EMTPL and its relation to first order logic

María Laura Cobo **Marcelo A. Falappa**

Department of Computer Science and Engineering – Universidad Nacional del Sur
Artificial Intelligence Research and Development Laboratory (L.I.D.I.A)
Institute of Computer Science and Engineering
Av. Alem 1253 – B8000CPB Bahía Blanca, ARGENTINA
Email: [mlc,mfalappa]@cs.uns.edu.ar

Key Words: Logic Programming, Metric Temporal Logic, Temporal Databases

Abstract

Time and change are notions that seems unavoidable in some areas of work and investigation, languages that can deal with these notions are necessary. At the same time, methods for a proper time handling are quite complex, mainly because problem's complexity and variety of solutions. Between the languages developed to cover these expectations, under a specific view of time, are [Cobo and Augusto, 1999a] *EMTLP* and a metric temporal logic's fragment, *bounded universal Horn formulae* analyzed by Brzoska [Brzoska, 1998]. Although both of them performed metric temporal programming, they face this fact from different perspectives. In this work we are going to try a comparison between them after a short overview over each. In this first stage we present a way of representing EMTPL's in first order logic using Brzoska's approximation as a bridge, and we also compare some aspects of both programming languages.

1 Introduction

The use of temporal operators has show to be a very simple and natural way of expressing temporal restrictions on a program. That is why philosopher's investigation in this area is very useful for the development of programming languages for specification [Prior, 1967a, Burgess, 1984]. There are some alternatives in the academic environment in this sense. The main problem behind these languages is that they are developed on isolation, i.e., without explicit consideration to previous and/or similar alternatives. In the particular case of metric programming languages there are not so many options as in no-metric case. The difficulty here is that resolution procedures are very different and usually there is no use of already developed tools. This an important topic because although metric temporal logic generalize temporal linear logic [Pnueli, 1977] the cost involved in its computation are highly expensive or impossible to compute sometimes. In this sense Brzoska's line of investigation [Brzoska, 1998] deals with these problems. As a matter of fact he defined a language that is a Horn fragment and defines a way of translating any program of this language to constraint logic programming (CLP) and so he established a relationship between the language and first order logic. This relationship allows the use of all tested

tools developed for this last formalism. This simple fact is very important if we take in consideration how hard is the development of correct proof methods, specially for this kind of language, and complexity issues.

EMTPL [Cobo and Augusto, 1999a] is a metric temporal language that fixed a problem that is common in temporal databases. Usually temporal languages, i.e., languages that add time through operators manage "imprecise" information. This causes a problem when times evolve, this is so because truth value of any formula depends on "present" moment. *EMTPL* solves this problem by using an operator that sets explicitly which is the "present" moment for each formula. This language has it's own proof algorithm but as can be seen in [Cobo and Augusto, 1999a] the process is highly complicated. If a relation to first order logic can be made we may count with a valuable alternative to check programs.

We present a short overview of *EMTPL* language in section 2 and of Brzoska's system in section 3. We give a short motivation, a brief idea of underlying logic and language definition of each of them. In section 4 we show the aspects where the proposals differ from each other. Finally, we give an idea of how to perform a translation from *EMTPL* to first order logic, showing how this is done for *Bounded universal Horn formulae*.

2 EMTPL

In [Cobo and Augusto, 2000] we present a programming language, *Metric Temporal Programming Language* (MTPL), with the capability of dealing with more precise information than languages without metric temporal operators like [Cobo and Augusto, 1999b] *Temporal Prolog*. This language is based in a deeply studied logic proposed by Prior ([Prior, 1967a] and [Prior, 1967b]) and follows one of the most known ways of seeing time, through A-series (past, present and future). There are other logic, and some of them formalize the other view of time, as B-series (before, after), where explicit references of time are used. An example of this type of logic is presented by Rescher [Rescher and Urquhart, 1971]. Although the apparent disagreement between these conceptions of time, there are circumstances where is very useful to count with both of them simultaneously. One of the areas where this need is quite clear is the databases area.

The problem of considering a logic like Prior's one, is reduced to the cost of maintenance, this is because the present changes continuously, and we need to move all the present facts to the past at the same speed that time flows. This has a very high cost, because we will spend more time updating the database than solving any other problem [Kowalski, 1992]. On the other hand, logic as Rescher's one, do not capture "imprecise" information, where imprecise means that the exact moment of occurrence of some event is not know with certainty.

As a way to avoid this natural problem of updating we are going to try to take the advantages of each of the ways of conceiving time. We mean with this we kept flexibility of Prior's operators and preciseness of Rescher's ones. In this way we avoid the continuous updating of the database.

A way to obtain our purpose is through the introduction of At operator to MTPL language; more details of the programming language and it's main aspects can be found in [Cobo and Augusto, 2000].

2.1 The Underlying Logic

We will describe in this section the logical foundations of *EMTPL* programming language. More details of some aspects related with the syntax and semantics of the temporal operators given below, can be found in [Cobo and Augusto, 1999a]

The syntax of the operators on *EMTPL* is as follows:

$$\phi = p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathsf{At}(n,\ \phi) \mid \Uparrow_n\phi \mid \exists n\ \Uparrow_n\phi \mid \forall n\ \Uparrow_n\phi \mid \Downarrow_n\phi \mid \exists n\ \Downarrow_n\phi \mid \forall n\ \Downarrow_n\phi$$

Disjunction and implication could be defined as usual. Here we assume a set of instants $\ldots, i_n, i_{n+1}, i_{n+2}, \ldots$ that represents an unbounded and linear temporal structure. The information in the system could be seen as a list of sets, σ, that contains all the truths we have in our theory. Each member of the list is a set denoted by $\sigma(j)$. Then $\sigma(j)$ represents the set of truths at an instant j. We follow with an inductive definition for "a temporal proposition true at an instant j". We start specifying that $p \in \sigma(j)$ means "p is true at instant j".

It is important to point out that the quantification is applied only over the members of the temporal structure, i.e., the set of numbers we choose to represent time. We ask this set of numbers to be a metric space.

$(\sigma, j) \models p$ iff $p \in \sigma(j)$
$(\sigma, j) \models \neg p$ iff $(\sigma, j) \not\models p$
$(\sigma, j) \models p \wedge q$ iff $(\sigma, j) \models p$ and $(\sigma, j) \models q$
$(\sigma, j) \models \Uparrow_c p$ iff $(\sigma, j + c) \models p$
$(\sigma, j) \models \Downarrow_c p$ iff $(\sigma, j - c) \models p$
$(\sigma, j) \models \exists n\ \Uparrow_n p$ iff exists an instant $n,\ n > 0$ such that $(\sigma, j + n) \models p$
$(\sigma, j) \models \exists n\ \Downarrow_n p$ iff exists an instant $n,\ n > 0$ such that $(\sigma, j - n) \models p$
$(\sigma, j) \models \forall n\ \Uparrow_n p$ iff for all instant $n,\ n > 0$ we have $(\sigma, j + n) \models p$
$(\sigma, j) \models \forall n\ \Downarrow_n p$ iff for all instant $n,\ n > 0$ we have $(\sigma, j - n) \models p$
$(\sigma, j) \models \mathsf{At}(c,\ p)\ p$ iff $(\sigma, c) \models p$

Propositions without any of these temporal operators are considered 'true in the present'. It should be observed that we made the strong assumption that instants required in the definitions of the operators always exist. According to the technical language, we are using the so-called *strong* operators [Barringer et al., 1996]. In case we consider a bounded temporal structure, the appropriate limits must be put in the definitions in order to assure a right behavior in border conditions. For example, in the case of asking for $\Downarrow_n A$ at the first instant, we must consider the *weak* versions of the operators.

It is important to point out that interval based operators can be defined in terms of the operators we already have, see [Cobo and Augusto, 1999a] for more details.

2.2 EMTPL: The language

The language is based on an extension of Prior's Metric Temporal Logic [Prior, 1967a, Prior, 1967b]. This extension consists mainly, on introducing At operator in MTPL's axiomatization [Cobo and Augusto, 2000]. According with Prior's opinion, the extension may be seen as the need of preceding every theorem of MTPL's logic by At operator. The

extension performed also adds some new axioms to allow the proper handling of information affected by this operator. The axiom set can be found in [Cobo and Augusto, 1999a]

The language based on this extended logic, *Extended Metric Temporal Programming Language* (EMTPL), is define as follows:

DEFINITION **1** (EMTPL)
Let us consider a language with propositional atoms, the logic connectives: $\wedge$, $\vee$, $\rightarrow$, $\neg$ and the temporal operators: $\Diamond\!\!\!\!\!\!+_n A$, $\Diamond_n A$, $\Diamond\!\!\!\!\!\!+ A$, $\Diamond A$, $\boxplus A$, $\boxminus A$ and At. We define the notion of *EMTPL*'s *program, clause, always clause, ordinary clause, head, body* and *goal* as follows:

A *program* is a set of "clauses".

A *clause* is $\mathsf{At}(i,\ C)$ or C where C is an "ordinary clause".

An *ordinary clause* is a "head" or a $B \rightarrow H$, where B is a "body" and H is a "head".

A *head* is an atomic formula, $\Diamond\!\!\!\!\!\!+_n A$ or $\Diamond_n A$ or $\Diamond\!\!\!\!\!\!+_A$ or $\Diamond_A$ or $\boxplus A$ or $\boxminus A$ where A is a conjunction of "ordinary clauses".

A *body* is an atomic formula, a conjunction of bodies, $\Diamond\!\!\!\!\!\!+_n B$ or $\Diamond_n B$ or $\boxplus A$ or $\boxminus A$ or $\neg B$ where B is a body.

A *goal* is $\mathsf{At}(n,\ B)$ or B where B is any body or a disjunction of bodies. ■

It is important to point out that the facts or rules ($Prop$) that are not affected with an At operator are interpreted as facts or rules that happen in the present moment, i.e., can be seen as $\mathsf{At}(m,\ Prop)$, where m means "now".

3 Brzoska: Programming in metric temporal logic

In his research, Cristoph Brzoska [Brzoska, 1998, Brzoska, 1993] established a relationship between a metric temporal language based on a particular logic [Koymans, 1990] and constraint logic programming. He developed a way to achieve translation between these two ways of manage information, and also resolution methods for this process. Although his analysis was made over discrete and dense time, in this work we will restrict ourselves to discrete case time only.

3.1 Brzoska's underlying logic: *MTL*

Bounded Universal Horn formulae are based on a *temporal metric logic*(MTL). This logic was developed for real time specifications, more details can be found in [Koymans, 1990]. Logic's well formed formulae (wff) are built up with the usual logic connectives (conjunction, disjunction, implication) an the following temporal operators:

$\Box_I \phi$: always within the interval I, ϕ
$\Diamond_I \phi$: sometime within the interval I, ϕ
$\phi\ \mathcal{S}\ \psi$: ϕ has always been true, since ψ was true
$\phi\ \mathcal{U}\ \psi$: ϕ will be always true, until ψ will be true

I is a time interval, i.e., $I = [c^-, c^+]$ with c^-, $c^+ \in \mathbb{Z} \cup \{-\infty, \infty\}$ in the discrete case. The operators $\mathcal{S}$ and $\mathcal{U}$ were introduced by Kamp [Kamp, 1968], they are expressive complete for time structures which are isomorphic to integer or reals. On their metric versions, $\phi\ \mathcal{S}_I\ \psi$ and $\phi\ \mathcal{U}_I\ \psi$, specify an interval, within which the second argument of the binary operator has to be true.

Let Σ be a signature that represents a triple (T, F, P), where T is a set of *sorts*, F is a set of functions symbols with temporal free interpretation, and P is a set of predicate symbols with denotations varying with time. In modal logic's terminology predicate symbols are *flexible* while function symbols are *rigid*.

DEFINITION **2** *Validity* of a formula A in a (MTL-Σ) structure $\mathcal{M}$ at time t under some variable assignment α, denoted by $(\mathcal{M}, \alpha) \vDash_t A$ is defined by:

1. $(\mathcal{M}, \alpha) \vDash_t p(r_1, \ldots r_n)$ if and only if predicate p holds on $\mathcal{M}$ under the assignment α over time t.

2. $(\mathcal{M}, \alpha) \vDash_t \Box_I A$ if and only if **for all** $t' \in I$, $(\mathcal{M}, \alpha) \vDash_{t+t'} A$

3. $(\mathcal{M}, \alpha) \vDash_t \Diamond_I A$ if and only if **for some** $t' \in I$, $(\mathcal{M}, \alpha) \vDash_{t+t'} A$

4. $(\mathcal{M}, \alpha) \vDash_t A\, \mathcal{S}_I\, B$ if and only if **there is a** $t' < 0$ and $t' \in I$, such that $(\mathcal{M}, \alpha) \vDash_{t+t'} B$ and **for all** t'' with $t + t' < t'' < t$, $(\mathcal{M}, \alpha) \vDash_{t''} A$

5. $(\mathcal{M}, \alpha) \vDash_t A\, \mathcal{U}_I\, B$ if and only if **there is a** $t' > 0$ and $t' \in I$, such that $(\mathcal{M}, \alpha) \vDash_{t+t'} B$ and **for all** t'' con $t < t'' < t + t'$, $(\mathcal{M}, \alpha) \vDash_{t''} A$

The remaining cases, i.e., non-temporal logic connectives, have the usual semantics.

Satisfiability and *logical inference* are defined in the standard way. Translation of classical temporal operators to metric operators is presented in [Brzoska, 1998].

We can notice that reflexivity of temporal operators in this context turns out be irrelevant. Justification is clear by the use of unbounded or bounded intervals, leading to a reflexive or irreflexive definition of operator according to user's particular needs.

3.2 Bounded universal (modality) goals

The language is conformed by temporal Horn formulae containing $\Diamond_I$, $\mathcal{S}_I$, $\mathcal{U}_I$ and $\Box_I$ operators in goals and bodies, where I may be a bounded or unbounded interval over the integers. Formally, the goals are called *bounded universal (modality) goals* and are defined by:

$$G ::= \epsilon \ \mid\ A \mid \Diamond_I G \mid G \wedge G \mid \Box_{I'} G \mid G\ \mathcal{S}_I\ G \mid G\ \mathcal{U}_I\ G$$

The Horn formulae called *bounded universal (modality) Horn formulae* by:

$$D ::= A \mid \Box_{I'} G \mid D \leftarrow G$$

where I and I' denotes interval with bound in $\mathbb{Z}$, ϵ represents the empty goal and A ranges over atoms.

This class of formula is sufficiently expressive for several applications from areas such as (deductive) databases to real-time systems.

4 Comparing languages

The main interest of this article is focused in a possible usage of *constraint logic programming tools* for *EMTPL* as an alternative to the developed algorithm (more details in [Cobo and Augusto, 1999a]). This may be achieved if a translation from *EMTPL* to *Bonded universal (modality) goals* or *MTL* could be performed.

Expressivity differences

As it was already stated *EMTPL* is and extension of *MTPL* [Cobo and Augusto, 2000], which make use of relative temporal references. As a consequence *MTPL*'s sentences are always *temporally indefinite*, i.e., truth value of any language formula must consider evaluation moment t. On the contrary *EMTPL*'s sentences are *temporally definite*, this means that truth value of a formula is independent from the flow of time, and then generally do not change unless information changes. In this sense Brzoska's approximation deals with *temporally indefinite* information, this is clear in definition 2 where dependency between information and evaluation time is stated.

Another difference that can be appreciated at simple glance, is related to the *metric* involved in languages. While *EMTPL*'s metric is $\mathbb{N} \cup \{0\}$ Koymans's logic [Koymans, 1990], *MTL*, metric uses $\mathbb{Z} \cup \{-\infty, \infty\}$. Despite this difference a direct rewriting of *EMTPL*'s operators can be made, excluding $\mathsf{At}(n,\ \alpha)$ from the analysis :

$$\begin{array}{rcl} \Diamond\!\!\!\!-_c\ phi & \leftrightarrow & \Diamond_{[c,c]}\phi \\ \exists n\ \Diamond\!\!\!\!-_n\ phi & \leftrightarrow & \Diamond_{(0,\infty)}\phi \\ \forall n\ \Diamond\!\!\!\!-_n\ phi & \leftrightarrow & \Box_{(0,\infty)}\phi \end{array} \qquad \begin{array}{rcl} \Diamond\!\!\!\!-_c\ phi & \leftrightarrow & \Diamond_{[-c,-c]}\phi \\ \exists n\ \Diamond\!\!\!\!-_n\ phi & \leftrightarrow & \Diamond_{(-\infty,0)}\phi \\ \forall n\ \Diamond\!\!\!\!-_n\ phi & \leftrightarrow & \Box_{(-\infty,0)}\phi \end{array}$$

In the last four operators the intervals used are unbounded on 0, because *EMTPL*'s operators are reflexive. For irreflexive ones the only visible difference is in the use of bounded intervals instead.

Although there is a clear translation from almost all *EMTPL*'s operators, equivalence can not be established because there is no way of setting an equivalence version in *MTL* for At operator. If we try to make the opposite analysis, rewriting is not guaranteed although a subset can be translated. This difference in expressivity is based on the use on Kamp's operators [Kamp, 1968] in Brzoska's proposal. This set of operators are expressive complete for integer and real flow of time. This simple fact made the *MTL*'s operator set more expressive than the corresponding set for *EMTPL*.

Languages definitions

Both languages are based on a Horn fragment. This is so, because both alternatives look for an implementation an the complexity of the underlying logic lead languages to *NP-Complete problems*. Because of this the only difference between the languages are determined by the expressivity differences in the underlying logic of them. And the fact that negation was not considered in *bounded universal Horn formulae or goals* originally (see section 3.2) while it appears in *EMTPL*'s definition (see section 2.2). Even though this fallacy in Brzoska's research is covered by a proper extension to manage negation [Brzoska, 1998].

5 *EMTPL*'s formulae to first-order logic

Following the approach of functional translation developed by Brzoska [Brzoska, 1998] for *bounded universal Horn formulae* and pioneered by [Ohlbach, 1991] and others for modal logic. Brzoska translate formulae of the temporal logic into first order logic with fixed interpretation of some symbols modelling the flow of time. The idea is to add an additional argument to each predicate and to express the temporal relations that are expressed by temporal operators in temporal logic by formulae of classical logic. More precisely, temporal Σ-formulae (see section 3) are translated into a formula over an enriched signature $\pi(\Sigma) = (\pi(S), \pi(F), \pi(P))$ with

$$\begin{array}{rcl} \pi(S) & = & S \uplus \{time\}, \\ \pi(F) & = & F \uplus \{0 :\rightarrow time, + : time\ time \rightarrow time\}, \text{ and} \\ \pi(P) & = & \{p : time\ s_1 \ldots s_n \mid p : s_1 \ldots s_n \in P\} \uplus \\ & & \{=: time\ time, \leq : time\ time, < : time\ time\}, \end{array}$$

where $\uplus$ denotes the disjoint union of sets. The translation itself is defined by

$$\begin{array}{rcl} \pi(A) & = & \pi(A, 0, \emptyset) \\ \\ \pi(p(\mathbf{r}), t, C) & = & p(t, \mathbf{r}) \\ \\ \pi(\Box_I A, t, C) & = & \forall x(\{x \in I\} \rightarrow \pi(A, t + x, \{x \in I\} \cup C)) \\ \\ \pi(\Diamond_I A, t, C) & = & \exists x(\{x \in I\} \wedge \pi(A, t + x, \{x \in I\} \cup C)) \\ \\ \pi(A\ \mathcal{S}_I\ B, t, C) & = & \exists y(\{y < 0, y \in I\} \wedge \pi(B, t + y, \{y < 0, y \in I\} \cup C)) \\ & & \wedge \forall y'(\{y < y' < 0\} \rightarrow \pi(A, t + y', \{y < y' < 0\} \cup C)) \\ \\ \pi(A\ \mathcal{U}_I\ B, t, C) & = & \exists y(\{y > 0, y \in I\} \wedge \pi(B, t + y, \{y > 0, y \in I\} \cup C)) \\ & & \wedge \forall y'(\{0 < y' < y\} \rightarrow \pi(A, t + y', \{0 < y' < y\} \cup C)) \\ \\ \pi(A \leftarrow B, t, C) & = & \pi(A, t, C) \leftarrow \pi(B, t, C) \\ \\ \pi(A \wedge B, t, C) & = & \pi(A, t, C) \wedge \pi(B, t, C) \end{array}$$

$\mathbf{r}$ denotes a tuple of terms $r_1, \ldots, r_n$. Restrictions such as $x < \infty$ or $-\infty < x$ represent empty constraints and are dropped out from the set of constraints for being trivial.

For *EMTPL*'s formulae translation, the above definition can be used according to was presented in section 4. The only concern left is how to translate formulae of the from $\mathsf{At}(n,\ \phi)$ since:

$$\begin{array}{c} \mathsf{At}(n,\ \phi) \neq \Box_{[n,n]}\phi \\ \mathsf{At}(n,\ \phi) \neq \Diamond_{[n,n]}\phi \end{array}$$

We need then to extend π's definition to consider formulae affected by At operator. The extension add the following definition:

$$\pi(\mathsf{At}(n,\ A), t, C) \quad = \quad \pi(A, n, C)$$

From *EMTPL*'s definition C can be replaced by $\emptyset$ because there are no restrictions at this point. This is so because At is always the first operator analyzed. Independently of which moment is t it has no meddling in At formulae evaluation, neither in the formula affected by it.

We can also give an alternative definition for *EMTPL*'s language alone, let π' this function, it's definition is:

$$\begin{aligned}
\pi'(p(\mathbf{r}), t, C) &= p(t, \mathbf{r}) \\
\pi'(\mathsf{At}(n,\ A), t, C) &= \pi'(A, n, C) \\
\pi'(\Box\ A, t, C) &= \forall x(\{x \in (-\infty, \infty)\} \rightarrow \pi'(A, t + x, \{x \in (-\infty, \infty)\} \cup C)) \\
\pi'(\boxplus\ A, t, C) &= \forall x(\{x \in (0, \infty)\} \rightarrow \pi'(A, t + x, \{x \in (0, \infty)\} \cup C)) \\
\pi'(\boxminus\ A, t, C) &= \forall x(\{x \in (-\infty, 0)\} \rightarrow \pi'(A, t + x, \{x \in (-\infty, 0)\} \cup C)) \\
\pi'(\Diamond\ A, t, C) &= \exists x(\{x \in (-\infty, \infty)\} \wedge \pi'(A, t + x, \{x \in (-\infty, \infty)\} \cup C)) \\
\pi'(\diamondplus\ A, t, C) &= \exists x(\{x \in (0, \infty)\} \wedge \pi'(A, t + x, \{x \in (0, \infty)\} \cup C)) \\
\pi'(\diamondplus_v A, t, C) &= x = v \wedge \pi'(A, t + x, \{x = v\} \cup C)) \text{ or} \\
&= \exists x(\{x \in [v, v]\} \wedge \pi'(A, t + x, \{x \in [v, v]\} \cup C)) \\
\pi'(\diamondminus\ A, t, C) &= \exists x(\{x \in (0, \infty)\} \wedge \pi'(A, t - x, \{x \in (0, \infty)\} \cup C)) \\
\pi'(\diamondminus_v A, t, C) &= x = v \wedge \pi'(A, t - x, \{x = t - 1\} \cup C)) \text{ or} \\
&= \exists x(\{x \in [-v, -v]\} \wedge \pi'(A, t - x, \{x \in [-v, -v]\} \cup C)) \\
\pi'(A \rightarrow B, t, C) &= \pi'(A, t, C) \rightarrow \pi'(B, t, C) \\
\pi'(A \wedge B, t, C) &= \pi'(A, t, C) \wedge \pi'(B, t, C)
\end{aligned}$$

again $\mathbf{r}$ denotes a tuple of terms $r_1, \ldots, r_n$. For $\diamondplus_n A$ and $\diamondminus_n A$ the first $\wedge$ term is not necessary y we replace x for v so the definition could look like this:

$$\begin{aligned}
\pi'(\diamondplus_v A, t, C) &= \pi'(A, t + v, C)) \\
\pi'(\diamondminus_v A, t, C) &= \pi'(A, t - v, C))
\end{aligned}$$

The translation of formulae over both languages results into implication type formulae with constraints. In the case of Brzoska's approximation they look like this:

$$D ::= \ A \mid \forall x(C \rightarrow D) \mid D \leftarrow G$$

where C ranges over constraints. And constraints deals with an algebra that is capable of establish relations over the structure that represents time, relations such as $=$, $\leq$ or $<$. Despite this function definition, a lot of work is still to be done. How to prove the goals that are obtained from this translation.

EXAMPLE **1**

$$\mathsf{At}(8,\ \boxplus\ (critical\_temperature \rightarrow\ \Diamond\!\!\!\!+_3\ \ open\_valve))$$

Represents an "always rule". The rule establishes that 3 minutes after critical temperature is aware the security valve must be opened. This is set for all the moments after moment 8, may be because at that time the alarm is installed.

Let us translate it to first order from

$$\pi'(\mathsf{At}(8,\ \boxplus\ (critical\_temperature \rightarrow\ \Diamond\!\!\!\!+_3\ \ open\_valve)), t, \emptyset)$$

We follow this steps:

1. Considering At operator's rule we get:

$$\pi'(\boxplus\ \ (critical\_temperature \rightarrow\ \Diamond\!\!\!\!+_3\ \ open\_valve), 8, \emptyset)$$

2. Next we consider $\rightarrow$ connective reaching to:

$$\pi'(\boxplus\ \ critical\_temperature, 8, \emptyset) \rightarrow\ \pi'(\Diamond\!\!\!\!+_3\ \ open\_valve, 8, \emptyset)$$

3. We must resolve both calls to π' leading us to:
$\forall x[(\{x \in (0,\infty)\} \rightarrow (\pi'(critical\_temperature, 8+x, \{x \in (0,\infty)\}))) \rightarrow$
$\pi'(open\_valve, 8+3, \emptyset)]$

4. Since *critical_temperature* and *open_valve* are atoms we finally have:

$$\forall x[(\{x \in (0,\infty)\} \rightarrow (critical\_temperature(8+x)\ \rightarrow\ open\_valve(11))]$$

After the previous steps we reach to the following translated formula:

$$\forall x(\{x \in (0,\infty)\} \rightarrow (critical\_temperature(8+x)\ \rightarrow\ open\_valve(11)))$$

A second alternative to the previous translation process is to rewrite subformula $\boxplus\ (critical\_temperature \rightarrow\ \Diamond\!\!\!\!+_3\ \ open\_valve)$ it to MTL in this way:

$$\Box_{(0,\infty)}\ (critical\_temperature \rightarrow\ \Diamond_{[3,3]}\ open\_valve)$$

So now we can use π instead of π'. The translation process can be performed like this:

1. Using our added rule we get

$$\pi(\Box_{(0,\infty)}\ (critical\_temperature \rightarrow\ \Diamond_{[3,3]}\ open\_valve), 8, \emptyset)$$

2. Then we have

$$\pi(\Box_{(0,\infty)}\ critical\_temperature, 8, \emptyset) \rightarrow\ \pi(\Diamond_{[3,3]}\ open\_valve, 8, \emptyset)$$

3. After this we use $\Box_I$ operator's rule

$$\forall x(\{x \in (0,\infty)\} \rightarrow \pi(critical\_temperature, 8+x, \{x \in (0,\infty)\}) \ \rightarrow \\ \exists y(\{y \in [3,3]\} \wedge \pi(open\_valve, 8+3, \{y \in [3,3]\})))$$

4. Rearranging the formula we get

$$\forall x(\{x \in (0,\infty)\} \rightarrow critical\_temperature(8{+}x) \ \rightarrow \ \exists y(\{y \in [3,3]\} \wedge open\_valve(8{+}3)))$$

Summarily we start with

$$\mathsf{At}(8, \ \boxplus \ (critical\_temperature \rightarrow \ \Diamond_3 \ open\_valve))$$

and we arrive to one of these possible translations:

Through π' function:

$$\forall x(\{x \in (0,\infty)\} \rightarrow (critical\_temperature(8+x) \ \rightarrow \ open\_valve(11)))$$

Through π function:

$$\forall x(\{x \in (0,\infty)\} \rightarrow critical\_temperature(8{+}x) \ \rightarrow \ \exists y(\{y \in [3,3]\} \wedge open\_valve(8{+}3)))$$

As we can see the first translation lead us to a formula that is easier to read, perhaps because it is obtained from function π' that is developed for *EMTPL*'s sentences. The other approximation appears from two translations one from *EMTPL* to *MTL* and then to *first order*. This last option seems to be more expensive because the double translation performed, but the other one was not fully explored yet.

6 Conclusions and Future Work

We made a first attempt to set a direct relation between *EMTPL* and first order logic. In this way we use another similar approach made over a different metric language [Brzoska, 1998]. In order to be able of take advantage of this we have overview of *EMTPL* and the language used in [Brzoska, 1998] which is based on *MTL* logic.

We compare expressivity of Prior's logic [Prior, 1967a] and *MTL* [Koymans, 1990]. This comparison is needed to be able to set the difference between the languages, this is so because *MTL* abd Prior's logic are the logical base of each of them. It was deeply analyzed the relation from *EMTPL* to *Bounded universal (modality) goals* than the other way round because is more important for our purpose. As a consequence of this comparison, we found a function π' that allows translation from any *EMTPL*'s program to a *constraint logic program* an so we have a relation to first order logic.

It must be pointed out that additional proof over the new translation function and analysis of the consequences of the extension has to be made.

References

[Barringer et al., 1996] Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M. (1996). *The Imperative Future: Principles of Executable Temporal Logic*. Research Studies Press Ltd.

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

[Brzoska, 1998] Brzoska, C. (1998). Programming in metric temporal logic. *Theoretical Computer Science*, 202:55–125.

[Burgess, 1984] Burgess, J. (1984). Basic tense logic. In Gabbay, D. and Guenthner, F., editors, *Handbook of Philosophical Logic, vol. 2: Extension of Classical Logic*, pages 89–133. Reidel, Dordrecht.

[Cobo and Augusto, 1999a] Cobo, M. L. and Augusto, J. C. (1999a). EMTPL: A Pogramming Language for Temporal Deductive Data Bases. In *Proceedings de la* XIX International Conference of the Chilean Computer Science Society, pages 170–178, Talca, Chile.

[Cobo and Augusto, 1999b] Cobo, M. L. and Augusto, J. C. (1999b). Fundamentos lógicos e implementación de una extensión a temporal prolog. *The Journal of Computer Science and Technology (JCS&T), sponsored by ISTEC (Iberoamerican Science & Technology Education Consortium*, 1(2):22–36.

[Cobo and Augusto, 2000] Cobo, M. L. and Augusto, J. C. (2000). Towards a Programming Language Based on Prior's Metric Temporal Operators. In *Proceedings del VI Congreso Argentino de Cs. de la Computación, CACiC2000*, pages 453 –464.

[Kamp, 1968] Kamp, J. A. W. (1968). *On Tense Logic and the Theory of Order*. PhD thesis, University of California - Los Angeles.

[Kowalski, 1992] Kowalski, R. (1992). Database updates in the event calculus. *Journal of Logic Programming*, 12:121–146.

[Koymans, 1990] Koymans, R. (1990). Specifying real-time properties with metric temporal logic. *Real Time Systems*, 2:255–299.

[Ohlbach, 1991] Ohlbach, H. J. (1991). A resolution calculus for modal logics. In *Proceedings 9th International Conference on Automated Deduction, Lectures notes in Computer Science, vol 310*.

[Pnueli, 1977] Pnueli, A. (1977). The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*.

[Prior, 1967a] Prior, A. (1967a). *Past, Present and Future*. Clarendon Press.

[Prior, 1967b] Prior, A. (1967b). Stratified metric tense logic. *Theoria 33*, pages 28–38.

[Rescher and Urquhart, 1971] Rescher, N. and Urquhart, A. (1971). *Temporal Logic*. Springer-Verlag.