The specification and implementation of a first order logic for uncertain temporal domains

Ehric Ho
Bell-Northern Research
P.O. Box 3511, Station C
Ottawa, Ontario, Canada, K1Y 4H7

André Trudel
Jodrey School of Computer Science
Acadia University
Wolfville, Nova Scotia, Canada, B0P 1X0

Abstract

We formally define a first order logic that is suitable for representing and reasoning about uncertain temporal domains. The logic can represent both interval and point based qualitative and quantitative information. We provide a syntax, semantics, and axiomatization for the logic. We then describe the constraint logic programming implementation of the logic. The implementation, along with its formal specification, is suitable for tackling real world temporal problems.

1 Introduction

A popular approach in Artificial Intelligence for representing and reasoning about temporal domains is to use a first order (temporal) logic. Ideally, the first order logic should be formally defined and implemented. The formal definition, which consists of a syntax and semantics, removes all ambiguity from the structure and meaning of formulas. The implementation allows a user to solve real world temporal problems and evaluate the usefulness of the logic.

Very few temporal logics in AI are formally defined and implemented. Table 1 contains a list of some of the most influential and popular temporal logics. Note that out of the six logics, only the situation calculus is formally defined and implemented. This logic is primarily suited for simple single agent domains where actions occur in isolation. If the user has a complex domain which involves interval based information and/or simultaneous actions, one of the other logics must be chosen. When choosing another logic, the user must sacrifice either the formal definition or the implementation. We are not aware of a popular logic in AI suitable for complex real world domains that is both implemented and formally defined.

In the next section, we formally define a first order temporal logic by giving its syntax, semantics, and axioms. The proposed logic is based on the technique used in RGCH [4] of defining interval based information in terms of what is true at the point level, and using the Riemann integral. We then describe the logic's implementation in the constraint logic programming language CLP($\mathbb{R}$) [5]. One feature of the implementation is the ability to specify temporal uncertainty with probabilities. The implementation, which is available from the authors, can be thought of as a temporal expert system shell that is ready to be used with real world problems.

2 Proposed logic

There are four types of temporal information that need to be represented: point and interval based qualitative and quantitative information. An example of point based qualitative information is:

John is working at time t_1. (1)

An example of point based quantitative information is:

John is walking at a speed of 5 km/hr at time t_1. (2)

An example of interval based qualitative information is:

John is not working from time t_1 to t_2. (3)

An example of interval based quantitative information is:

John is walking at a speed of 3 km/hr from time t_1 to t_2. (4)

Both qualitative and quantitative temporal information are represented by real valued functions of several variables. The quantitative examples (2) and (4) are written as:

$$velocity(t_1, john) = 5, \text{ and}$$
$$velocity(t_1, t_2, john) = 3. \quad (5)$$

A 0-1 valued function is used to represent qualitative information. Zero and one represent falsity and truth respectively. Examples (1) and (3) are written as:

$$work(t_1, john) = 1, \text{ and}$$
$$work(t_1, t_2, john) = 0. \quad (6)$$

Formula (6) is equivalent to:

$$\forall T.\ t_1 < T < t_2 \rightarrow work(T, john) = 0 \quad (7)$$

Logic	Implementation	Syntax and semantics
Situation calculus [9, 7]	Yes	Yes
Allen [1]	Yes	No
McDermott [10]	?	No
Shoham [11]	No	Yes
Kowalski and Sergot [8]	Yes	No
BTK [2]	No	Yes

Table 1: Comparison of popular logics in AI

(i.e., John is not working at each point in the open interval). Similarly for formula (5).

The uniform representation of qualitative and quantitative information as real valued functions simplifies the implementation. Externally, the user makes a clear distinction between the two types of information. However, both types of information are represented with identical functions internally.

We use the approach from [4] where real valued functions can be integrated. When integrating a 0-1 valued qualitative function, the duration of truth is obtained. The result of integrating a real valued quantitative function depends on the function being integrated. For example, the integral of a velocity function is the total displacement:

$$displacement(t_1, t_2, john) = \int_{t_1}^{t_2} velocity(t, john)dt.$$

The syntax, semantics and axioms for the proposed logic are presented below.

2.1 Syntax

Given a set of non-temporal function symbols F_N, non-temporal constant symbols C_N, non-temporal variable symbols V_N, temporal function symbols F_T, temporal constant symbols C_T, and temporal variable symbols V_T (where $t \in V_T$), temporal terms (T_T) are defined as:

- $C_T, V_T \subseteq T_T$.
- If $g \in F_T$ is an n-ary ($n \geq 1$) temporal function symbol, $s_1, \ldots, s_n \in T_T$, then $g(s_1, \ldots, s_n) \in T_T$.

The temporal terms T_T give time a special status in the logic. Non-temporal terms (T_N) are defined as:

- $C_N, V_N \subseteq T_N$.
- If $f \in F_N$ is an n-ary ($n \geq 1$) non-temporal function symbol, $p_1, p_2 \in T_T$, $p_1 \leq p_2$, and $r_1, \ldots, r_{n-1} \in T_N$, then $f(p_1, p_2, r_1, \ldots, r_{n-2})$, $f(p_1, r_1, \ldots, r_{n-1})$, and $\int_{p_1}^{p_2} f(t, r_1, \ldots, r_{n-1})dt \in T_N$. The first argument to a non-temporal function is temporal, the second argument is temporal for interval based information and non-temporal for point based information, and the remaining arguments are non-temporal.
- If $r_1, r_2 \in T_T \cup T_N$, then $(r_1 + r_2), (r_1 - r_2), (r_1 \times r_2) \in T_N$.

Well-formed formulas (wffs) are defined as:

- If $\pi_1, \pi_2 \in T_N \cup T_T$, then $\pi_1 < \pi_2$, $\pi_1 \leq \pi_2$, $\pi_1 > \pi_2$, $\pi_1 \geq \pi_2$, and $\pi_1 = \pi_2$ are wffs.
- If ϕ_1, ϕ_2 are wffs, and $z \in V_T \cup V_N$ then $[\phi_1 \wedge \phi_2]$, $[\phi_1 \vee \phi_2]$, $[\phi_1 \rightarrow \phi_2]$, $[\phi_1 \leftrightarrow \phi_2]$, $[\neg\phi_1]$, $[\forall z.\ \phi_1]$, and $[\exists z.\ \phi_1]$ are wffs.

When there is no ambiguity, parentheses and square brackets are sometimes omitted.

Our proposed logic is based on another first order temporal logic called RGCH [4]. As in RGCH, we have real valued functions that are integrated to produce interval based information. One major difference with RGCH, is our distinction between temporal and non-temporal terms. This distinction, which does not exist in RGCH, allows the user to customize the temporal terms which are used for representing time. In RGCH, the user is forced to use the real numbers. Another difference with RGCH, are interval based real valued functions (e.g., formulas (5) and (6)). Only point based functions are allowed in RGCH.

2.2 Semantics

The semantic domain or ontology is $\mathbb{R}$. An interpretation for the proposed logic is a tuple $I = \langle MC, SF, MF \rangle$ where:

- $MC : C_T \cup C_N \mapsto \mathbb{R}$.
- SF is a set of piece wise continuous functions. Each element of SF is a function from $\mathbb{R}^n$ to $\mathbb{R}$ for some n.
- $MF : F_T \cup F_N \mapsto SF$.

A variable assignment is a function $VA : V_T \cup V_N \mapsto \mathbb{R}$. The function TA assigns an element of $\mathbb{R}$ to each temporal or non-temporal term as follows:

- If $x \in C_T \cup C_N$ then $TA(x) = MC(x)$.
- If $x \in V_T \cup V_N$ then $TA(x) = VA(x)$.
- If $g \in F_T$ is an n-ary ($n \geq 1$) temporal function symbol, and $s_1, \ldots, s_n \in T_T$, then $TA(g(s_1, \ldots, s_n)) = MF(g)(TA(s_1), \ldots, TA(s_n))$.

- If $f \in F_N$ is an n-ary ($n \geq 1$) non-temporal function symbol, $p_1, p_2 \in T_T$, $p_1 \leq p_2$, and $r_1, \ldots, r_{n-1} \in T_N$, then:

$$TA(f(p_1, p_2, r_1, \ldots, r_{n-2})) = MF(f)(TA(p_1), TA(p_2), TA(r_1), \ldots, TA(r_{n-2})),$$
$$TA(f(p_1, r_1, \ldots, r_{n-1})) = MF(f)(TA(p_1), TA(r_1), \ldots, TA(r_{n-1})),$$
$$TA(\int_{p_1}^{p_2} f(t, r_1, \ldots, r_{n-1})dt) = \int_{TA(p_1)}^{TA(p_2)} MF(f)(t, TA(r_1), \ldots, TA(r_{n-1}))dt.$$

  Note that the above definite integral is always defined because the integrand is a piece wise continuous function.

- If $r_1, r_2 \in T_T \cup T_N$, then:

$$TA((r_1 + r_2)) = TA(r_1) + TA(r_2),$$
$$TA((r_1 - r_2)) = TA(r_1) - TA(r_2),$$
$$TA((r_1 \times r_2)) = TA(r_1) \times TA(r_2)$$

The interpretation $I = \langle MC, SF, MF \rangle$ and variable assignment VA satisfy a formula φ (written $\models_I \varphi$ [VA]) under the following conditions:

- $\models_I \pi_1 < \pi_2$[VA] iff TA(π_1) < TA(π_2).
- $\models_I \pi_1 \leq \pi_2$[VA] iff TA($\pi_1$) $\leq$ TA(π_2).
- $\models_I \pi_1 > \pi_2$[VA] iff TA(π_1) > TA(π_2).
- $\models_I \pi_1 \geq \pi_2$[VA] iff TA($\pi_1$) $\geq$ TA(π_2).
- $\models_I \pi_1 = \pi_2$[VA] iff TA(π_1) = TA(π_2).
- $\models_I [\varphi_1 \wedge \varphi_2]$[VA] iff $\models_I \varphi_1$[VA] and $\models_I \varphi_2$[VA].
- $\models_I [\varphi_1 \vee \varphi_2]$[VA] iff $\models_I \varphi_1$[VA] or $\models_I \varphi_2$[VA].
- $\models_I [\varphi_1 \rightarrow \varphi_2]$[VA] iff $\models_I [\neg\varphi_1 \vee \varphi_2]$[VA].
- $\models_I [\varphi_1 \leftrightarrow \varphi_2]$[VA] iff $\models_I [\varphi_1 \rightarrow \varphi_2]$[VA] and $\models_I [\varphi_2 \rightarrow \varphi_1]$[VA].
- $\models_I [\neg\varphi]$[VA] iff $\not\models_I \varphi$[VA].
- $\models_I [\forall z.\ \varphi]$[VA] iff $\models_I \varphi$[VA'] for all VA' that agree with VA everywhere except possibly on z.
- $\models_I [\exists z.\ \varphi]$[VA] iff $\models_I \varphi$[VA'] for some VA' that agrees with VA everywhere except possibly on z.

2.3 Axioms

Besides the standard integral axioms for polynomials, we have an axiom for converting between interval and point based information:

$$[\ Predicate(Time_{begin},\ Time_{end},\ Arguments) = RealValuedFunction\ ] \Leftrightarrow [\ \forall T.\ Time_{begin} < T < Time_{end} \rightarrow Predicate(T,\ Arguments) = RealValuedFunction\ ]. \quad (8)$$

This axiom is used to convert formula (6) into formula (7).

Note that in axiom (8), $RealValuedFunction$ cannot contain any temporal terms. For example, the axiom is not applicable to $displacement(T_1, T_2, john) = T_2 - T_1$. This formula has no point based equivalent.

3 Implementation

Our proposed logic is implemented in the constraint logic programming language CLP($\Re$) [5]. Figure 1 gives an overview of the implemented system which consists of a temporal format specification component (Time File) and three typical expert system shell components: a knowledge base, an inference engine, and a user interface.

Recall that in the syntax (section 2.1), temporal terms are not explicitly specified. The onus is on the user to supply temporal constants, variables, and functions that are appropriate for the particular problem domain. The code for the temporal representation and its operators is stored in the "Time File".

The knowledge base contains problem domain dependent facts and rules that are stored in files, which are loaded during user consultation.

The inference engine is the work horse of the system and is divided into three modules: the interpreter, the analyser, and the estimator. The interpreter retrieves pertinent point and interval information from the knowledge base. Using the retrieved information and axioms (e.g., axiom (8)), the analyser attempts to derive required conclusions. The estimator, an optional feature, tries to compute missing point information. The final result produced by the inference engine is passed to the user interface.

The user interface is responsible for formatting, validating and packaging the input and output.

The system also optionally supports uncertainty with probabilities.

An overview of the implemented system is given in the remainder of the paper. See [6] for a detailed description. A sample session with the implementation is given in the Appendix.

4 Temporal format specification

The user must specify a temporal representation in the time file. A sample time file, which is included with the implementation, contains a specification for a temporal representation called MDHM. MDHM uses a Month:Day@Hour:Minute format. For example, 3:00am on June 10 is written as 6:10@3:0, and 2:30pm on July 8 is written as 7:8@14:30. Intervals are written as pairs of points. All points are assumed to be in the same calendar year (not a leap year). In practice, the finite number of time points is not a limitation since there are over 500,000 point constants.

It is possible to represent repeated time points. For example, X:1@Y:2 means the second minute of every hour of the first day of every month and X@3:Y means every minute of the third hour of every day.

4.1 Syntax of MDHM

Given a set of temporal constants C_T where $C_T \in \{0, \ldots, 59\}$, a set of temporal functions F_T={:, @}, disjoint sets of temporal variable symbol V_{Month},

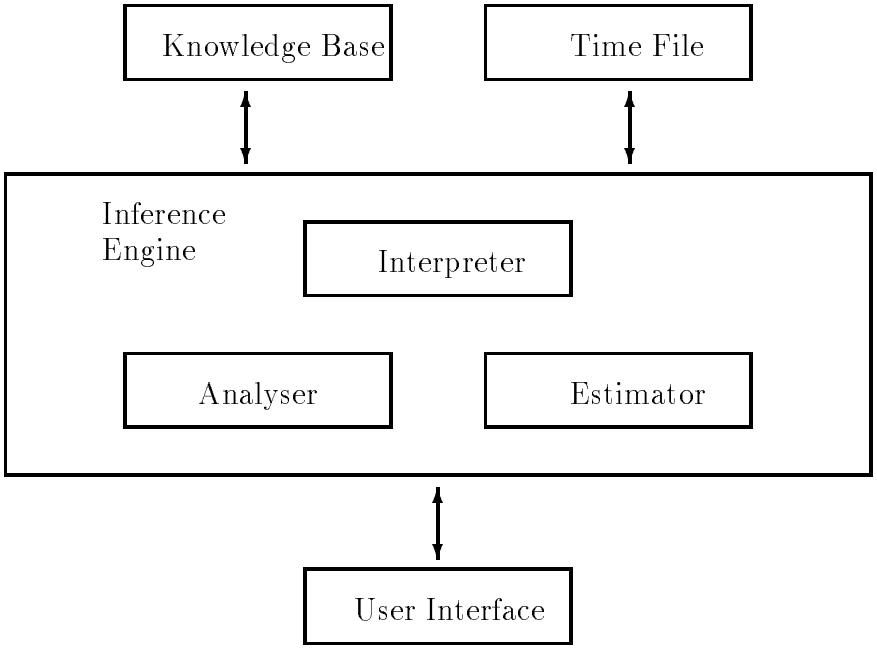

Figure 1: Implementation Overview

V_{Day}, V_{Hour}, V_{Minute}, V_{LHS}, V_{RHS}, and V_W, temporal terms T_T are defined as follows:

- Month $\in V_{Month} or \{1, \ldots, 12\}$,
- Day $\in V_{Day} or \{1, \ldots, 31\}$,
- Hour $\in V_{Hour} or \{0, \ldots, 23\}$,
- Minute $\in V_{Minute} or \{0, \ldots, 59\}$,
- LHS $\in V_{LHS}$ or Month:Day,
- RHS $\in V_{RHS}$ or Hour:Minute,
- V_W, LHS@RHS $\subseteq T_T$.

The Month:Day pair must be a valid combination. For example, 2:30 is invalid because February 30 is not a valid date.

4.2 Operators for MDHM

System defined operators, such as $<$ and $\leq$, can only be used to compare real numbers. Binary operators are introduced for comparing temporal terms: >@, >=@, <@, <=@ and =@. The comparison of temporal terms usually results in a system of constraints. For example, the solution to the inequality:

?- X@12:0 >@ Y@Z.

is:

X = Y, Z = HH:MM, HH $<$ 12; or
X = M:D1, Y = M:D2, Z = HH:MM, D2 $<$ D1; or
X = M1:D1, Y = M2:D2, Z = HH:MM, M2 $<$ M1.

5 Knowledge base

The syntax used in the knowledge base differs from the syntax given in section 2.1 for the proposed logic. Information is stored in the knowledge base with the kb prefixed predicates: kb_point/4, kb_interval/5 and kb_integral/5. Since we make no distinction between qualitative and quantitative information, only two predicates, kb_point/4 and kb_interval/5, are needed to represent point and interval based information. A third predicate, kb_integral/5, is used for representing integrals.

To represent a wff from the proposed logic in the knowledge base, we extract the temporal arguments. For example, $velocity(t_1, john) = 5\ km/hr$ is represented by the following fact in the knowledge base:

$$kb\_point(t_1, velocity(john), 5).$$

Note that inequalities, such as $velocity(t_1, john) > 0$, cannot be stored in the knowledge base.

The reason for changing the syntax in the knowledge base is to simplify the implementation. The kb predicates allow us to write generic code for point, interval, and integral information. For example, the code for kb_point/4 works for any possible point based information that can appear as its second argument. In addition, the kb predicates allow the implementation to easily distinguish between point and interval information.

To deal with uncertainty, we use probabilities. The last argument of each relation in the knowledge base is optional and is used to store a probability value.

If the argument is omitted, a probability of one is assumed.

In the remainder of this section, we discuss the relations used for representing point, interval, and integral information.

5.1 Point information

Qualitative and quantitative point based information is represented with the kb_point/4 predicate. To specify that $Fact$ has a $Value$ at $Time$ with $Probability$, we write

$$kb\_point(Time,\ Fact,\ Value,\ Probability)$$

where $Time$ is a time point, $Fact$ is the symbolic representation of the information, $Value$ is a real number or a polynomial, and $Probability$ (optional argument) is a real number which reflects the probability of the fact. For example,

$$kb\_point(12{:}31@12{:}00, work(john), \#true, 0.7)$$

expresses the fact that there is a 70 percent chance of John working on New Year's Eve at noon. #true and #false are constants which are defined as one and zero respectively.

The probability (the 4th argument) in the kb_point/4 predicate may be omitted. For example, the following represents the formula p $\wedge$ q $\rightarrow$ r without relying on probabilities:

$$kb\_point(T, r, Z) \text{ :- } kb\_point(T, p, Z),\ kb\_point(T, q, Z).$$

Probabilities at a point are related by the following axiom:

$$kb\_point(T, F, \#true, P) \Leftrightarrow kb\_point(T, F, \#false, 1-P). \quad (9)$$

The implementation makes a distinction between explicit and derived information in the knowledge base. Derived point based information is specified with the point_value/4 predicate. For example, consider the case where $q \rightarrow p$ at time 1:1@5:0 and q is true over the interval (1:1@1:0, 1:1@10:0):

$$kb\_point(1{:}1@5{:}0, p, Z) \text{:-} kb\_point(1{:}1@5{:}0, q, Z),\ kb\_interval(1{:}1@1{:}0, 1{:}1@10{:}0, q, \#true).$$

The query $kb\_point(1{:}1@5{:}0, p, \#true)$ fails because the knowledge base does not contain a fact that explicitly specifies that q is true at time 1:1@5:0. If instead we write:

$$kb\_point(1{:}1@5{:}0, p, Z) \text{ :- } point\_value(1{:}1@5{:}0, q, Z),\ kb\_interval(1{:}1@1{:}0, 1{:}1@10{:}0, q, \#true),$$

we can prove $kb\_point(1{:}1@5{:}0, p, \#true)$. The subgoal $point\_value(1{:}1@5{:}0, q, \#true)$ succeeds because we derive it from the interval information stored in the kb_interval relation. Similarly, there are interval_value/5 and integral_value/5 predicates for derived interval and integral information respectively.

5.2 Interval information

Qualitative and quantitative interval based information is captured with the kb_interval/5 predicate. The general format of kb_interval/5 is:

$$kb\_interval(Time_{begin},\ Time_{end},\ Fact,\ Value,\ Probability),$$

where $Time_{begin}$ is the beginning time of the interval, $Time_{end}$ is the ending time of the interval, $Fact$ is the symbolic representation of the information, $Value$ is a real number or a polynomial function, and $Probability$ (optional argument) is a real number which reflects the probability of the fact. Over the open interval from $Time_{begin}$ to $Time_{end}$, the value of $Fact$ is $Value$ with $Probability$ at each point. For example, assume John works from nine to five everyday and at each point in time between nine and five there is a 75% probability that he is actually working:

$$kb\_interval(X@9{:}0, X@17{:}0, work(john), \#true, 0.75).$$

Intervals are open at both ends. Values at end points are specified with kb_point/4. This allows us to specify facts that are true only at one of the end points. Consider the above example, John starts work at nine in the morning and leaves at 5:00pm (he does not work at 5:00pm):

$$kb\_point(X@9{:}0, work(john), \#true),\ kb\_interval(X@9{:}0, X@17{:}0, work(john), \#true).\ kb\_point(X@17{:}0, work(john), \#false).$$

The following example states that r is true when p is followed by q and they overlap:

$$p(A,B)\ \&\ q(C,D) \rightarrow r(A,D) \text{ when } A < C < B < D.$$

Assume the probability of r equals the probability of q. In our implemented system, we have:

$$kb\_interval(A, D, r, \#true, Pq) \text{ :-}\ interval\_value(A, B, p, \#true, Pp),\ interval\_value(C, D, q, \#true, Pq),\ A <@ C,\ C <@ B,\ B <@ D.$$

Note that interval_value/5 is used instead of kb_interval/5 since p or q may be computed from other information.

From axioms (8) and (9), the following axiom is derived:

$$kb\_interval(T_1, T_2, F, \#true, P) \Leftrightarrow kb\_interval(T_1, T_2, F, \#false, 1-P). \quad (10)$$

5.3 Integral information

It is possible to integrate qualitative and quantitative point based information. For example, John walks 5 km to work each morning between 8:00am and 9:00am is expressed as:

$$kb\_integral(X@8{:}0, X@9{:}0, velocity(john), 5).$$

The general format for the predicate is:

$$kb\_integral(T_1,\ T_2,\ F,\ Z,\ P)$$

which holds if $\int_{T_1}^{T_2} F(t)dt = Z$, with probability P (optional).

Subintervals must sometimes be considered when computing an integral. For example, assume John walks at a speed of five and three km/hr from 8:00 to 8:30 and 8:30 to 9:00 respectively. In order to compute his total displacement, two kb_interval/5 facts are used. The probability for the integral is defined as the weighted average of the P's of all subinterval values used in computing the integral.

6 Inference engine

The inference engine is made up of three components: an interpreter, an analyser and an estimator. The interpreter interacts with the knowledge base. The analyser deals with the conversion between point, interval, and integral information. Incomplete knowledge is handled by the estimator.

When a query is received, the interpreter accesses the knowledge base and returns pertinent information to the analyser and estimator. The interpreter ensures that returned information is within the time interval of interest which is set by the user. The interpreter also ensures that all temporal terms passed to the estimator are bound. The estimator only deals with completely specified time points. Another task of the interpreter is to compute the probability for rules when required.

The analyser deals with the transformation between point, interval and integral information. Point information may be computed from interval information. Integral information may be computed from one or more pieces of subinterval information. The analyser also computes probabilities for the transformed information. When point information is derived from interval information, it takes the probability of the corresponding interval information. When integrating an interval, the probability is defined as the weighted average of the probabilities of all the subintervals.

As its name implies, the estimator estimates unknown point information. Exponential decay functions are used to approximate probabilities (a similar approach is used in [3]). With quantitative information, linear interpolation is used to estimate a point value. For qualitative information, the estimated point value is based on the computed probabilities. User defined approximation functions can be associated with temporal information. See [6] for a detailed description of the estimator. We conclude with a qualitative example. John works between 9:00am and noon, and has lunch between 12:30 and 1:30pm:

$$kb\_interval(X@9{:}0, X@12{:}0, work(john),\ \#true, 1),$$

$$kb\_interval(X@12{:}30, X@13{:}30, work(john),\ \#false, 1).$$

The truth value of $work(john)$ is unknown between noon and 12:30. Figure 2 shows the estimated values of $work(john)$ over this interval. The probabilities are computed to be near 1 over the interval (12:00, 12:09), and near 0 over the interval (12:29, 12:30). We estimate $work(john)$ to be true and false over these intervals respectively. Between 12:09 and 12:29, the probabilities are estimated to be near 0.5 and we make no prediction about the truth value of $work(john)$. The coefficient of the decay function is determined by the length of the interval used in computing the estimated value (there is a three hour and one hour duration between 9:00-12:00 and 12:30-13:00 respectively). Hence, the duration of the two intervals estimated above are different.

7 User interface

The command line user interface supports a predicate, called ui, which takes a variable number of arguments. It validates input arguments and formats output data. All user interface routines are invoked from this ui predicate. The first argument of the predicate is a selector which describes the routine to be called. The following example specifies the time range that is of interest:

?-
$ui(change\_time\_range, 6{:}6@0{:}0, 6{:}7@23{:}59).$

Any information that lies outside of the above time range will not be reported to the user. The following example queries the point information of any fact in the knowledge base within the time boundary:

?-
ui(point_value,Time,Fact,Value,Probability).

When specifying temporal information, the first argument to *ui* makes the qualitative or quantitative distinction. For example, qualitative information is entered as:

?-
ui(point_truth,X@9:15,work(john),#true).

Quantitative information is entered as:

?-
$ui(point\_value, X@8{:}45, velocity(john), 3).$

8 Directions for future work

The following is a list of improvements to the implementation that we are contemplating:

- Re-implement the logic in a constraint logic programming language other than CLP($\mathbb{R}$) that supports graphical user interfaces. Instead of the current command line interface, all functions would be entered and displayed using graphs. Different colors could be used for point, interval, integral, and estimated values.

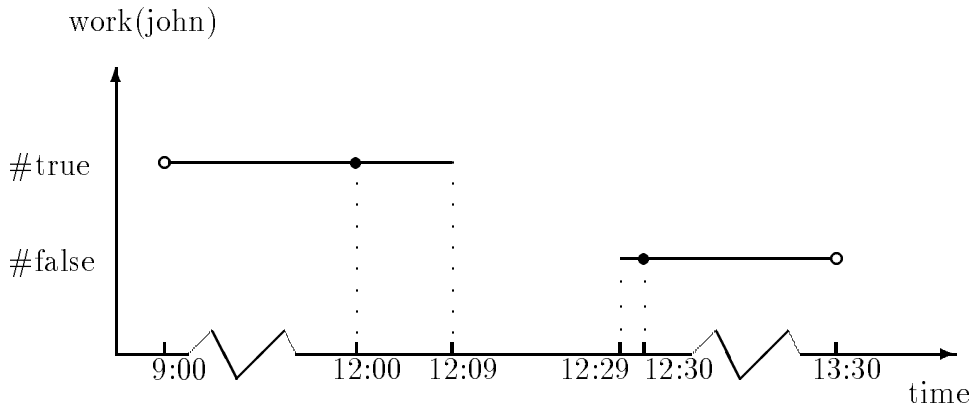

Figure 2: A plot of work(john) against time

- As in a typical expert system, the implementation should prompt the user for missing information.
- Units of measurement should be added to the system (e.g., miles and kilometres).
- Using our implementation, construct a temporal expert system to solve a complex real world problem.
- Each consultation is independent. Routines should be provided to save the environment of a session, which includes the user's time of interest and the certainty threshold.
- Currently, the system supports integration over truth. It should be an option for a user to integrate over true or false.

9 Conclusion

We proposed a first order logic for temporal domains. The logic is formally specified via a syntax, semantics, and axiomatization. Point and interval based qualitative and quantitative information are uniformly dealt with in the logic. Also, uncertainty can be represented with probabilities. The logic is implemented using a constraint logic programming language. The implementation, along with its formal specification, is suitable for tackling real world temporal problems.

Acknowledgements

Research of the first author is supported by an Acadia University Graduate Fellowship. Research of the second author is supported by Natural Sciences and Engineering Research Council of Canada grant OGP0046773. We would like to thank Scott Goodwin and Eric Neufeld for helpful comments on the paper.

A Example

We present an example, its representation in the proposed logic, and its knowledge base file. The example deals with John who works in an office from nine to five everyday including Saturday and Sunday. He takes an hour lunch break from 12:30 to 13:30. It is unknown if John works between 12:00 and 12:30. Every morning, he walks to work at a velocity of five and three kilometres per hour from eight to eight thirty and eight thirty to nine respectively. It takes him an hour to walk to work. The example is represented in the logic as follows:

work(X@0:00, john) = #false.
work(X@0:00, X@9:00, john) = #false.
work(X@9:00, john) = #true.
work(X@9:00, X@12:00, john) = #true.
work(X@12:00, john) = #true.
work(X@12:30, X@13:30, john) = #false.
work(X@13:30, john) = #true.
work(X@13:30, X@17:00, john) = #true.
work(X@17:00, john) = #false.
work(X@17:00, X@23:59, john) = #false.
work(X@23:59, john) = #false.
$\forall T$. work(T, john) $\rightarrow$ in_office(T, john).
velocity(8:00, 8:30, john) = 5.
velocity(8:30, 9:00, john) = 3.
$\forall X, Y$. displacement(X, Y, john) = $\int_X^Y$ velocity(t, john) dt.

Note that probabilities are omitted. Also note that it is unknown if John is working between 12:00 and 12:30. See figure 3 for a plot of the work function. The code in the knowledge base for the example is:

```
kb_point( X@9:0, work(john), #true ).
kb_interval( X@9:0, X@12:0, work(john), #true ).
kb_point( X@12:0, work(john), #true ).
kb_interval(X@12:30,X@13:30,work(john),#false).
kb_point( X@13:30, work(john), #true).
kb_interval(X@13:30,X@17:0,work(john),#true).
```

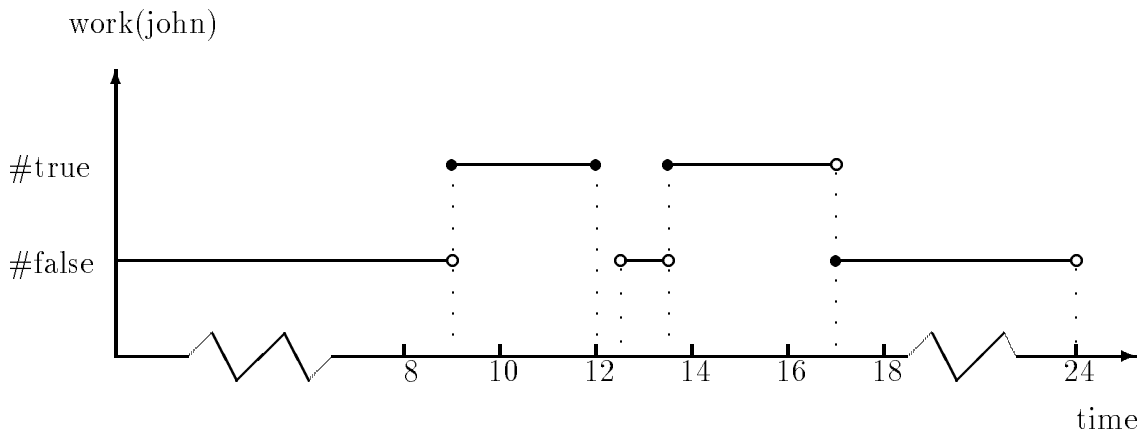

Figure 3: A plot of work(john) against time

```
kb_point( X@17:0, work(john), #false ).
kb_interval(X@17:0,Y@9:0,work(john),#false):-
    time_next_day( X, Y ).
kb_point( X, in_office(john), A, P ) :-
    point_value(X,work(john),A,P).
kb_interval( X, Y, in_office(john), A, P ) :-
    interval_value(X,Y,work(john),A,P).
kb_interval( X@8:0, X@8:30, velocity(john), 5 ).
kb_interval( X@8:30, X@9:0, velocity(john), 3 ).
kb_interval( X, Y, displacement(john), D, P ) :-
    integral_value(X,Y,velocity(john),D,P).
```

References

[1] J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[2] F. Bacchus, J. Tenenberg, and J. A. Koomen. A non-reified temporal logic. In *First International Conference on Principles of Knowledge Representation and Reasoning*, pages 2–10, Toronto, Canada, May 1989.

[3] Eugene Eberbach and André Trudel. Representing spatial and temporal uncertainty. In *the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 529–532, Mallorca, Spain, July 1992.

[4] Scott D. Goodwin, Eric Neufeld, and André Trudel. Temporal reasoning with real valued functions. In *Proceedings of the 2nd Pacific Rim International Conference on Artificial Intelligence*, Seoul, Korea, September 1992.

[5] Nevin C. Heintze, Joxan Jaffar, Spiro Michaylov, Peter J. Stuckey, and Roland H. C. Yap. *The CLP(R) Programmer's Manual*, 1992.

[6] Ehric K. H. Ho. The specification and implementation of a first order logc for uncertain temporal domains. Master's thesis, Acadia University, 1994.

[7] R. A. Kowalski. *Logic for Problem Solving.* Elsevier North Holland, New York, 1979.

[8] R. A. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[9] J. McCarthy. Progams with common sense. In R.J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 299–307, Los Altos, USA, 1985. Morgan Kaufmann.

[10] D. V. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

[11] Y. Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, 33:89–104, 1987.