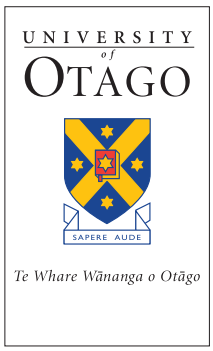

A Rule Language for Modelling and Monitoring Social Expectations in Multi-Agent Systems

Stephen Cranefield

The Information Science Discussion Paper Series

Number 2005/01
February 2005
ISSN 1177-455X

University of Otago

Department of Information Science

The Department of Information Science is one of seven departments that make up the School of Business at the University of Otago. The department offers courses of study leading to a major in Information Science within the BCom, BA and BSc degrees. In addition to undergraduate teaching, the department is also strongly involved in postgraduate research programmes leading to MCom, MA, MSc and PhD degrees. Research projects in spatial information processing, connectionist-based information systems, software engineering and software development, information engineering and database, software metrics, distributed information systems, multimedia information systems and information systems security are particularly well supported.

The views expressed in this paper are not necessarily those of the department as a whole. The accuracy of the information presented in this paper is the sole responsibility of the authors.

Correspondence

This paper represents work to date and may not necessarily form the basis for the authors' final conclusions relating to this topic. It is likely, however, that the paper will appear in some form in a journal or in conference proceedings in the near future. The authors would be pleased to receive correspondence in connection with any of the issues raised in this paper, or for subsequent publication details. Please write directly to the authors at the address provided below. (Details of final journal/conference publication venues for these papers are also provided on the Department's publications web pages: http://www.otago.ac.nz/informationscience/pubs/). Any other correspondence concerning the Series should be sent to the DPS Coordinator.

Department of Information Science
University of Otago
P O Box 56
Dunedin
NEW ZEALAND

Fax: +64 3 479 8311
email: dps@infoscience.otago.ac.nz
www: http://www.otago.ac.nz/informationscience/

A rule language for modelling and monitoring social expectations in multi-agent systems

Stephen Cranefield
Department of Information Science
University of Otago, PO Box 56, Dunedin, New Zealand
scranefield@infoscience.otago.ac.nz

Abstract

This paper proposes a rule language for defining social expectations based on a metric interval temporal logic with past and future modalities and a current time binding operator. An algorithm for run-time monitoring compliance of rules in this language based on formula progression is also presented.

1 Introduction

The study of *electronic institutions*—explicit declarative models of the rules governing particular open systems of autonomous agents—has gained much recent attention [Cortés, 2004]. An institution provides a social model of a multi-agent system in which agents agree (by the act of joining the society) or are required to conform to particular norms of behaviour and role and empowerment structures. However, in an open system it is not sufficient to simply formally or semi-formally define an institution and hope that agents will follow its rules. As in human society, the successful functioning of an institution requires that all (or at least most) members will conform. There is therefore a need for mechanisms to ensure the conformance of all agents by formal verification of agents' code—which is not possible in an open system, or by run-time compliance checking.

There has been a significant amount of recent research on statically verifying properties of institutions as well as interpreting institutions to manage or guide agent interaction (examples include several papers in the DALT'04 workshop [Leite *et al.*, 2004] as well as earlier work such as that by Huget *et al.* [2002] and Cliffe and Padget [2002]). However, there has been little attention paid to mechanisms for run-time compliance checking, i.e. monitoring events in a running agent system, determining the future expectations of agents' behaviour according to norms of the institution, and checking if these are fulfilled or violated. This paper focuses on this issue.

Verdicchio and Colombetti [2003; 2004a] have developed a formal model of social commitment with its semantics based on a propositional branching time logic with future and past-time modal operators (CTL$^{\pm}$), but with an axiomatic account of events and commitments expressed using predicate logic and CTL$^{\pm}$ operators together. This paper presents the results of an investigation into how implication formulae of the style used by Verdicchio and Colombetti can be formally characterised and, with appropriate syntax restrictions, be used for practical reasoning by agents at run time. An important requirement for this purpose is the ability to reason efficiently about how event occurrences relate to specific points or intervals in time. We have therefore developed a logic named hyMITL$^{\pm}$ that combines CTL$^{\pm}$ with Metric Interval Temporal Logic (MITL) [Alur *et al.*, 1996], as well as features of hybrid logics [Blackburn *et al.*, 2001]. We present a subset of hyMITL$^{\pm}$ that provides a rule language for defining social expectations and show how the technique of formula progression from the planning system TLPlan [Bacchus and Kabanza, 2000] can be used to monitor social expectations until they are fulfilled or violated.

The structure of the paper is as follows. Sections 2 and 3 define the syntax and semantics of hyMITL$^{\pm}$, respectively, with the rule language presented in Section 4 and an example of its use in Section 5. Section 6 gives details of the compliance-testing algorithm. Finally some related work is discussed in Section 7 and Section 8 concludes the paper.

2 Syntax of hyMITL$^{\pm}$

Formulae of hyMITL$^{\pm}$ are defined by the following grammar:

$$\begin{aligned}
\phi &::= p \mid \neg\phi \mid \phi \wedge \phi \mid \forall x.\phi_x \mid \mathsf{X}^{+}\phi \mid \mathsf{X}^{-}\phi \mid \\
&\quad\ \phi\,\mathsf{U}^{+}_{I}\phi \mid \phi\,\mathsf{U}^{-}_{I}\phi \mid \mathsf{A}\,\phi \mid \mathsf{E}\,\phi \mid \downarrow^{u} x.\phi_x \mid I \\
I &::= (-\infty,+\infty) \mid [b,b] \mid [b,b) \mid (b,b] \mid (b,b) \\
b &::= a \mid +r \mid -r
\end{aligned}$$

where:

- p is an atomic formula from a first order language L.
- ϕ_x denotes a formula ϕ in which variable x is free (i.e. not bound by $\forall$ or $\downarrow$).
- a and r are terms, possibly containing variables, that denote (respectively) absolute and relative points in time[1].

[1] We do not define a language for these terms in this paper, but note that Verdicchio and Colombetti [2004b] have proposed a suitable language, which has inspired the treatment here.

- u is a unit selector on the $\downarrow$ binding operator, referring to the desired granularity of time (e.g. year or minute) for binding x to the current time. A value of now indicates maximum precision.

We constrain the use of variables within interval bounds b: any such variables must be bound by an enclosing $\downarrow$ operator.

In this logic, the temporal operators X (the next/previous state) and U (until) can be applied in the future direction (when adorned with a superscript '$+$') or the past (indicated by a '$-$'). Following MITL[2] [Alur *et al.*, 1996], the two U operators are qualified by an interval I that can be open or closed at each end (depending on whether a round or square bracket is used, respectively). The meaning of $\mathsf{X}^+\phi$ is that ϕ is true in the next state, and $\phi\,\mathsf{U}_I^+\psi$ asserts that ϕ will remain true from the current state for some (possibly empty) sequence of consecutive future states, followed by a state that is within the time interval I and for which ψ holds. X^- and U_I^- are defined similarly, but in the past direction.

The bounds of intervals can be specified either relatively or absolutely—a prefix of "$+$" or "$-$" indicates a relative time value. Relative times (except for values of plus or minus zero) must indicate the units used, and the language for expressing time points must define a syntax for this, e.g. "-3 hours". When qualifying U^-, the interval bounds are written in the reverse order from usual, e.g. $[-2\,\text{hours}, -3\,\text{hours}]$.

A and E are temporal path quantifiers. They assert that the formula that follows the operator applies to all, or respectively at least one, of the possible sequences of states passing through the current state.

The $\downarrow$ operator is the "binder" operator used in hybrid logics [Blackburn *et al.*, 2001]. It binds a variable to a term denoting the current date/time, using the same syntax as absolute interval bounds. The optional unit selector u is a time unit constant from the date/time sublanguage and indicates that the variable should be bound to the time point resulting from rounding down the current date/time to a particular degree of precision, e.g. to the start of the current year, month or day.

The final type of formula is an interval formula. This is true if the timepoint associated with the current state is within the interval[3]. The usual abbreviations of predicate logic are defined for disjunction ($\vee$), implication ($\rightarrow$) and existential quantification ($\exists$). We also use the standard abbreviations for existential and universal quantification over states in a path: $\mathsf{F}_I^+\phi \equiv \mathsf{true}\,\mathsf{U}_I^+\phi$ and $\mathsf{G}_I^+\phi \equiv \neg\mathsf{F}_I^+\neg\phi$, with similar definitions for F_I^- and G_I^-. We define future and past "weak until"

[2]We do not choose to qualify X^- and X^+ by an interval. Although one version [Haslum, 2002] of MITL qualifies its next-state operator in this way, the version used in TLPlan [Bacchus and Kabanza, 1998] does not, and, in fact, the original definition of MITL [Alur *et al.*, 1996] did not include this operator at all.

[3]This is a generalisation of the notion of a *nominal* in hybrid logics: a formula that names a point in a model and is true if the current point is the one named.

$\langle M,p,V\rangle \models \phi$	where ϕ is an atomic formula, iff $\langle p_0, V\rangle \models \phi$.
$\langle M,p,V\rangle \models \neg\phi$	iff $\langle M,p,V\rangle \not\models \phi$.
$\langle M,p,V\rangle \models \phi \wedge \psi$	iff $\langle M,p,V\rangle \models \phi$ and $\langle M,p,V\rangle \models \psi$.
$\langle M,p,V\rangle \models \forall x.\phi_x$	iff for all $d \in D$, $\langle M,p,V[d/x]\rangle \models \phi_x$.
$\langle M,p,V\rangle \models \mathsf{X}^+\phi$	iff $\langle M,p^1,V\rangle \models \phi$.
$\langle M,p,V\rangle \models \mathsf{X}^-\phi$	iff for some path q, $q^1 = p$ and $\langle M,q,V\rangle \models \phi$.
$\langle M,p,V\rangle \models \phi\,\mathsf{U}_I^+\psi$	iff for some $n \geq 0$, $\langle M,p^n,V\rangle \models \psi$, $\tau(p^n) \in I^{M,V}$ and for all m s.t. $0 \leq m < n$, $\langle M,p^m,V\rangle \models \phi$.
$\langle M,p,V\rangle \models \phi\,\mathsf{U}_I^-\psi$	iff for some path q and for some n, $q^n = p$, $\langle M,q,V\rangle \models \psi$, $\tau(q) \in I^{M,V}$, and for all m s.t. $0 < m \leq n$, $\langle M,q^m,V\rangle \models \phi$.
$\langle M,p,V\rangle \models \mathsf{A}\,\phi$	iff for all $q \in \mathit{Paths}(p_0)$, $\langle M,q,V\rangle \models \phi$.
$\langle M,p,V\rangle \models \mathsf{E}\,\phi$	iff for some $q \in \mathit{Paths}(p_0)$, $\langle M,q,V\rangle \models \phi$.
$\langle M,p,V\rangle \models \downarrow^u x.\phi_x$	iff $\langle M,p,V[\mathit{floor}(\tau(p),u^M)/x]\rangle \models \phi_x$.
$\langle M,p,V\rangle \models I$	where I is an interval formula, iff $\tau(p) \in I^{M,V}$.

Figure 1: The semantics of hyMITL$^\pm$

operators in the following way[4]:

$$\phi\,\mathsf{W}_{[l,u]}^+\psi \;\equiv \downarrow t.(\mathsf{G}_{[t,u]}^+\phi \vee \phi\,\mathsf{U}_{[l,u]}^+\psi)$$

with similar definitions for intervals with open bounds and for W^-. Finally, if a temporal operator is qualified by the interval $(-\infty,+\infty)$, we allow this to be suppressed for brevity.

3 Semantics of hyMITL$^\pm$

Let S be a set of states, each being a first-order model for the language L over the fixed domain D, and all having the same interpretation for the date/time sublanguage of L. We denote the image of date terms under this shared interpretation by *Date* and the image of the set of time unit constants by U.

A hyMITL$^\pm$ model M is a tuple $\langle S, <, \tau, \prec, \mathit{floor}\rangle$ where $<$ is a a total order relation on *Date*, τ is a function mapping from S into *Date*, $\prec$ is a state predecessor relation in which every state has a unique predecessor and a non-empty set of successors and which is consistent with the ordering on dates: $\forall s_1, s_2 \in S, s_1 \prec s_2 \rightarrow \tau(s_1) < \tau(s_2)$, and *floor* is a function from *Date* $\times\, U$ to *Date* representing the notion of 'rounding down' a time value to a particular level of granularity[5].

A *path* in a model is an infinite sequence of states with each pair of adjacent elements s_i and s_{i+1} satisfying $s_i \prec s_{i+1}$.

[4]A straightforward extension of the usual definition would give $\phi\,\mathsf{W}_I^+\psi \equiv \mathsf{G}_I^+\phi \vee \phi\,\mathsf{U}_I^+\psi$, which would be true if ϕ is true throughout a future interval I but not before then.

[5]The *floor* function is subject to a number of semantic constraints that we do not discuss here.

We write p_i to denote element $i+1$ of a path p (with indices starting at 0), p^n for the subsequence of p beginning with state p_n, and extend the date function τ to operate on paths: $\tau(p) = \tau(p_0)$. The set of all paths starting from state s is denoted *Paths*(s).

Let V be a variable assignment mapping variables to elements of the domain D. The notation $V[d/x]$ represents a variable assignment that is identical to V, except with x mapping to d. For interval expressions I in our language we write $I^{M,V}$ (or just I^M for ground interval expressions) to denote the interval in *Date* formed by applying V and the interpretation of date constants and function symbols that is common in all states of M to the bounds of I. We define $(-\infty,+\infty)^{M,V} = \mathit{Date}$. The interpretation in D of a ground term t in the date/time sublanguage is denoted t^M.

The truth of a formula in a model M and for a path p in M is then defined as shown in Figure 1.

4 The Rule Language

We now identify a subset of hyMITL$^{\pm}$ that is suitable for encoding social expectations in a form that can be used in run-time compliance testing. We define the language R to be the set of all formulae of the following form:

$$\mathsf{AG}^{+}\,\forall_{1\leq i\leq n} x_i.(\phi \rightarrow \psi)$$

for $n \geq 0$, where:

- ϕ and ψ are linear-time formulae, i.e. they do not contain A or E;
- *free_variables*(ϕ) = *free_variables*$(\psi) = \{x_1, \ldots, x_n\}$;
- ϕ and ψ do not contain any occurrences of $\forall$, except when represented using the $\exists$ abbreviation as outlined in the following clause:
- Any occurrence of $\exists$ must be of the following restricted form[6]: $\exists x.(\alpha_x \wedge \beta_x)$ where x is free in α_x and β_x.

The intent of the last restriction is that matching α_x to the current state should produce a finite set of variable bindings for x, each of which should leave β_x with no free variables. This can not be expressed syntactically and remains the responsibility of the rule designer (although any insufficiently instantiated instances of β_x can be detected and discarded at run time).

Rules of this form are intended to be used in the following compliance-testing process:

> Given a current state and the history of all prior states and their associated times, for each rule, match the left hand side (ϕ) against the current state and history, resulting in a set of instances of the right hand side (ψ). Add these instances to the set of current expectations, then check all expectations to see which are fulfilled or violated. Any expectations that can not yet be evaluated because they involve future modalities will be 'progressed' to the next state when it is created by an event observation.

[6]This is equivalent to TLPlan's bounded existential quantification [Bacchus and Kabanza, 2000].

This process requires that the left hand side of a rule can be matched against the current state and history, leaving no residual formula involving future states. This is not a syntactic constraint—future modalities can legitimately appear in the left hand side of a rule: consider $\mathsf{F}_I^{-}(\alpha \wedge \mathsf{X}^{+}\beta)$. However, this constraint can be checked at run time, with a rule application simply failing if its left hand side can not be matched using the current state and history alone. The rule designer must also use his/her knowledge of the domain model to ensure that the left hand side can only have a finite (and preferably bounded) number of matches for any state and history.

The following section presents an example rule in this notation and then Section 6 describes the compliance-testing process in more detail.

5 Example

Consider the case of an agent that can provide weekly reports on a particular market for an annual fee. A potential customer is advised of a fee for the service and has one week to confirm the order and make payment. After this time, the price is not valid and a new quote must be sought. Once payment is made, the service-providing agent is committed to sending a report to the customer once a week for 52 weeks or until the customer cancels the order. If the customer cancels the order before 52 weeks have passed, it may be eligible for a partial refund, but we do not model that here.

Figure 2 shows how the service-providing agent could encode its conditional commitment using our rule syntax (where *p* and *c* are the names of service provider and customer agents respectively, *t* is an expression representing the time the offer was made, and *amount* and *prod_id* are expressions representing the amount to be paid for the service and the service provider's identification number for this product). This rule could be sent from the provider to the customer as the content of a communicative act that explicitly asserts the commitment is being made. Alternatively, making this commitment may be an "institutional action" [Verdicchio and Colombetti, 2004a] that is inferred by both p and c to have occurred as a result of a particular dialogue between them having been completed.

The rule in Figure 2 states that if the current state is one in which c has just made payment for the service, and the current state is within the one week period from the time this offer is made (time *t*) then weekly reports will be sent during the next 52 weeks until p optionally cancels the order. The assertion that weekly reports will be sent (the left hand side of the W^{+} operator) is encoded as the implication that if the report has not been sent since the start of the current week then it will be sent some time before the end of the week[7].

This rule assumes that the actions of making a payment, sending a report and cancelling an order can be observed by both agents as occurring at a unique well defined time. In practice, agents will not observe events simultaneously, and their clocks cannot be guaranteed to be perfectly synchronised. However, if these actions are implemented by sending messages, the sending time (as recorded by the sender

[7]A tighter specification could identify a particular day of the week on which the report will be sent

$$\mathsf{AG}^{+}\ (\text{Done}(c,\ \text{make\_payment}(c, p, \textit{amount}, \textit{prod\_num}))\ \wedge\ [t,\ t{+}1\ \text{week})\ \rightarrow$$
$$\downarrow^{\text{week}} \text{w}.((\neg\, \mathsf{F}^{-}_{[-0,\ \text{w}]}\text{Done}(p,\ \text{send\_report}(c, \textit{prod\_num}, \text{w}))\ \rightarrow\ \mathsf{F}^{+}_{(+0,\ \text{w}+1\ \text{week})}\text{Done}(p,\ \text{send\_report}(c, \textit{prod\_num}, \text{w})))$$
$$\mathsf{W}^{+}_{[+0,\ \text{w}+52\ \text{weeks}]}$$
$$\text{Done}(c,\ \text{cancel\_order}(c, p, \textit{prod\_num})))$$

Figure 2: A rule expressing the terms of service offered by agent p

function *check_state*
inputs: A history of state/time pairs $h = \langle (s_0, t_0), \ldots, (s_n, t_n) \rangle$, where s_n is the new state to be checked, a set of formulae E_{n-1} representing expectations that could not be fully evaluated in s_{n-1}, and a set of rules R.
outputs: A set of partially evaluated formulae E_n and a set of notification assertions N
begin
  vars $E = \textit{progress\_formulae}(E_{n-1}, t_n - t_{n-1}) \cup$
    $\textit{new\_expectations}(h, R)$,
    $E_n = \varnothing$, $N = \varnothing$, $dph = \textit{gensym}()$
  for each ϕ in E:
    var $\phi' = \textit{peval}(\phi, h, n, dph)$
    if $\phi' = \mathsf{true}$, $N = N \cup \{\mathsf{fulfilled}(\phi)\}$
    else if $\phi' = \mathsf{false}$, $N = N \cup \{\mathsf{violated}(\phi)\}$
    else if $\textit{worth\_progressing}(\phi')$, $E_n = E_n \cup \{\phi'\}$
  return $\langle E_n, N \rangle$
end

Figure 3: The main algorithm: *check_state*

and included in the message header) can be taken as the time the action occurs. Provided that the intervals in a commitment are of a significantly greater magnitude than the likely clock slippage and message delivery delay, this approximation should be acceptable. The possibility of significantly inaccurate message times (either forged or caused by inaccurate clocks) is difficult to deal with; however, for ease of modelling, the attempted detection of such occurrences (if possible and required) is best handled by a separate mechanism.

6 The Compliance Testing Process

The compliance testing process is performed by function *check_state* shown in Figure 3. This should be called by an agent when it has performed an action or observed some event that it (or the agent programmer) considers significant. The *check_state* function assumes that the agent has already created a new state name and asserted into its world model for that state any facts that it knows to hold (including facts expressing the occurrence of the actions and events that are considered to have triggered the transition to a new state). The function receives as arguments the history of states, the current unfulfilled expectations, and the set of rules defining the social expectations of the institution to which the agent currently belongs.

The function *progress_formulae* applies a modified version of the *progress* algorithm of Bacchus and Kabanza [1998] to every unfulfilled expectation from the previous state, with the diffence in time between the previous and current state supplied as an additional argument. This algorithm generates a formula expressing what needs to be true in the new state if the input expectation was true in the previous state, but was not able to be evaluated there. For example, $\textit{progress}(\mathsf{X}^{+}\phi, \Delta) = \phi$, and if the interval I is not in the past, $\textit{progress}(\phi\, \mathsf{U}^{+}_{I}\psi, \Delta)$ has the following value:

$$\ulcorner \textit{progress}(\psi, \Delta) \urcorner \vee (\ulcorner \textit{progress}(\phi, \Delta) \urcorner \wedge \phi\, \mathsf{U}^{+}_{\ulcorner \textit{arb}(I, -\Delta) \urcorner}\psi)$$

where corner quotes ($\ulcorner$ and $\urcorner$) are used to indicate the parts of the formula that should be evaluated to generate subexpressions, and *arb* ("adjust relative bounds") is a function that takes an interval term and a relative time and returns an adjusted version of the interval with that relative time added to any relative bounds appearing in the interval.

The function *new_expectations* matches the left hand side of each rule to the state history, and for each resulting rule instantiation, adds the instantiated right hand side to the set of new expectations that this function returns. Any expectations that are not fully instantiated by this process (i.e. they have free variables) are discarded. The *match* function used in this process is shown in Figure 4. It is presented in the figure as a non-deterministic function that can either fail or return multiple variable bindings (one at a time). The notation $\llcorner t \lrcorner$ indicates a mapping from the semantic to the syntactic domain that chooses a term that names a given time point. This must be built in to an implementation.

Once the new expectations have been computed, each formula in the combined set of old and new expectations is partially evaluated using the function *peval* shown in Figure 5. This uses the history to evaluate a formula as much as possible, resulting in true or false if the truth of the formula can be determined yet, and otherwise returning a formula equivalent to the original one (given the facts in the history states) but modified where possible to make progression and future evaluation easier. The *simplify* function removes double negations and simplifies formulae that have true and false as subformulae. In the U^{+}_{I} case of the *match* function, when $t_i < I$, an *arb* functional term is inserted into the resulting formula instead of being evaluated. As the time of the next state is not yet known, the evaluation must be delayed until the formula is progressed when the next state is generated and *check_state* is called again. For this reason, a symbol dph ("delta place holder") is provided as an argument to *peval*. When the arb term is encountered during progression, the place holder is replaced by the Δ argument and the function is evaluated. In the case for $\downarrow$ formulae, a time constant must be generated using the *floor* function that is part of the semantic domain.

function *match* (non-deterministic)
inputs: A formula ϕ, a history of state/time pairs $h = \langle (s_0, t_0), \ldots, (s_n, t_n) \rangle$, and an index i for the current state
output: A variable binding or $\perp$ (failure)
begin
  if $(i<0 \vee i>n)$ **fail**
  case ϕ is an atomic formula:
    choose any σ s.t. $Dom(\sigma) = vars(\phi)$ and $\langle s_i, \sigma \rangle \models \phi$
    return σ
  case $\phi = \neg\phi_1$:
    if *free_variables*$(\phi) = \varnothing$ and *match*(ϕ_1, h, i) fails, **return** $\{\}$
    else fail
  case $\phi = \phi_1 \wedge \phi_2$:
    choose $\sigma =$ *match*(ϕ_1, h, i) and **return** *match*$(\phi_2\sigma, h, i)$
  case $\phi = \exists x.(\phi_1 \wedge \phi_2)$:
    let Ψ be the set $\{\phi_2\sigma | \sigma =$ *match*(ϕ_1, h, i) $\wedge$ *free_variables*$(\phi_2\sigma) = \varnothing\}$
    if $\Psi \neq \varnothing$ **return** $\bigvee_{\psi \in \Psi} \psi$ **else fail**
  case $\phi = \mathsf{X}^+\phi_1$: **return** *match*$(\phi_1, h, i+1)$
  case $\phi = \mathsf{X}^-\phi_1$: **return** *match*$(\phi_1, h, i-1)$
  case $\phi = \phi_1 \mathsf{U}_I^+ \phi_2$:
    begin
      case $t_i > I$: **fail**
      case $t_i < I$: **choose** $\sigma =$ *match*(ϕ_1, h, i)
        return *match*$(\phi_1\sigma \mathsf{U}^+_{arb(I, t_i - t_{i+1})} \phi_2\sigma, h, i+1)$
      case $t_i \in I$: **either choose** σ as for case $t_i < I$
        or choose $\sigma =$ *match*(ϕ_2, h, i)
    end
  case $\phi = \phi_1 \mathsf{U}_I^- \phi_2$:
    (Mirror image of U_I^+ *case — omitted due to lack of space)*
  case $\phi = \downarrow^u x.\phi_x$:
    return *match*$(\phi_x[\lfloor floor(t_i, u^M) \rfloor / x], h, i)$
  case $\phi = I$, where I is an interval formula:
    if $t_i \in I$ **return** $\{\}$ **else fail**
end

Figure 4: The *match* function

The test *worth_progressing* can be used to discard expectations that could not be evaluated for reasons other than lack of future information, such as atomic formulae for which *peval* did not return true or false (if closed-world reasoning is not used within states) or formulae with past modalities that needed a longer history to be evaluated.

The use of the *peval* function means that the progression function does not need to handle atomic formulae—so it needs no state argument as in the original definition [Bacchus and Kabanza, 1998]—or $\exists$ or $\downarrow$ formulae.

In the *match* and *peval* functions, comparisons and operations involving intervals are required. The semantics of hyMITL$^{\pm}$ assumed that there is a date/time sublanguage with a total order $<$ on time, and in the following we assume we have an implementation of that relation, extended in the obvious way to allow comparisons with $\pm\infty$. As both absolute and relative times can appear in interval bounds, we extend equality and the $<$ relation to apply to the combined set of absolute and relative times: for any absolute time t and relative time r, $t \neq r$, $t < r \equiv r > 0$ and $r < t \equiv r < 0$. We define membership of a point in an interval as follows:

$$\begin{aligned} t \in [l, u] \equiv ((l < t < u) &\vee (l \text{ is a relative bound and } l = \pm 0) \\ &\vee (l \text{ is an absolute bound and } l = t) \\ &\vee (u \text{ is a relative bound and } u = \pm 0) \\ &\vee (u \text{ is an absolute bound and } u = t)) \end{aligned}$$

with similar (but simpler) definitions for intervals with open ends. A consequence of these definitions is that (e.g.) $\forall t \in Date, t \in [-1, +1]$. The intuition is that all timepoints will be in a interval defined in that way, at the moment of comparison.

For a time point t and interval I with lower bound l we define $t < I \equiv (t < l \vee (t = l \wedge l \notin I))$. We define $t > I$ in similar way.

7 Related Work

The closest work to that described here is the SOCS-SI system [Alberti *et al.*, 2004], which performs run-time protocol compliance testing based on *social integrity constraints*: rules that express positive and negative expectations as the consequences of observed actions. Abductive inference is used to generate expectations during run time and these are monitored to determine their fulfilment or violation. The semantics do not include an underlying model of time. Instead explicit time variables are associated with the observation and expectation atoms in rules, and constraint logic programming constraints can be used to relate these time points.

Mallya *et al.* [2004] proposed a language for representing social commitments that have a temporal nature. Their notation uses interval expressions representing universal or existential state quantification within these intervals, with semantics based on a timed version of CTL. They provided an analysis showing how to determine when a given commitment could be known to be fulfilled or violated.

Verdicchio and Colombetti [2004b] presented a rich language for making statements involving time, including interval expressions that are a generalisation of the work by Mallya *et al.* The language is defined axiomatically, and so would not support run-time use as efficiently as the approach proposed here, where metric time is built in to the semantics and evaluation mechanism. This work inspired the use of a date/time sublanguage in the present paper.

8 Conclusion

This paper has defined a rule language for defining social expectations based on a metric interval temporal logic and has presented an algorithm that can be used at run time in a multi-agent system to monitor when expectations are generated, fulfilled and violated—either for the system as a whole (if all events can be detected by a specialised monitoring agent) or within an individual agent wishing to monitor the expectations it has of other agents. A prototype implementation of the compliance testing procedure has been implemented using SWI Prolog. Most of the features described here have been implemented, although currently the system is not connected to an agent—it uses a static database of states and their

Case	Result
$i<0 \vee i>n$	ϕ
ϕ is atomic	true if $s_i \models \phi$ false if $s_i \not\models \phi$ ϕ otherwise
$\phi = \neg\phi_1$	*simplify*$(\neg \ulcorner$*peval*$(\phi_1)\urcorner)$
$\phi = \phi_1 \wedge \phi_2$	*simplify*$(\ulcorner$*peval*$(\phi_1)\urcorner \wedge \ulcorner$*peval*$(\phi_2)\urcorner)$
$\phi = \exists x.(\phi_x \wedge \psi_x)$	false if *match*(ϕ_x, h, i) fails *simplify*$(\bigvee_{\psi\in\Psi}\psi)$ if $\Psi = \{\psi_x\sigma \mid \sigma =$ *match*$(\phi_x, h, i) \wedge$ *free_variables*$(\psi_x\sigma) = \varnothing\}$ is non-empty ϕ otherwise
$\phi = \mathsf{X}^+\phi_1$	true if *peval*$(\phi_1, h, i{+}1, dph) =$ true false if *peval*$(\phi_1, h, i{+}1, dph) =$ false $\mathsf{X}^+\ulcorner$*peval*$(\phi_1, h, i{+}1, dph)\urcorner$ otherwise
$\phi = \mathsf{X}^-\phi_1$	*Mirror image of* X^+ *case — omitted due to lack of space*
$\phi = \phi_1 \mathsf{U}_I^+ \phi_2$	true if $t_i \in I \wedge$ *peval*$(\phi_2, h, i, dph) =$ true false if $t_i > I \vee$ (*peval*$(\phi_1, h, i, dph) =$ false $\wedge$ *peval*$(\phi_2, h, i, dph) =$ false) $\mathsf{X}^+(\phi_1 \mathsf{U}^+_{arb(I,-\ulcorner dph\urcorner)}\phi_2)$ if $(t_i < I \vee t_i \in I) \wedge$ *peval*$(\phi_1, h, i, dph) =$ true ϕ otherwise
$\phi = \phi_1 \mathsf{U}_I^- \phi_2$	*Mirror image of* U_I^+ *case, except* *arb*$(I, -\ulcorner dph\urcorner)$ becomes $\ulcorner$*arb*$(I, t_i - t_{i-1})\urcorner$
$\phi = \downarrow^u x.\phi_x$	*peval*$(\phi_x[\llcorner$ *floor*$(t_i, u^M)\lrcorner/x], h, i, dph)$
$\phi = I$ (an interval formula)	true if $t_i \in I$ false otherwise

Figure 5: Function *peval*(ϕ, h, i, dph), where $h = \langle (s_0, t_0), \ldots, (s_n, t_n)\rangle$

facts—and integers are used to represent times rather than a specialised date/time language.

Future work includes investigating the expressivity of the rule language for modelling complex scenarios, enhancing it with additional operators (e.g. bounded universal quantification) and applying it to interaction protocol verification—where the X^+ operator will have particular relevance. It is also intended to deploy and evaluate this approach in a distributed multi-agent application.

Acknowledgements

Thanks to Hans van Ditmarsch and Willem Labuschagne for their comments on a draft of this paper.

References

[Alberti *et al.*, 2004] M. Alberti, F. Chesani, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. Compliance verification of agent interaction: a logic-based software tool. In R. Trappl, editor, *Cybernetics and Systems 2004*, volume II, pages 570–575. Austrian Society for Cybernetics Studies, 2004.

[Alur *et al.*, 1996] R. Alur, T. Feder, and T. A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM*, 43(1):116–146, 1996.

[Bacchus and Kabanza, 1998] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence*, 22(1-2):5–27, 1998.

[Bacchus and Kabanza, 2000] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2):123–191, 2000.

[Blackburn *et al.*, 2001] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, chapter 7, pages 436–447. Cambridge University Press, 2001.

[Cliffe and Padget, 2002] O. Cliffe and J. Padget. A framework for checking agent interaction within institutions. In MoChArt [2002].

[Cortés, 2004] U. Cortés. Electronic institutions and agents. *AgentLink News*, 15:14–15, September 2004.

[Haslum, 2002] P. Haslum. Partial state progression: An extension to the Bacchus-Kabanza algorithm, with applications to prediction and MITL consistency. In *Proceedings*

of the Workshop on Planning via Model-Checking, Sixth International Conference on AI Planning and Scheduling, pages 64–71, 2002.

[Huget *et al.*, 2002] M.-P. Huget, M. Esteva, S. Phelps, C. Sierra, and M. Wooldridge. Model checking electronic institutions. In MoChArt [2002].

[Leite *et al.*, 2004] J. A. Leite, A. Omicini, P. Torroni, and P. Yolum, editors. *Proceedings of the Workshop on Declarative Agent Languages and Technologies (DALT 2004), Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[Mallya *et al.*, 2004] A. U. Mallya, P. Yolum, and M. P. Singh. Resolving commitments among autonomous agents. In *Advances in Agent Communication*, volume 2922 of *Lecture Notes in Computer Science*, pages 166–182. Springer, 2004.

[MoChArt, 2002] *Proceedings of the Workshop on Model Checking and Artificial Intelligence (MoChArt-2002), 15th European Conference on Artificial Intelligence*, 2002.

[Verdicchio and Colombetti, 2003] M. Verdicchio and M. Colombetti. A logical model of social commitment for agent communication. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2003)*, pages 528–535. ACM Press, 2003.

[Verdicchio and Colombetti, 2004a] M. Verdicchio and M. Colombetti. A logical model of social commitment for agent communication. In F. Dignum, editor, *Advances in Agent Communication, International Workshop on Agent Communication Languages, ACL 2003*, volume 2922 of *Lecture Notes in Computer Science*, pages 128–145. Springer, 2004.

[Verdicchio and Colombetti, 2004b] M. Verdicchio and M. Colombetti. Dealing with time in content language expressions. In *Proceedings of the Workshop on Agent Communication, Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 90–104, 2004.