An Experiment in Formalizing Commitments Using Action Languages

Tran Cao Son and **Enrico Pontelli**
Computer Science Department
New Mexico State University
tson|epontell@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 multi-agent domains. The paper introduces the language $\mathcal{L}^{mt}$ with features motivated by the problem of representing commitments; in particular, it shows how $\mathcal{L}^{mt}$ can handle both simple commitment actions and complex commitment protocols. The semantics of $\mathcal{L}^{mt}$ provides a uniform solution to different problems in reasoning about commitments, e.g., the problem of (*i*) verifying whether an agent fails (or succeeds) to deliver on its commitments; (*ii*) identifying pending commitments; and (*iii*) suggesting ways to satisfy pending commitments.

The Language $\mathcal{L}^{mt}$

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 $\mathcal{L}^{mt}$ with features motivated by the problem of representing commitments; in particular, it shows how $\mathcal{L}^{mt}$ can handle both simple commitment actions and complex commitment protocols.

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

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

An $\mathcal{L}^{mt}$ domain specification contains axioms of the form:

a **causes** ℓ **if** ψ (1)
ψ **if** ψ' (2)
impossible A **if** ψ (3)
initially ℓ (4)
ψ **starts** $process\_id$ [**reversible**|**irreversible**] $\ell^{\hat{t}}$ (5)
ψ **stops** $process\_id$ (6)
a **starts** $process\_id$ [**reversible**|**irreversible**] $\ell^{\hat{t}}$ **if** ψ (7)
a **stops** $process\_id$ **if** ψ (8)

where $a \in \bigcup_{i\in\mathcal{AG}} \mathcal{A}_i$ is an action, ℓ is a fluent literal, ψ and ψ' are sets of fluent literals (interpreted as conjunctions), $A \subseteq \bigcup_{i\in\mathcal{AG}} \mathcal{A}_i$ is a set of actions, $\ell^{\hat{t}}$ is a time annotated literal, of the form $\vee[t_1, t_2]$ with $1 \leq t_1 \leq t_2$,[1] and $process\_id$ belongs to $\mathcal{P}$. 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 $\ell^{\hat{t}}$, and the time interval $\hat{t}$ 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 $\{a_i\}_{i\in\mathcal{AG}}$ where $a_i \in \mathcal{A}_i \cup \{\texttt{noop}\}$. An *interpretation* I is a maximal consistent set of fluent literals drawn from $\mathcal{F}$ (i.e., each fluent appears positive or negated). A fluent f is said to be true (resp. false) in I iff $f \in I$ (resp. $\neg f \in I$). 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 φ ($I \models \varphi$) if φ is true in I. A *set of literals* S is closed under D if for every law (ψ **if** ψ') 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** ℓ **if** ψ, $a \in A$, 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 $(x : \ell^{\hat{t}})$, where x is a process name and $\ell^{\hat{t}}$ is an annotated fluent. s encodes the *current* state of the world,

[1] Note that a law with the annotation $\wedge[t_1, t_2]$ can be replaced by a set of laws whose annotation is $\vee[t_i, t_i]$ for $t_1 \leq t_i \leq t_2$.

while IR and RE contain the irreversible and reversible processes, respectively. (s, IR, RE) is *complete* if $IR = \emptyset$ and $ER = \emptyset$.

The semantics of the language is specified by a *transition function* Φ^t_D which maps each extended state and action snapshot to a set of extended states. The definition of Φ^t_D 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 Φ^t_D. The interested readers can find it in (Son, Pontelli, and Sakama 2010). For our purpose, it suffices to note that $\Phi^t_D(\hat{s}, A)$ is a set of extended states if A is not prohibited in s, and $\Phi^t_D(\hat{s}, A) = \emptyset$ otherwise. Intuitively, $\Phi^t_D(\hat{s}, A)$ encodes the possible trajectories of the world given that A is executed in $\hat{s}$. It is possible to extend Φ^t_D to $\widehat{\Phi^t_D}$ which operates on *sequences* of action snapshots in a straightforward manner. Also, $\Phi^t_D(\hat{s}, A, t)$ denotes the set of extended states resulting from executing A at t units of time from $\hat{s}$, and $\Phi^t_D(\hat{s}, A, t) + t_1$ denotes the set of extended states resulting from allowing t_1 units of time to elapse after $\Phi^t_D(\hat{s}, A, t)$.

A *timed action snapshot* is a pair (A, t) where A is an action snapshot and t is a time reference. The function $\widehat{\Phi^t_D}$ is extended to operate on sequences of timed action snapshots $\alpha = [(A_1, t_1), \ldots, (A_n, t_n)]$, denoted by $\widehat{\Phi^t_D}(\hat{s}, \alpha)$.

An *initial state* is a complete state $(s_0, \emptyset, \emptyset)$ such that, for each statement of type (4) in D we have that $s_0 \models \ell$. We will assume from now on that there exists at least one initial state. A trajectory is a sequence $\hat{s}_0\beta_0\hat{s}_1\beta_1 \ldots \beta_{n-1}\hat{s}_n$ such that each β_j is a timed action snapshot, $\hat{s}_0$ is an initial state, and $\hat{s}_i \in \widehat{\Phi^t_D}(\hat{s}_0, [\beta_0, \ldots, \beta_{i-1}])$ for $0 \leq i < n$.

We allow *queries* of the form: ψ **after** α, where α is a sequence of timed action snapshots. A query q is true w.r.t. an initial state $\hat{s}_0$, denoted $\hat{s}_0 \models q$, if $\widehat{\Phi^t_D}(\hat{s}_0, \alpha) \neq \emptyset$ and $\forall \hat{s} \in \widehat{\Phi^t_D}(\hat{s}_0, \alpha) : \hat{s} \models \psi$. A query q is entailed by D ($D \models q$) if for each initial state $\hat{s}_0$ of D we have $\hat{s}_0 \models q$.

Customer	Merchant
sRequest **causes** $request$	*sGoods* **causes** $goods$
sAccept **causes** $accept$	*sReceipt* **causes** $receipt$
sPayment **starts** pmt	*sQuote* **causes** $quote$
reversible $pay^{\vee[3,5]}$	**impossible** $\{sGoods\}$
cancelP **stops** pmt	**if** $\neg accept$
impossible $\{sAccept\}$ **if** $\neg quote$	**impossible** $\{sReceipt\}$
impossible $\{cancelP\}$ **if** pay	**if** $\neg pay$

Example 1 *Let us consider a modification of the 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* $\mathcal{AG} = \{m, c\}$. *Both the (m)erchant and the (c)ustomer use the fluents* $\mathcal{F} = \{request, pay, goods, receipt, quote, accept\}$*; the agents use the actions:* $\mathcal{A}_m = \{sQuote, sGoods, sReceipt\}$ *and* $\mathcal{A}_c = \{sRequest, sAccept, sPayment, cancelP\}$. *The domain consists of the axioms in the table above* ($\mathcal{P} = \{pmt\}$). *Let* $s_0 = \{request, quote, accept, \neg pay, \neg receipt, \neg goods\}$, *and* $\alpha_1 = \{noop, sGoods\}$. α_1 *is executable in* s_0 *and* $\Phi^t_{D_n}((s_0, \emptyset, \emptyset), \alpha_1) = \{(s'_0, \emptyset, \emptyset)\}$, *where* $s'_0 = \{request, quote, accept, \neg pay, \neg receipt, goods\}$. *Let* $\hat{u} = (s'_0, \emptyset, \emptyset)$ *and* $\alpha_2 = \{sPayment, \texttt{noop}\}$. *Then,* $\Phi^t_D(\hat{u}, \alpha_2) = \{update((s'_0, \emptyset, \{(pmt : pay^i)\})) \mid i = 3, 4, 5\}$. *Thus,* $\Phi^t_D(\hat{u}, \alpha_2) + 3 = \{(u', \emptyset, \emptyset)\} \cup \{update((s'_0, \emptyset, \{(pmt : pay^i)\})) \mid i = 1, 2\}$ *where* $u' = \{request, quote, accept, pay, \neg receipt, goods\}$. *We can see that* $\Phi^t_D(\hat{u}, \alpha_2) + 5 = \{(u', \emptyset, \emptyset)\}$. □

Basic Commitments in $\mathcal{L}^{mt}$

Commitments are encoded in $\mathcal{L}^{mt}$ 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 $x, y \in \mathcal{AG}$, $0 < t_1 \leq t_2$, and φ is formula. This states that the debtor x agrees to establish φ 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 $c\_x\_y\_name(\varphi)$ where $name(\varphi)$ is a propositional variable representing the "name" of the formula φ. We assume that the various propositions $c(x, y, \varphi)$ are in $\mathcal{F}$. We also assume that, if $c(x, y, \varphi)$ is a commitment fluent, then φ is a fluent formula which uses fluents from $\mathcal{F}_x \cup \mathcal{F}_y$.

The following operations manipulate commitments:

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

φ **triggers** $c\_activity$ $\qquad$ a **triggers** $c\_activity$

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

$$pay \textbf{ triggers } create(m, c, receipt, 1, 3) \tag{9}$$

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

$$sAccept \textbf{ triggers } create(c, m, pay, 1, 5) \tag{10}$$

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 $\mathcal{L}^{mt}$ 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

$\mathcal{L}^{mt}$ domain D' where D' consists of D and a collection of dynamic and static laws originating from C. We describe here below the translation of the action triggers—the fluent triggers are similar and omitted due to lack of space. Let a **triggers** $c\_activity$ belongs to C. in this case,

- if $c\_activity = create(x, y, \varphi, t_1, t_2)$, then the laws a **causes** $c(x, y, \varphi)$ and a **starts** $c(x, y, \varphi)$ **reversible** $done(x, y, \varphi)^{\vee[t_1,t_2]}$ are added to D'. The dynamic law records the commitment $c(x, y, \varphi)$ created by action a. The second law starts a process indicating that $c(x, y, \varphi)$ must be satisfied between t_1 and t_2.
- if $c\_activity = discharge(x, y, \varphi)$ then D' contains a **stops** $c(x, y, \varphi)$ **if** $c(x, y, \varphi)$, a **causes** $\neg c(x, y, \varphi)$ **if** $c(x, y, \varphi)$, and a **starts** $discharging(x, y, \varphi)$ **irreversible** φ **if** $c(x, y, \varphi)$. Action a stops the process $c(x, y, \varphi)$ by starting a process achieving φ.
- if $c\_activity = release(x, y, \varphi)$ then D' contains a **stops** $c(x, y, \varphi)$ **if** $c(x, y, \varphi)$ and a **causes** $\neg c(x, y, \varphi)$ **if** $c(x, y, \varphi)$. The action stops the commitment process and records that the commitment has been removed.
- if $c\_activity = assign(x, y, k, \varphi, t_1, t_2)$ then D' contains a **stops** $c(x, y, \varphi)$ **if** $c(x, y, \varphi)$, a **causes** $\neg c(x, y, \varphi)$ **if** $c(x, y, \varphi)$, a **causes** $c(x, k, \varphi)$ and a **starts** $c(x, k, \varphi)$ **reversible** $done(x, k, \varphi)^{\vee[t_1,t_2]}$. The action stops and releases the commitment process $c(x, y, \varphi)$ and starts the process $c(x, k, \varphi)$.
- if $c\_activity = delegate(x, y, k, \varphi, t_1, t_2)$ then D' contains a **stops** $c(x, y, \varphi)$ **if** $c(x, y, \varphi)$, a **causes** $\neg c(x, y, \varphi)$ **if** $c(x, y, \varphi)$, a **causes** $c(k, y, \varphi)$, and a **starts** $c(k, y, \varphi)$ **reversible** $done(k, y, \varphi)^{\vee[t_1,t_2]}$. This is similar to the case of $release$, only with different debtor.
- if $c\_activity = cancel(x, y, \varphi, \psi, t_1, t_2)$ then D' contains a **stops** $c(x, y, \varphi)$ **if** $c(x, y, \varphi)$, a **causes** $\neg c(x, y, \varphi)$ **if** $c(x, y, \varphi)$, a **causes** $c(x, y, \psi)$, and a **starts** $c(x, y, \psi)$ **reversible** $done(x, y, \psi)^{\vee[t_1,t_2]}$.

We further need to include some additional static laws: if $c(x, y, \varphi)$ is present and φ is true, then the commitment can be released: $\neg c(x, y, \varphi)$ **if** $\varphi, done(x, y, \varphi)$.

Let $\mathcal{M} = (D, C)$ be a domain with commitments. We denote with $\tau(C)$ the collection of axioms generated from the previous translation process; with a slight abuse of notation, we denote $\tau(\mathcal{M}) = D \cup \tau(C)$. The domain $\tau(\mathcal{M})$ defines a transition function $\Phi^t_{\tau(\mathcal{M})}$ which determines the evolutions of the world given a state and a sequence of timed action snapshots. The function $\Phi^t_{\tau(\mathcal{M})}$ can be used to specify the transition function for $\mathcal{M}$, i.e., the transition function $\Phi_{\mathcal{M}}$ for $\mathcal{M}$ is defined to be the function $\Phi^t_{\tau(\mathcal{M})}$. The presence of $c(x, y, \varphi)$ in a state indicates that the commitment $c(x, y, \varphi)$ has been made. $done(x, y, \varphi)$ encodes the fact that the commitment $c(x, y, \varphi)$ needs to be realized by the debtor.

Example 2 *Consider the domain with commitments $\mathcal{M}_1 = (D_n, C_2)$, where D_n is the domain description described in Example 1 and C_2 is the set of statements consisting of (9), (10), and the following statements*

$$request \ \textbf{triggers} \ create(m, c, quote, 1, 1)$$
$$accept \ \textbf{triggers} \ create(m, c, goods, 1, 1).$$

*The set of fluents in $\tau(\mathcal{M}_1)$, denoted by $\mathcal{F}_1$, consists of $\mathcal{F}$, the commitment fluents such as $c(m, c, receipt)$, $c(c, m, pay)$, $c(m, c, quote)$, and $c(m, c, goods)$, and fluents of the form $done(x, y, \varphi)$. Let $s_0 = \{\neg f \mid f \in \mathcal{F}_1\}$, we have that $\Phi^t_{\tau(\mathcal{M}_1)}(s_0, \{sendR\}) = \{[s_0, u, v]\}$ where $u = s_0 \setminus \{\overline{request}, \overline{c(m, c, quote)}\} \cup \{request, c(m, c, quote)\}$ and $v = u \setminus \{\overline{done(m, c, quote)}\} \cup \{done(m, c, quote)\}$. The presence of $c(m, c, quote)$ and $done(m, c, quote)$ in u and v is due to the laws $c(x, y, quote)$ **if** $request$ and $request$ **starts** $c(x, y, quote)$ **reversible** $done(x, y, \varphi)^1$ respectively, both are the result of the translation to laws in $\tau(\mathcal{M}_1)$ of the statement $request$ **triggers** $create(m, c, quote, 1, 1)$.* □

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

- *Satisfied* in γ if $s_n \models \neg c(x, y, \varphi)$;
- *Violated* in γ if $s_n \models c(x, y, \varphi) \wedge done(x, y, \varphi)$; or
- *Pending* in γ if $s_n \models c(x, y, \varphi)$ and $s_n \not\models done(x, y, \varphi)$.

The reasoning about commitments given the execution of a sequence of action snapshots can then be defined as follows. Let $\mathcal{M} = (D, C)$ be a domain with commitments, s_0 be a state in D, and $A = [(\alpha_1, t_1), \ldots, (\alpha_n, t_n)]$ 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 $\widehat{\Phi^t_{\tau(\mathcal{M})}}(s_0, A)$ and $c(x, y, \varphi)$ appears in γ. 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 $\widehat{\Phi^t_{\tau(\mathcal{M})}}(s_0, A)$.
- *Strongly (weakly) violated* after the execution of A in s_0 if it is violated in every (some) sequence of states belonging to $\widehat{\Phi^t_{\tau(\mathcal{M})}}(s_0, A)$.
- *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 $\widehat{\Phi^t_{\tau(\mathcal{M})}}(s_0, A)$.

Example 3 *$c(m, c, quote)$ is violated after the execution of $sRequest$ at s_0 in Ex. 2, since $\Phi^t_{\tau(\mathcal{M}_1)}(s_0, \{sendR\}) = \{[s_0, u, v]\}$. We can check that for $A = [(sRequest, 0), (sQuote, 1)]$, $\Phi^t_{\tau(\mathcal{M}_1)}(s_0, A) = \{[s_0, u, v']\}$ where $v' = u \setminus \{\neg done(m, c, quote), \neg quote, c(m, c, quote)\} \cup \{done(m, c, quote), quote, \neg c(m, c, quote)\}$, 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 $\mathcal{L}^{mt}$ 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:

φ **at** s	(11)	α **occurs_at** s	(12)
s **at** t	(13)	α **between** $\mathsf{s}_1, \mathsf{s}_2$	(14)
$\mathsf{s}_1 \prec \mathsf{s}_2$	(15)		

where φ is a fluent formula, α is a sequence of timed action snapshots, and $\mathsf{s}, \mathsf{s}_1, \mathsf{s}_2$ are situation constants which differ from s_c. (11) (*fluent fact*) states that φ 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 α starts its execution in the situation s. On the other hand, (14) states that α 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, Γ) where D is a domain description and Γ is a set of observations of the form (11)-(15) such that $\{\mathsf{s}_0 \prec \mathsf{s},\ \mathsf{s} \prec \mathsf{s_c} \mid \mathsf{s} \in \mathbf{S}\} \subseteq \Gamma$.

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 φ **after** α **at** s, where the testing of the entailment starts from the states in $\Psi(\Sigma(\mathsf{s}))$. Given a narrative (D, Γ) and a fluent formula φ, we are also interested in knowing whether φ^t is true in a situation s for some $t_1 \leq t \leq t_2$. This is expressed using a query of the form $\varphi^{\vee[t_1,t_2]}$ **at** s; a query q of this type holds w.r.t. (D, Γ) (i.e., $(D, \Gamma) \models q$) if, for every model $M = (\Psi, \Sigma, \Delta)$ of (D, Γ), there exists $t_1 \leq t \leq t_2$ s.t. φ is true in $\Psi(\Sigma(\mathsf{s})) + t$.

Narratives and Commitments

A *narrative with commitments* is a triple (D, Γ, C) where (D, C) is a domain with commitments and Γ is a collection of observations. The semantics of a narrative with commitments (D, Γ, C) is defined by (*i*) translating it to the narrative $(\tau(\mathcal{M}), \Gamma)$ in $\mathcal{L}^{mt}$ where $\mathcal{M} = (D, C)$; and (*ii*) specifying models of $(\tau(\mathcal{M}), \Gamma)$ to be models of (D, Γ, 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** $\mathsf{s_c}$.
- *Violated* by M if $M \models (done(c, y, \varphi) \wedge c(x, y, \varphi))$ **at** $\mathsf{s_c}$.
- *Pending* w.r.t. M if $M \models \neg done(c, y, \varphi) \wedge c(x, y, \varphi)$ **at** $\mathsf{s_c}$.

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 $\mathcal{M}_1 = (D_n, C_2)$ is the domain description in Ex. 2 and Γ consists of the precedence facts $\mathsf{s}_0 \prec \mathsf{s}_1 \prec \mathsf{s}_2 \prec \mathsf{s}_3 \prec \mathsf{s_c}$ and the following observations:*

$$\neg pay \wedge \neg accept \wedge \neg quote \wedge \neg goods \ \textbf{at}\ \mathsf{s}_0$$

$$sRequest\ \textbf{occurs\_at}\ \mathsf{s}_0 \quad \textit{and} \quad sAccept\ \textbf{occurs\_at}\ \mathsf{s}_2$$

where s_0, s_1, s_2, s_3, $\mathsf{s_c}$ 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 $\Sigma(\mathsf{s}_0)$=[ ], $\Sigma(\mathsf{s}_1)$=[$\{sRequest\}$], $\Sigma(\mathsf{s}_2)$=[$\{sRequest\}, \{sQuote\}$], and $\Sigma(\mathsf{s}_3) = \Sigma(\mathsf{s_c}) = [\{sRequest\}, \{sQuote\}, \{sAccept\}]$.*
- *$\Psi([\,])$ is the state where all fluents are false and $\Psi(\mathsf{s}_i) = \widehat{\Phi^t}_{\mathcal{M}_1}(\Sigma(\mathsf{s}_i), \Psi([\,]))$.*
- *The time assignment for situation constants is given by $\Delta(\mathsf{s}_i) = i$ for each i and $\Delta(\mathsf{s_c}) = 3$. This is because each action only takes one unit of time to accomplish.*

$sQuote$ is needed since $quote$ is the precondition for $sAccept$. M is a model of the narrative N_1.

*We can prove that, for every model $(\Psi', \Sigma', \Delta')$ of $\mathcal{M}_1$, the situation assignment Σ' is identical to Σ and $\Psi'([\,])$ must satisfy $\{\neg pay, \neg accept, \neg quote, \neg goods\}$. This allows us to conclude that $N_1 \models (\neg pay$ **at** $\mathsf{s})$ for $\mathsf{s} \in \mathbf{S}$ and $N_1 \models c(c, m, pay) \wedge \neg done(c, m, pay)$ **at** $\mathsf{s_c}$. Here, $c(m, c, quote)$ is satisfied and $c(c, m, pay)$ 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 $\mathcal{L}^{mt}$. 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, Γ, 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

Baral, C.; McIlraith, S.; and Son, T. C. 2000. Formulating diagnostic problem solving using an action language with narratives and sensing. *KR*, 311–322.

Baral, C.; Son, T. C.; and Tuan, L.-C. 2002. A transition function based characterization of actions with delayed and continuous effects. *KR*, 291–302. Morgan Kaufmann Publisher.

Mallya, A. U., and Huhns, M. N. 2003. Commitments among agents. *IEEE Internet Computing* 7(4):90–93.

Singh, M. P. 1999. An ontology for commitments in multi-agent systems. *Artif. Intell. Law* 7(1):97–113.

Son, T. C.; Pontelli, E.; and Sakama, C. 2010. Formalizing Commitments Using Action Languages: The Action Language $\mathcal{L}^{mt}$. Technical report, NMSU–2010–01, `www.cs.nmsu.edu/~tson/papers/techrep1001.pdf`.