An Experiment in Formalizing Commitments
Using Action Languages

Tran Cao Son and Enrico Pontelli
Computer Science Department
New Mexico State University
tson|epontelli@cs.nmsu.edu

Chiaki Sakama
Computer and Communication Sciences
Wakayama University
sakama@sys.wakayama-u.ac.jp

Abstract
This paper investigates the use of high-level action languages for representing and reasoning about commitments in multiagent domains. The paper introduces the language $L^{mt}$ with features motivated by the problem of representing commitments; in particular, it shows how $L^{mt}$ can handle both simple commitment actions and complex commitment protocols. The semantics of $L^{mt}$ provides a uniform solution to different problems in reasoning about commitments, e.g., the problem of (i) verifying whether an agent fails (succeeds) to deliver on its commitments; (ii) identifying pending commitments; and (iii) suggesting ways to satisfy pending commitments.

The Language $L^{mt}$
Action languages provide a viable alternative for the description of agents and their capabilities, and to address various forms of reasoning about actions and change. The paper introduces the language $L^{mt}$ with features motivated by the problem of representing commitments; in particular, it shows how $L^{mt}$ can handle both simple commitment actions and complex commitment protocols.

The signature of $L^{mt}$ is $\langle AG, \{F_i, A_i\}_{i \in AG} \rangle$, where $AG$ is a finite set of agent identifiers and $F_i$ and $A_i$ are the sets of fluents and the set of actions of the agent $i$. We assume that $A_i \cap A_j = \emptyset$ for any distinct $i, j \in AG$. $A$ and $F$ denote the union of all the $A_i$ and $F_i$. A fluent literal is a fluent or a fluent preceded by $\neg$. Given a literal $\ell$, we denote with $\hat{\ell}$ its complement. A fluent formula is a propositional formula constructed from fluent literals. To handle time, we introduce a countable set $P$ of process names.

An annotated literal is a formula of the form $\ell t$, where $\ell$ is a fluent literal and $t > 0$ is an integer, representing a future point in time. We allow annotations of the form $\ell^{[t_1, t_2]}$, denoting $\ell^{t_1} \lor \cdots \lor \ell^{t_2}$ for $t_1 \leq t_2$. Annotated formulae are propositional formulae that use annotated literals. Given a fluent formula $\varphi$ (i.e., where fluents are not annotated), $\varphi^t$ ($\varphi^{[t_1, t_2]}$) is obtained by replacing each literal $\ell$ in $\varphi$ with the annotated literal $\ell^t$ ($\ell^{[t_1, t_2]}$). For a formula $\varphi$, $\varphi^{-t}$ is obtained by replacing each $\ell^t$ in $\varphi$ with $\ell^{t-1}$. An $L^{mt}$ domain specification contains axioms of the form:

- $\ell \text{ causes } \psi$ if $\psi$ (1)
- $\ell \text{ if } \psi$ if $\psi'$ (2)
- $\text{impossible } A$ if $\psi$ (3) initially $\ell$ (4)

Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

\begin{align*}
\psi \text{ starts } & \text{process_id} \{\text{reversible/irreversible}\} \ell^t & (5) \\
\psi \text{ stops } & \text{process_id} & (6) \\
a \text{ starts } & \text{process_id} \{\text{reversible/irreversible}\} \ell^t \text{ if } \psi & (7) \\
a \text{ stops } & \text{process_id} \text{ if } \psi & (8)
\end{align*}

where $a \in \bigcup_{i \in AG} A_i$ is an action, $\ell$ is a fluent literal, $\psi$ and $\psi'$ are sets of fluent literals (interpreted as conjunctions), $A \subseteq \bigcup_{i \in AG} A_i$ is a set of actions, $\ell^t$ is a time annotated literal, of the form $\bigvee_{[t_1, t_2]}$ with $1 \leq t_1 \leq t_2$, and $\text{process_id}$ belongs to $P$. A dynamic law (axiom (1)) describes the direct effects of execution of one action, static laws (axiom (2)) describe integrity constraints on states of the world, and non-executability laws (axiom (3)) describe conditions that prevent the (concurrent) execution of groups of actions. Statements of type (4) are employed to describe the initial state of the world. A process is associate to a delayed effect, denoted by $\ell^t$, and the time interval $t$ indicates when the process will produce its effect. A process can be started by an action or a property. Each reversible process can be interrupted by a stops action/condition before materializing its effects, while irreversible processes cannot be stopped. We denote with $D$ a generic domain specification.

We define an action snapshot as a set $\{a_i\}_{i \in AG}$ where $a_i \in A_i \cup \{\text{noop}\}$. An interpretation $I$ is a maximal consistent set of fluent literals drawn from $F$ (i.e., each fluent appears positive or negated). A fluent $\ell$ is said to be true (resp. false) in $I$ iff $\ell \in I$ (resp. $\ell \notin I$). The truth value of a fluent formula in $I$ is defined recursively over the propositional connectives in the usual way. We say that $I$ satisfies $\varphi$ ($I \models \varphi$) if $\varphi$ is true in $I$. A set of literals $S$ is closed under $D$ if for every law $(\psi \text{ if } \psi')$ in $D$, if $S \models \psi$ then $S \models \psi'$. $Cl_D(S)$ denotes the smallest superset of $S$ which is closed under $D$. A state is an interpretation that is closed under $D$.

A set of actions $B$ is prohibited in a state $s$ if there is a non-executability law (type (3)) in $D$ s.t. $A \subseteq B$ and $s \models \psi$. The effect of an action $a$ in a state $s$ is the set $e_A(s) = \{\ell \mid D$ contains a law a causes $\ell$ if $\psi$, $a \in A$, and $s \models \psi\}$.

To represent processes, we introduce the notion of an extended state as a triple $(s, IR, RE)$ where $s$ is a state and $IR$ and $RE$ are sets of pairs of future effects—each future effect has the form $(x : \ell^t)$, where $x$ is a process name and $\ell^t$ is an annotated fluent. $s$ encodes the current state of the world.

\begin{footnote}
Note that a law with the annotation $\land [t_1, t_2]$ can be replaced by a set of laws whose annotation is $\bigvee_{[t_1, t_2]}$.
\end{footnote}
while $IR$ and $RE$ contain the irreversible and reversible processes, respectively. $(s, IR, RE)$ is complete if $IR = \emptyset$ and $RE = \emptyset$.

The semantics of the language is specified by a transition function $\Phi_D^t$, which maps each extended state and action snapshot to a set of extended states. The definition of $\Phi_D^t$ is a generalization of the transition function developed in (Baral, Son, and Tuan 2002). Since this is not the main purpose of this paper, to save space, we omit the precise definition of $\Phi_D^t$. The interested readers can find it in (Son, Pontelli, and Sakama 2010). For our purpose, it suffices to note that $\Phi_D^t(s, A)$ is a set of extended states if $A$ is not prohibited in $s$, and $\Phi_D^t(s, A) = \emptyset$ otherwise. Intuitively, $\Phi_D^t(s, A)$ encodes the possible trajectories of the world given that $A$ is executed in $s$. It is possible to extend $\Phi_D^t$ to $\Phi_D^{\infty}$ which operates on sequences of action snapshots in a straightforward manner. Also, $\Phi_D^t(s, A, t)$ denotes the set of extended states resulting from executing $A$ at $t$ units of time from $s$, and $\Phi_D^t(s, A, t) + t_1$ denotes the set of extended states resulting from allowing $t_1$ units of time to elapse after $\Phi_D^t(s, A, t)$.

A timed action snapshot is a pair $(A, t)$ where $A$ is an action snapshot and $t$ is a time reference. The function $\Phi_D^t$ is extended to operate on sequences of timed action snapshots $\alpha = [(A_1, t_1), \ldots, (A_n, t_n)]$, denoted by $\Phi_D^t(\alpha, \alpha')$.

An initial state is a complete state $(s_0, \emptyset, \emptyset)$ such that, for each statement of type (4) in $D$ we have that $s_0 \models \emptyset$. We will assume from now on that there exists at least one initial state. A trajectory is a sequence $s_0, s_1, \ldots, s_n$ such that each $s_i$ is a timed action snapshot, $s_0$ is an initial state, and $s_i = \Phi_D^t(s_{i-1}, [\beta_0, \ldots, \beta_{i-1}])$ for $0 \leq i < n$.

We allow queries of the form: $\psi$ after $\alpha$, where $\alpha$ is a sequence of timed action snapshots. A query $q$ is true w.r.t. an initial state $s_0$, denoted $s_0 \models q$, if $\Phi_D^t(s_0, \alpha) \neq \emptyset$ and $\forall s \in \Phi_D^t(s_0, \alpha): s \models \psi$. A query $q$ is entailed by $D$ $(D \models q)$ if for each initial state $s_0$ of $D$ we have $s_0 \models q$.  

<table>
<thead>
<tr>
<th>Customer</th>
<th>Merchant</th>
</tr>
</thead>
<tbody>
<tr>
<td>request causes request</td>
<td>request causes goods</td>
</tr>
<tr>
<td>accept causes accept</td>
<td>goods causes receipt</td>
</tr>
<tr>
<td>pmt stops pmt</td>
<td>quote causes quote</td>
</tr>
<tr>
<td>reversible pay</td>
<td>impossible {quote} if $\neg$quote</td>
</tr>
<tr>
<td>cancelP stops pmt</td>
<td>impossible {cancelP} if $\neg$pay</td>
</tr>
</tbody>
</table>

Example 1 Let us consider a modification of the Netbill example (Mallya and Huhns 2003). Each action takes one day to complete but the action of sending the payment might take 3 to 5 days. As long as the payment has not been made, the customer can cancel the payment. We envision $A_G = \{m, c\}$. Both the (m)erchant and the (c)ustomer use the fluents $F = \{request, pay, goods, receipt, quote, accept\}$; the agents use the actions: $A_m = \{sQuote, sGoods, sReceipt\}$ and $A_c = \{sRequest, sAccept, sPayment, cancelP\}$. The domain consists of the axioms in the table above $(P = \{pmt\})$. Let $s_0 = \{request, quote, accept, \neg$pay, $\neg$receipt, goods$\}$. Let $\hat{u} = (s'_0, 0, \emptyset)$ and $\alpha_2 = \{sPayment, noop\}$. Then, $\Phi_D^t(\hat{u}, \alpha_2) = \{update(s'_0, \emptyset, \{pmt : pay\})\}$. Thus, $\Phi_D^t(\hat{u}, \alpha_2) + 3 = \{(u'_0, 0, 0)\}$ and $\{update(s'_0, 0, \{pmt : pay\})\}$ if $i = 1, 2$. We can see that $\Phi_D^t(\hat{u}, \alpha_2) + 5 = \{(u'_0, 0, 0)\}$.

Basic Commitments in $L^{mnt}$

Commitments are encoded in $L^{mnt}$ as a new class of fluents and are manipulated by commitment actions. Due to the lack of space, we present our study on unconditional commitments (Singh 1999). A commitment is of the form $c(x, y, \varphi, t_1, t_2)$, where $x, y \in A_G$, $0 < t_1 \leq t_2$, and $\varphi$ is formula. This states that the debtor $x$ agrees to establish $\varphi$ between $t_1$ and $t_2$ for the creditor $y$. For example, the statement “A commits to visit B in three hours,” conveys the commitment $c(A, B, arrived, 3, 3)$. A commitment where we do not care when the property is made true can be expressed using a disjunctive annotation.

Observe that we can think of commitment fluents as propositions, i.e., $c(x, y, \varphi)$ is a syntactic sugar for $c_{x,y}\text{name}(\varphi)$ where name$(\varphi)$ is a propositional variable representing the “name” of the formula $\varphi$. We assume that the various propositions $c(x, y, \varphi)$ are in $F$. We also assume that, if $c(x, y, \varphi)$ is a commitment fluent, then $\varphi$ is a fluent formula which uses fluents from $F_x \cup F_y$.

The following operations manipulate commitments:

- $\text{create}(x, y, \varphi, t_1, t_2)$: agent $x$ creates a commitment to $\varphi$ in the period between $t_1$ and $t_2$. We assume that each created commitment is associated to a unique identifier;
- $\text{discharge}(x, y, \varphi)$: agent $x$ discharges a commitment towards agent $y$ (by satisfying the request);
- $\text{release}(x, y, \varphi)$: agent $y$ releases $x$ from its obligation;
- $\text{assign}(x, y, k, \varphi, t_1, t_2)$: agent $y$ transfers the commitment to a different creditor (with a new time frame);
- $\text{delegate}(x, y, k, \varphi, t_1, t_2)$: agent $x$ delegates the commitment to another debtor (with a new time frame);
- $\text{cancel}(x, y, \varphi, \psi, t_1, t_2)$: $x$ modifies the terms of the commitment.

These manipulations of commitments are the consequence of actions performed by the agents or conditions occurring in the state of the world. We consider two types of commitment manipulation enabling statements (trigger statements):

- $\varphi$ triggers $c$.activity $a$ triggers $c$.activity

where $\varphi$ is a fluent formula, $a \in A$, and $c$.activity is one of the commitment actions. They indicate that the commitment activity $c$.activity should be executed whenever $\varphi$ holds or $a$ is executed. An example of the first type of statement is

- $\text{pay triggers create}(m, c, \text{receipt}, 1, 3)$

indicating that the merchant agrees to send a receipt between 1 and 3 units of time after the payment. The statement

- $\text{accept triggers create}(m, c, \text{pay}, 1, 5)$

states that the customer agrees to pay for the goods between 1 to 5 units of time after sending the acceptance notification.

A domain with commitments is a pair $(D, C)$ where $D$ is a domain specification in $L^{mnt}$ and $C$ is a collection of trigger statements—representing (social or contractual) agreements between agents in the domain. We define the semantics of a domain with commitments $(D, C)$ by translating it into a
The set of fluents in \(\tau(M_1)\), denoted by \(F_1\), consists of \(F\), the commitment fluents such as \(c(m,c,quote)\), \(c(c, m, pay)\), \(c(m, c, quote)\), and \(c(m, c, goods)\), and fluents of the form \(done(x, y, \varphi)\). Let \(s_0 = \{f | f \in F_1\}\), we have that \( \Phi_{\tau(M_1)}^e(s_0, \{sendR\}) = \{[s_0, u, v]\} \) where \(u = s_0 \setminus \{\text{request}, c(m, c, quote)\} \cup \{\text{request}, c(m, c, quote)\} \cup \{\text{done}(m, c, quote)\} \) and \(v = u \setminus \{\text{done}(m, c, quote)\} \cup \{\text{done}(m, c, quote)\} \). The presence of \(c(m, c, quote)\) and \(\text{done}(m, c, quote)\) in \(u\) and \(v\) is due to the laws \(c(x, y, \varphi)\) if \(\text{request}\) and \(\text{request}\) \(c(x, y, \varphi)\) \(\text{reversible}\) \(\text{done}(x, y, \varphi)\) respectively, both are the result of the translation to laws in \(\tau(M_1)\) of the statement request triggers \(c(m, c, quote)\) and \(\text{done}(m, c, quote)\) in \(\tau(M_1)\).

Let \(M = (D, C)\) be a domain with commitments and \(\gamma = [s_0, \ldots, s_n]\) be a sequence of states in \(\tau(M)\). A commitment fluent \(c(x, y, \varphi)\) appearing in \(\gamma\) is

- **Satisfied** in \(\gamma\) if \(s_n \models \neg c(x, y, \varphi)\);
- **Violated** in \(\gamma\) if \(s_n \models c(x, y, \varphi) \land \text{done}(x, y, \varphi)\);
- **Pending** in \(\gamma\) if \(s_n \models c(x, y, \varphi)\) and \(s_n \not\models \text{done}(x, y, \varphi)\).

The reasoning about commitments given the execution of a sequence of action snapshots can then be defined as follows. Let \(M = (D, C)\) be a domain with commitments, \(s_0\) be a state in \(D\), and \(A = \{a_1, t_1, \ldots, a_n, t_n\}\) be a sequence of timed action snapshots. A commitment \(c(x, y, \varphi)\) is factual during the execution of \(A\) in \(s\) if there is a sequence of states \(\gamma = [s_0, \ldots, s_m]\) in \(\Phi_{\tau(M)}^e(s_0, A)\) and \(c(x, y, \varphi)\) appears in \(\gamma\). A factual commitment \(c(x, y, \varphi)\) is

- **Satisfied** after the execution of \(A\) in \(s_0\) if it is satisfied in every sequence of states belonging to \(\Phi_{\tau(M)}^e(s_0, A)\).
- **Strongly (weakly) violated** after the execution of \(A\) in \(s_0\) if it is violated in every (some) sequence of states belonging to \(\Phi_{\tau(M)}^e(s_0, A)\).
- **Pending** after the execution of \(A\) in \(s_0\) if it is not violated in any sequence of states and not satisfied in some sequences of states belonging to \(\Phi_{\tau(M)}^e(s_0, A)\).

**Example 3** \(c(m,c,quote)\) is violated after the execution of \(s\text{Request}\) at \(s_0\) in Ex. 2, since \(\Phi_{\tau(M)}^e(s_0, \{\text{sendR}\}) = \{[s_0, u, v]\}\). We can check that for \(A = \{(s\text{Request}), v', 1\}\), \(\Phi_{\tau(M)}^e(s_0, A) = \{[s_0, u, v',] \text{done}(m, c, quote)\} \cup \{c(x, y, \varphi)\} \cup \{\text{done}(m, c, quote)\} \cup \{c(x, y, \varphi)\}\), i.e., \(c(m,c,quote)\) is satisfied after \(A\)'s execution in \(s_0\).

**Observations and Narratives**

**Observation Language**

We extend the action language by introducing **observations**—by adding to the signature of the language \(L^m\) a set of **situation constants** \(S\), containing two special constants, \(s_0\) and \(s_c\), denoting the initial situation and the current situation. Observations are axioms of the forms:

\[
\begin{align*}
\varphi \text{ at } s & \quad (11) \\
\varphi \text{ at } t & \quad (12) \\
\alpha \text{ occurs at } s & \quad (13) \\
\alpha \text{ between } s_1, s_2 & \quad (14) \\
s_1 < s_2 & \quad (15)
\end{align*}
\]

where \(\varphi\) is a fluent formula, \(\alpha\) is a sequence of timed action snapshots, and \(s, s_1, s_2\) are situation constants which differ from \(s_c\). (11) (fluent fact) states that \(\varphi\) is true in the situation...
s. (15) (precedence fact) says that \( s_1 \) occurs before \( s_2 \). Axioms of the forms (14) and (12) are referred to as occurrence facts. (12) indicates that \( \alpha \) starts its execution in the situation \( s \). On the other hand, (14) states that \( \alpha \) starts and completes its execution in \( s_1 \) and \( s_2 \), respectively. Axioms of the form (13) link situations to time points.

A narrative of a multi-agent system is a pair \((D, \Gamma)\) where \( D \) is a domain description and \( \Gamma \) is a set of observations of the form \((11)-(15)\) such that \( \{s_0 < s, s < s_2 | s \in S\} \subseteq \Gamma \).

The notion of an interpretation and a model for narratives of single-agent domains (Baral, McIlraith, and Son 2000), can be extended to define models of a narrative of multi-agent domain—see (Son, Pontelli, and Sakama 2010).

We can also envision an extension of the query language by allowing queries of the form \( \forall s, t \), where the testing of the entailment starts from the states in \( \Psi(\Sigma(s)) \). Given a narrative \((D, \Gamma)\) and a fluent formula \( \varphi \), we are also interested in knowing whether \( \varphi^\ast \) is true in a situation \( s \) for some \( t_1 \leq t \leq t_2 \). This is expressed using a query of the form \( \varphi^\ast[t_1,t_2] \) at \( s \); a query \( q \) of this type holds w.r.t. \((D, \Gamma)\) (i.e., \((D, \Gamma) \models q\)) if, for every model \( M = (\Psi, \Sigma, \Delta) \) of \((D, \Gamma)\), there exists \( t_1 \leq t \leq t_2 \) s.t. \( \varphi \) is true in \( \Psi(\Sigma(s)) + t \).

Narratives and Commitments

A narrative with commitments is a triple \((D, \Gamma, C)\) where \((D, C)\) is a domain with commitments and \( \Gamma \) is a collection of observations. The semantics of a narrative with commitments \((D, \Gamma, C)\) is defined by (i) translating it to the narrative \((\tau(M), \Gamma)\) in \( L^mt \) where \( M = (D, C) \); and (ii) specifying models of \((\tau(M), \Gamma)\) to be models of \((D, \Gamma, C)\). We omit the specific details on the semantics of narratives with commitments. Let \( N = (D, \Gamma, C) \) be a narrative and \( M \) be a model of \( N \). We say that a commitment \( c(x, y, \varphi) \) is:

- **Satisfied by \( M \)** if \( M \models \neg c(x, y, \varphi) \) at \( s_c \).
- **Violated by \( M \)** if \( M \models (\text{done}(c, y, \varphi) \land c(x, y, \varphi)) \) at \( s_c \).
- **Pending w.r.t. \( M \)** if \( M \models \neg \text{done}(c, y, \varphi) \land c(x, y, \varphi) \) at \( s_c \).

Given a narrative \( N \), we will say that a commitment is satisfied if it is satisfied in all models of \( N \); it is strongly violated if it is violated in all models of \( N \); and it is weakly violated if it is violated in some models of \( N \).

**Example 4** Consider the narrative \( N_1 = (D_n, \Gamma, C_2) \) where \( M_1 = (D_n, C_2) \) is the domain description in Ex. 2 and \( \Gamma \) consists of the precedence facts \( s_0 < s_1 < s_2 < s_3 \) and the following observations:

\[
\text{\texttt{sRequest occurs at } s_0 \text{ and } \texttt{sAccept occurs at } s_2}
\]

where \( s_0, s_1, s_2, s_3, s_c \) are situation constants. A model \( M = (\Psi, \Sigma, \Delta) \) for this narrative can be built as follows:

- The sequences of actions leading to the various situations are \( \Sigma(s_0) = [\text{\texttt{sRequest}}], \Sigma(s_1) = [\text{\texttt{sRequest}}, \text{\texttt{sQuote}}], \) and \( \Sigma(s_3) = [\text{\texttt{sRequest}}, \text{\texttt{sQuote}}, \text{\texttt{sAccept}}] \).

- \( \Psi(\Gamma) \) is the state where all fluents are false and \( \Psi(s) = \Phi^\ast[M_1, \Sigma(s_c), \Psi(\Gamma)] \).

- The time assignment for situation constants is given by \( \Delta(s_1) = \{i \mid i \geq 0\} \) and \( \Delta(s_2) = \{i \mid i \geq 0\} \).

We can prove that, for every model \( (\Psi', \Sigma', \Delta') \) of \( M_1 \), the situation assignment \( \Sigma' \) is identical to \( \Sigma \) and \( \Psi'([\Gamma]) \) must satisfy \( \{\neg \text{\texttt{pay}}, \neg \text{\texttt{accept}}, \neg \text{\texttt{quote}}, \neg \text{\texttt{goods}}\} \). This allows us to conclude that \( N_1 \models (\neg \text{\texttt{pay at } s}) \) for \( s \in S \) and \( N_1 \models (\text{\texttt{c(s, m, pay)}} \land \neg \text{\texttt{done(c, m, pay)}}) \) at \( s_c \). Here, \( \text{\texttt{c(m, c, quote)}} \) is satisfied and \( \text{\texttt{c(m, c, pay)}} \) is pending.

Complex Commitments and Protocols

A basic commitment represents a promise made by an agent to another one, but without specifying a precise procedure to accomplish the commitment. Basic commitments also do not describe complex dependencies among “promises”. To represent such dependencies, protocols can be used. A protocol is a program constructed from action occurrences, formula, and the usual program constructs (e.g., if-then-else, while) or an ordering. Intuitively, a protocol specifies the behavioral norm for agents to act to fulfill their commitments.

The language for specification of narratives in the previous section can be extended to allow statements that trigger complex commitments and observations of protocol occurrences (Son, Pontelli, and Sakama 2010).

Discussion and Conclusion

In this paper, we showed how various problems in reasoning about commitments can be described by a suitable instantiation of commitment actions in the language \( L^mt \). In particular, we showed how the problem of verifying commitments or identifying pending commitments can be posed as queries to a narrative with commitments and how the language can be extended to consider commitment protocols.

Since our framework provides a way to identify pending, violated, and satisfiable commitments given a narrative \((D, \Gamma, C)\), a natural question is what should the agents do to satisfy the pending commitments. The semantics of domains with commitments suggests that we can view the problem of identifying a possible course of actions for the agents to satisfy the pending commitments as an instance of the planning problem, to be solved by planning techniques. This is one of our main goals in this research in the near future.

References


