Formalizing Commitments Using Action Languages

Tran Cao Son[1], Enrico Pontelli[1], and Chiaki Sakama[2]

[1] Dept. Computer Science, New Mexico State University, tson|epontell@cs.nmsu.edu
[2] Computer and Comm. Sciences, Wakayama Univ., 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. We introduce the language $\mathcal{L}^{mt}$, an extension of the language $\mathcal{L}$, with new features motivated by the problem of representing and reasoning about commitments. The paper demonstrates how features and properties of commitments can be described in this action language. We show how $\mathcal{L}^{mt}$ can handle both simple commitment actions as well as complex commitment protocols. Furthermore, the semantics of $\mathcal{L}^{mt}$ provides a uniform solution to different problems in reasoning about commitments such as 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.

1 Introduction and Motivation

Commitments are an integral part of societies of agents. Modeling commitments has been an intensive topic of research in autonomous agents. The focus has often been on the development of ontologies for commitments [6, 15], on the identification of requirements for formalisms to represent commitments [13], and the development of formalisms for specifying and verifying protocols or tracking commitments [7, 18, 11].

Commitments are strongly related to agents' behavior and capabilities, and they are often associated with time constraints, such as a specific time (or time interval) in the future. For example, a customer will not pay for the promised goods if the goods have not been delivered; a client will have to wait for her cheque if the insurance agent does not keep her promise of entering her claim into the system; or an on-line shopper needs to pay for the order within 10 minutes after clicking the 'Check Out' button. Thus, any formalization of *commitments* should be considered in conjunction with a formalization of *actions* and *changes*, which allows us to reason about narratives in presence of (quantitative) time constraints, actions with durations, etc.

Action languages (e.g., $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ [10]), with their English like syntax and simple transition function based semantics, provide an easy and compact way for describing dynamic systems. Unlike event calculus —an action description formalism often used in the literature for reasoning about commitments—action languages can elegantly deal with indirect effects of actions and static laws. Furthermore, off-the-shelf implementations of various action languages are available. Research has provided various avenues to extend the basic action languages with advanced features, such as resources, deadlines, and preferences. Existing action languages, on the other hand, do not provide means for expressing statements like *"I will make some sandwiches"* or *"I will come at 7pm."* Both statements are about achieving a certain state of the world without specifying how. The first statement does not indicate a specific time in the future while the

second does. Moreover, with a few exceptions, action languages have been developed mostly for single-agent environments. Action languages have been successfully used in specifying and reasoning about narratives (e.g., [2, 4]). Some attempts to use action languages in formalizing commitments have been made [8, 9]. However, these attempts do not consider time constraints and actions with durations.

In this paper we answer the question of whether action languages, like $\mathcal{B}$ or $\mathcal{L}$, can be enriched with adequate features to enable the representation of domains where agents can interact through commitments, maintaining the desirable features of having a clear semantics and a declarative representation. In particular, we develop an action language, called $\mathcal{L}^{mt}$, to perform this activity. $\mathcal{L}^{mt}$, an extension of the action language $\mathcal{L}$ [2–4], is a language for multi-agent domains with features related to time, observations, and delayed effects. We show that several tasks related to reasoning with commitments, such as identifying satisfied, pending, and unsatisfied commitments, can be expressed as *queries* in $\mathcal{L}^{mt}$. Furthermore, the problem of finding a way to satisfy pending commitments can be directly addressed using planning. The language also provides a natural means for specifying, verifying, and reasoning about protocols among agents.

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

2.1 $\mathcal{L}^{m}$: Concurrency and Multi-agency

In this section, we introduce an action language which supports concurrency and multi-agency; the language is an extension of the language $\mathcal{L}$ [3, 4].

The signature of the language 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, respectively. We assume that $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$ for any two distinct $i, j \in \mathcal{AG}$. Observe also that $\bigcap_{i\in S} \mathcal{F}_i$ may be not empty for some $S \subseteq \mathcal{AG}$. This represents the fact that fluents in $\bigcap_{i\in S} \mathcal{F}_i$ are relevant to all the agents in S. A *fluent literal* (or *literal*) is either 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 literals.

A multi-agent domain specification is a set of axioms of the following forms:

a **causes** ℓ **if** ψ (1) $\quad$ φ **if** ψ (2)

impossible A **if** ψ (3) $\quad$ **initially** ℓ (4)

where $a \in \bigcup_{i\in\mathcal{AG}} \mathcal{A}_i$ is an action, ℓ is a fluent literal, ψ and φ are sets of fluent literals (interpreted as conjunctions), and $A \subseteq \bigcup_{i\in\mathcal{AG}} \mathcal{A}_i$ is a set of actions.

Axioms of type (1), (2), and (3) are referred to as *dynamic laws*, *static laws* (or *state constraints*), and *non-executability laws*, respectively. Intuitively, a dynamic law describes the direct effects of execution of one action (possibly concurrently to other actions), static laws describe integrity constraints on states of the world, and non-executability laws 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.

The semantics of a multi-agent domain is defined by the transition function Φ_D, which maps a set of actions and a state to a set of states, where $D = \bigcup_{i\in\mathcal{AG}} D_i$ is the domain description defined over the set of fluents $\bigcup_{i\in\mathcal{AG}} \mathcal{F}_i$ and the set of actions $\bigcup_{i\in\mathcal{AG}} \mathcal{A}_i$. For later use, we define an *action snapshot* as a set $\{a_i\}_{i\in\mathcal{AG}}$ where $a_i \in \mathcal{A}_i \cup \{\texttt{noop}\}$. Intuitively, each action snapshot encodes the set of actions that the agents in $\mathcal{AG}$ concurrently execute in a state. Intuitively, given an action snapshot A and a state

s, the transition function Φ_D defines the set of states that may be reached after executing A in state s. If $\Phi_D(A, s)$ is the empty set, then A is not executable in s.

An *interpretation* I of the fluents in D is a maximal consistent set of fluent literals drawn from $\mathcal{F}$. 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.

Let I be a set of fluent literals. We say that I is closed under D if for every rule (φ **if** ψ) in D, if $I \models \psi$ then $I \models \varphi$. By $Cl_D(I)$ we denote the smallest superset of I which is closed under D. A *state* of D is an interpretation that is closed under the set of static causal laws of D.

A set of actions B is *prohibited (not executable)* in a state s if there exists an executability condition of the form (3) in D such that $A \subseteq B$ and $s \models \psi$.

The *effect of an action a* in a state s of D is the set of formulae $e_A(s) = \{\ell \mid D$ contains a law a **causes** ℓ **if** ψ, $a \in A$, and $s \models \psi\}$.

Given the domain description D, if A is prohibited in s, then $\Phi_D(A, s) = \emptyset$, otherwise $\Phi_D(A, s) = \{s' \mid s' = Cl_D((s \cap s') \cup e_A(s))$ *and* s' *is a state*$\}$. The function Φ_D is extended to define $\widehat{\Phi_D}$ for reasoning about the effects of sequences of action snapshots as follows. For a state s and a sequence of action snapshots $\alpha = [A_1, \ldots, A_n]$, let $\alpha_{n-1} = [A_1, \ldots, A_{n-1}]$, we define

$$\widehat{\Phi_D}(\alpha, s) = \begin{cases} \{s\} & \text{if } n = 0 \\ \emptyset & \text{if } \widehat{\Phi_D}(\alpha_{n-1}, s) = \emptyset \vee \exists s'.[s' \in \widehat{\Phi_D}(\alpha_{n-1}, s) \wedge \Phi_D(A_n, s') = \emptyset] \\ \bigcup_{s' \in \widehat{\Phi_D}(\alpha_{n-1}, s)} \Phi_D(A_n, s') & \text{otherwise} \end{cases}$$

An *initial state* is a state s_0 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 $s_0\beta_0 s_1 \beta_1 \ldots \beta_{n-1} s_n$ such that each β_j is a snapshot, s_0 is an initial state, and $s_i \in \Phi_D(s_{i-1}, \beta_{i-1})$ for $1 \leq i \leq n$.

We allow *queries* to be composed, of the form: φ **after** α, where α is a sequence of action snapshots. A query q is true w.r.t. an initial state s_0, denoted $s_0 \models q$, if $\widehat{\Phi_D}(\alpha, s_0) \neq \emptyset$ and $\forall s \in \widehat{\Phi_D}(\alpha, s_0)$ we have that $s \models \varphi$. A query q is entailed by D ($D \models q$) if for each initial state s_0 of D we have $s_0 \models q$.

2.2 Considering Time: The Action Language $\mathcal{L}^{mt}$

The language proposed so far does not allow for the specification of durative actions. In particular, we wish to be able to model actions with delayed effects and actions whose effects can be overridden by the execution of another action. For example, pumping gasoline into the tank causes the tank to be full after 5 minutes; drilling a hole in the tank takes only 1 minute and will cause the tank never to be full. The execution of drilling 1 minute after initiating the pumping action will cause the tank to never become full. Thus, the execution of the action drill makes the tank no longer full and this effect cannot be reversed by other actions. To address the first issue, we introduce the notion of *annotated fluents*, i.e., fluents associated to relative time points, and use annotated fluents in axioms of the form (1)-(3). To deal with the second issue, we introduce the notions of *irreversible* and *reversible* processes.

The signature of $\mathcal{L}^{mt}$ extends the signature of $\mathcal{L}^{m}$ with a countable set of *process names* $\mathcal{P}$. 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 also 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 the formula obtained by replacing each literal ℓ in φ with the annotated literal ℓ^t ($\ell^{\vee[t_1,t_2]}$). An annotated formula is *single time* if it is of the form $\varphi^{\vee[t_1,t_2]}$ for some non-annotated formula φ. An annotated formula is *actual* if no literal in the formula is annotated. For an annotated formula φ, φ^{+t} is the formula obtained by replacing each ℓ^r in φ with ℓ^{r+t}.

A multi-agent domain specification is a collection of laws of the form (1)-(3) and laws of following forms:

$$\varphi \textbf{ starts } process\_id \text{ [}\textbf{reversible}|\textbf{irreversible}\text{]} \ \ell^{\hat{t}} \quad (5)$$

$$\varphi \textbf{ stops } process\_id \quad (6)$$

$$a \textbf{ starts } process\_id \text{ [}\textbf{reversible}|\textbf{irreversible}\text{]} \ \ell^{\hat{r}} \textbf{ if } \varphi \quad (7)$$

$$a \textbf{ stops } process\_id \textbf{ if } \varphi \quad (8)$$

where the φ's are sets of fluent literals, $a \in \cup_{i \in \mathcal{AG}} \mathcal{A}_i$, $\ell^{\hat{t}}$ and $\ell^{\hat{r}}$ are time annotated literals, of the form $\vee[t_1, t_2]$ with $1 \leq t_1 \leq t_2$ and $\vee[r_1, r_2]$ with $0 \leq r_1 \leq r_2$,[1] and $process\_id$ belongs to $\mathcal{P}$. The main novelty is the introduction of the notion of *process*. 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 interrupted.

The notion of a state in an $\mathcal{L}^{mt}$ domain D is similar to a state in $\mathcal{L}$ domain, in that it is an interpretation of the fluents in D and needs to satisfy the constraints imposed by static laws in D. In presence of processes, a state of the world needs to account for changes that will occur only in the future, when a process reaches its completion. For example, an action $sendPayment$ may state that the action starts a process named *payment_process* whose effect is to make $paid$ true 3, 4, or 5 units of time after the execution of the action. For this reason, 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 of 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, while IR and RE contain the irreversible and reversible processes, respectively. (s, IR, RE) is *complete* if $IR = \emptyset$ and $ER = \emptyset$.

In presence of future effects encoded by the processes, the world changes due to (*i*) the completion of a process; or (*ii*) action occurrences. Let us consider an extended state $(s, \{(x : p^1)\}, \emptyset)$ with $(x : p^1)$ as a process whose effect is p. Intuitively, if nothing happens, we would expect that p would be true in the world state one unit of time from the current time. This results in the new extended state of the world $(s \setminus \{\neg p\} \cup \{p\}, \emptyset, \emptyset)$. If instead we perform in the initial extended state an action a, whose effect is to make q true in the next moment of time, then the next state will be $(s \setminus \{\neg p, \neg q\} \cup \{p, q\}, \emptyset, \emptyset)$. Thus, in order to define the semantics of $\mathcal{L}^{mt}$ domains, we need two steps. First, we specify an update function, which computes the extended state which is t units of time from the current state assuming that no action occurs during this time span. Second, we define the transition function that takes into consideration the action occurrences.

[1] For simplicity, we do not consider $\wedge[t_1, t_2]$. This is because 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 $t_1 \leq t_i \leq t_2$.

The *update* of an extended state (s, IR, RE) is used to move forward by one time step; the time of the annotated fluents is decreased by one. Fluents that have become actual are used to update the state—in such a case we need to ensure that irreversible changes prevail over reversible ones. Formally, for $\hat{s} = (s, IR, RE)$, the set of literals that should be used in updating s in the next moment of time is

$$\tau(\hat{s}) = \{\ell \mid (x : \ell^1) \in IR\} \cup \{\ell \mid (x : \ell^1) \in RE \text{ such that } \nexists (z : \bar{\ell}^1) \in IR\}.$$

For a state s, the set of processes started and stopped by s in the next moment of time is $IR_1(s) = \{(process\_id : \ell^{\hat{t}}) \mid$ there exists a law of the form (5) with the option **irreversible** such that $s \models \varphi\}$, $RE_1(s) = \{(process\_id : \ell^{\hat{t}}) \mid$ there exists a law of the form (5) with the option **reversible** such that $s \models \varphi\}$, and $P_2(s) = \{process\_id \mid$ there exists a law of the form (6) such that $s \models \varphi\}$. For a set of process names N and a set of future effects X, let $X \setminus N = X \setminus \{(x : \ell^t) \mid x \in N, (x : \ell^t) \in X\})$.

The update of $\hat{s}$ by one unit of time is a set of extended states defined as follows:

$$update(\hat{s}) = \{(s', I(IR, s'), R(ER, s') \mid s' = Cl_D(\tau(\hat{s}) \cup (s \cap s')) \text{ and } s' \text{ is a state}\}$$

where, $I(IR, s') = (IR{-}1) \cup IR_1(s')$ and $R(ER, s') = ((RE{-}1) \cup RE_1(s')) \setminus P_2(s')$, and for a set of future effects X, we have $X - d = \{(x : \ell^{t-d}) \mid (x : \ell^t) \in X\}$. Intuitively, s' is a state that satisfies the effects that need to be true one unit from the current state. For $t > 0$, let $\hat{s} + t = \bigcup_{\hat{u} \in update(\hat{s}+t-1)} update(\hat{u})$ where $\hat{s} + 0 = \hat{s}$.

Given an extended state $\hat{s} = (s, IR, ER)$ and an annotated literal ℓ^t, we say that ℓ^t holds in $\hat{s}$, denoted $\hat{s} \models \ell^t$, if, for $t = 0$, $\hat{s} \models \ell^t$ if $s \models \ell$, and, for $t > 0$, $\hat{s} \models \ell^t$ if $\hat{u} \models \ell$ for every $\hat{u} \in \hat{s} + t$.

Let us now consider the case where an action snapshot $A = \{a_i\}_{i \in \mathcal{AG}}$ is executed in the extended state $\hat{s}$. Intuitively, there are two possible types of effects: the direct effect of the actions ($e_A(s)$) and the processes that are created by the actions. We know that $e_A(s)$ must be satisfied in the next time point. The effects of the processes starting by A in s, denoted by $procs_A(s)$, is a set of pairs (IR', RE') where:

- For each (a_i **starts** p_{id} **irreversible** $\ell^{\vee[t_1,t_2]}$ **if** φ) in D, with $a_i \in A$ and $s \models \varphi$, we have that IR' contains $(p_{id} : \ell^t)$ for some t s.t. $t_1 \leq t \leq t_2$;
- For each (a_i **starts** p_{id} **reversible** $\ell^{\vee[t_1,t_2]}$ **if** φ) in D, with $a_i \in A$, and $s \models \varphi$, we have that RE' contains $(p_{id} : \ell^t)$ for some t s.t. $t_1 \leq t \leq t_2$.

In addition, the set of processes stopped by A in s is defined as $stop_A(s) = \{p_{id} \mid$ (a_i **stops** p_{id} **if** φ) $\in D, s \models \varphi\}$. Intuitively, each (IR', RE') encodes a possible set of effects that the snapshot A can create given the current state of the world is s. $stop_A(s)$ is the set of processes that need to be stopped.

We are now ready to define transition function Φ_D^t for $\mathcal{L}^{mt}$ domains which maps extended states and action snapshots to sets of extended states. We assume that $\top$ is a special process name in $\mathcal{P}$ that does not appear in any laws of D. For a set of literals L, we define $\oplus(L) = \{(\top : \ell^1) \mid \ell \in L\}$. Given an extended state $\hat{s} = (s, IR, RE)$, a fluent literal ℓ holds in $\hat{s}$ if ℓ holds in s. The notion of executability of a set of actions can be carried over to $\mathcal{L}^{mt}$ domains without changes as it only considers the current state of the world. The transition function Φ_D^t is:

$$\Phi_D^t(A, \hat{s}) = \bigcup_{(I,R) \in procs_A(s)} update((s, IR \cup I \cup \oplus(e_A(s)), (RE \cup R) \setminus stop_A(s))$$

if A is executable in s, and $\Phi_D^t(\hat{s}, A) = \emptyset$ otherwise. Intuitively, $\Phi_D^t(\hat{s}, A)$ encodes the possible trajectories of the world given that A is executed in $\hat{s}$. We extend Φ_D^t to $\widehat{\Phi_D^t}$ which operates on sequences of action snapshots in the same way as done for Φ_D.

In presence of time, we might be interested in the states of the world given that A is executed t units of time from the current state of the world. We overload Φ_D^t and define

$$\Phi_D^t(\hat{s}, A, t) = \widehat{\Phi_D^t}(\hat{s}, [\underbrace{\{\texttt{noop}\}_{i\in\mathcal{AG}}, \ldots, \{\texttt{noop}\}_{i\in\mathcal{AG}}}_{t}] \circ [\, A\,])$$

We also write $\Phi_D^t(\hat{s}, A, t)+t_1$ to denote

$$\Phi_D^t(\hat{s}, A, t) + t_1 = \bigcup_{\hat{s'}\in\Phi_D^t(\hat{s},A,t)} \widehat{\Phi_D^t}(\hat{s'}, [\underbrace{\{\texttt{noop}\}_{i\in\mathcal{AG}}, \ldots, \{\texttt{noop}\}_{i\in\mathcal{AG}}}_{t_1}])$$

Intuitively, a member of $\Phi_D^t(\hat{s}, A, t) + t_1$ is a possible extended state after t_1 time steps from the execution of A, which in turn was executed t time steps from $\hat{s}$.

Let us define a *timed action snapshot* to be a pair (A, t) where A is an action snapshot and t is a time reference. $\widehat{\Phi_D^t}$ can also be extended to a transition function that operates on sequences of timed action snapshots $\alpha = [(A_1, t_1), \ldots, (A_n, t_n)]$ where $t_1 < t_2 < \ldots < t_n$ and A_i's are action snapshots as follows:

- For $n = 0$: $\widehat{\Phi_D^t}(\hat{s}, \alpha) = \hat{s}$; and
- For $n > 0$: $\widehat{\Phi_D^t}(\hat{s}, \alpha) = \bigcup_{\hat{u}\in\Phi_D^t(\hat{s},A_1,t_1)} \widehat{\Phi_D^t}(\hat{u}, \beta)$
  where β=$[(A_2, t_2-t_1), \ldots, (A_n, t_n-t_1)]$ if $\widehat{\Phi_D^t}(\hat{u}, \beta) \neq \emptyset$ for every $\hat{u}$∈$\Phi_D^t(\hat{s}, A_1, t_1)$; otherwise, $\widehat{\Phi_D^t}(\hat{s}, \alpha) = \emptyset$.

For a state s and a sequence of timed action snapshot α, $\widehat{\Phi_D^t}(s, \alpha) = \widehat{\Phi_D^t}((s, \emptyset, \emptyset), \alpha)$.

Example 1. Let us consider a slight modification of the the popular Netbill example [13]. Let us assume that every action takes one day to complete but the action of sending the payment might take 3 to 5 days for its effects to materialize. Also, as long as the payment has not been made, the customer can still cancel the payment. We envision $\mathcal{AG} = \{$*merchant*, *customer*$\}$. Both the *merchant* and the *customer* use the set of fluents $\mathcal{F} = \{$*request*, *paid*, *goods*, *receipt*, *quote*, *accept*$\}$; the agents use the sets of actions:

$$\mathcal{A}_{merc} = \{\textit{sendQuote}, \textit{sendGoods}, \textit{sendReceipt}\}$$
$$\mathcal{A}_{cust} = \{\textit{sendRequest}, \textit{sendAccept}, \textit{sendPayment}\}$$

The domain specification D_n consists of the following axioms ($\mathcal{P} = \{pmt\}$):

Customer	**Merchant**
sendRequest **causes** $request$	*sendGoods* **causes** $goods$
sendAccept **causes** $accept$	*sendReceipt* **causes** $receipt$
sendPayment **starts** pmt **reversible** $paid^{\vee[3,5]}$	*sendQuote* **causes** $quote$
cancelPayment **stops** pmt	**impossible** $\{$*sendReceipt*$\}$ **if** $\neg paid$
impossible $\{$*sendAccept*$\}$ **if** $\neg quote$	**impossible** $\{$*sendGoods*$\}$ **if** $\neg accept$
impossible $\{cancelPayment\}$ **if** $paid$	

The last two laws state that the *Merchant* cannot execute the action $sendReceipt$ if $\neg paid$ is true (the *Customer* has not paid yet); he cannot execute the action $sendGoods$ if $\neg accept$ is true (the *Customer* has not accepted the offer). On the other hand, the *Customer* cannot execute the action $sendAccept$ if he has not received the quote.

Let $s_0 = \{request, quote, accept, \neg paid, \neg receipt, \neg goods\}$, and $\alpha_1 = \{\texttt{noop}, sendGoods\}$. α_1 is executable in s_0 and $\Phi^t_{D_n}((s_0, \emptyset, \emptyset), \alpha_1) = \{(s'_0, \emptyset, \emptyset)\}$, where

$$s'_0 = \{request, quote, accept, \neg paid, \neg receipt, goods\}.$$

Let $\hat{u} = (s'_0, \emptyset, \emptyset)$ and $\alpha_2 = \{sendPayment, \texttt{noop}\}$. It is easy to see that

$$\Phi^t_D(\hat{u}, \alpha_2) = \{update((s'_0, \emptyset, \{(pmt : paid^i)\})) \mid i = 3, 4, 5\}$$

Thus, $\Phi^t_D(\hat{u}, \alpha_2) + 3 = \{(u', \emptyset, \emptyset)\} \cup \{update((s'_0, \emptyset, \{(pmt : paid^i)\})) \mid i = 1, 2\}$ where $u' = \{request, quote, accept, paid, \neg receipt, goods\}$. We can see that $\Phi^t_D(\hat{u}, \alpha_2) + 5 = \{(u', \emptyset, \emptyset)\}$. □

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

We demonstrate that $\mathcal{L}^{mt}$ is adequate to encode commitments and their manipulation. Commitments are encoded 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 [15]. We observe that the treatment of conditional commitments can be done similarly.

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 $\bigcap_{i \in \mathcal{AG}} \mathcal{F}_i$. We also assume that, to enable communication, 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 are used to manipulate commitments:

- *Creation:* $create(x, y, \varphi, t_1, t_2)$ describes the fact that agent x creates a commitment towards agent y in the period between t_1 and t_2. We assume that each created commitment is associated to a unique identifier;
- *Discharge:* $discharge(x, y, \varphi)$ indicates that agent x discharges a commitment towards agent y (by satisfying the request);
- *Release:* $release(x, y, \varphi)$ indicates that agent y releases x from its obligation;
- *Assignment:* $assign(x, y, k, \varphi, t_1, t_2)$ indicates that agent y transfers the commitment to a different creditor (with a new time frame);
- *Delegation:* $delegate(x, y, k, \varphi, t_1, t_2)$ indicates that agent x delegates the commitment to another debtor (with a new time frame);
- *Cancel:* $cancel(x, y, \varphi, \psi, t_1, t_2)$ indicates that x modifies the terms of the commitment (by canceling the previous one and generating a new one).

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 enabling statements, called *trigger statements*, for commitment manipulation

$$[\varphi | a] \ \textbf{triggers} \ c\_activity$$

where φ is a fluent formula, $a \in \mathcal{A}$, and $c\_activity$ is one of the activities (or 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

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

which encodes the fact that the merchant agrees to send the customer the receipt between 1 and 3 units of time since receiving the payment. The statement

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

states that the customer agrees to pay for the goods between 1 to 5 units of time after sending the acceptance notification. A more complicated trigger statement is the following, taken from an example in [7],

$$broken \ \textbf{triggers} \ create(s, c, (broken \Rightarrow paid\_10), k, k)$$

for $k \geq 3$, which represents the agreement between the service provider (s) and a customer (c) that, if the printer is broken, the service provider needs to fix it within three days or faces the consequence of paying $10 each day the printer is not fixed.

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. Intuitively, a domain with commitments is an action theory enriched with a set of (social or contractual) agreements between agents in the domain which are expressed by the set of trigger statements.

In the following, we will 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 laws and static laws originating from C.

Action Triggers: 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 fact that the commitment $c(x, y, \varphi)$ has been made by the execution of the action a. The second law starts a process which indicates that the commitment 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** $c(x, y, \varphi)$

  a **starts** $discharging(x, y, \varphi)$ **irreversible** φ **if** $c(x, y, \varphi)$

  Here, the action a stops the commitment process $c(x, y, \varphi)$ by starting a process of achieving φ. It also records the fact that the commitment $c(x, y, \varphi)$ has been satisfied.
- if $c\_activity = release(x, y, \varphi)$ then

  a **stops** $c(x, y, \varphi)$ **if** $c(x, y, \varphi)$ and a **causes** $\neg c(x, y, \varphi)$ **if** $c(x, y, \varphi)$

  belongs to D'. 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** $c(x, k, \varphi)$ a **starts** $c(x, k, \varphi)$ **reversible** $done(x, k, \varphi)^{\vee[t_1, t_2]}$

  The action stops the commitment process $c(x, y, \varphi)$ and starts the commitment process $c(x, k, \varphi)$. It also releases the process $c(x, y, \varphi)$.
- 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** $c(k, y, \varphi)$ 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** $c(x, y, \psi)$ a **starts** $c(x, y, \psi)$ **reversible** $done(x, y, \psi)^{\vee[t_1, t_2]}$

The action stops the commitment $c(x, y, \varphi)$ and starts a new one $c(x, y, \psi)$.

The translation of fluent triggers is similar. For each fluent trigger ψ **triggers** $c\_activity$, the translation is obtained from the corresponding action trigger one by:

- replacing a dynamic law of the form (a **causes** φ **if** λ) with (φ **if** λ, ψ);
- replacing a law of the form (a **starts** p_{id} [**reversible**|**irreversible**] φ **if** λ) with the law (ψ **starts** p_{id} [**reversible**|**irreversible**] φ **if** λ); and
- replacing a law of the form (a **stops** p_{id} **if** λ) with the law (ψ **stops** p_{id} **if** λ).

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 translation process mentioned above; with a slight abuse of notation, we denote $\tau(\mathcal{M}) = D \cup \tau(C)$. By definition, the domain $\tau(\mathcal{M})$ defines a transition function $\Phi^t_{\tau(\mathcal{M})}$ which determines the possible evolutions of the world given a state and the sequence of timed action snapshots $[(\alpha_1, t_1), \ldots, (\alpha_n, t_n)]$. 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 $\Phi^t_{\tau(\mathcal{M})}$. Observe that each state of $\tau(\mathcal{M})$ consists of fluent literals in D and commitments which appear in $\tau(C)$. In the definition of $\Phi^t_{\tau(\mathcal{M})}$, this is treated as any normal fluent. 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

$request$ **triggers** $create(m, c, quote, 1, 1)$ $\quad$ $accept$ **triggers** $create(m, c, goods, 1, 1)$.

So, the set of fluents in $\tau(\mathcal{M}_1)$, denoted by $\mathcal{F}_1$, consists of $\mathcal{F}$ (the set of fluents of D_1) and the commitment fluents such as $c(m, c, receipt)$, $c(c, m, paid)$, $c(m, c, quote)$, and $c(m, c, goods)$, and fluents of the form $done(x, y, \varphi)$ which are introduced by the translation from $\mathcal{M}_1$ to $\tau(\mathcal{M}_1)$. Let $s_0 = \{\neg f \mid f \in \mathcal{F}_1\}$, we have that

$$\Phi^t_{\tau(\mathcal{M}_1)}(s_0, \{sendRequest\}) = \{[s_0, u, v]\}$$

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

$$request \textbf{ starts } c(x, y, quote) \textbf{ reversible } done(x, y, \varphi)^1$$

respectively, both are the result of the translation to laws in $\tau(\mathcal{M}_1)$ of the statement

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

Let $\mathcal{M} = (D, C)$ be a domain with commitments and $\gamma = [s_0, \ldots, s_n]$ be a sequence of states in $\tau(\mathcal{M})$. Let $c(x, y, \varphi)$ be a commitment fluent appearing in γ. We say that $c(x, y, \varphi)$ 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. We say that a commitment $c(x, y, \varphi)$ is *factual* during the execution of A in s if there exists 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 violated* after the execution of A in s_0 if it is violated in every sequence of states belonging to $\widehat{\Phi^t}_{\tau(\mathcal{M})}(s_0, A)$.
- *weakly violated* after the execution of A in s_0 if it is violated in 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. Consider the domain $\mathcal{M}_1$ and the state s_0 in Ex. 2. We have that $c(m, c, quote)$ is violated after the execution of $sendRequest$ at s_0, since $\Phi^t_{\tau(\mathcal{M}_1)}(s_0, \{sendRequest\}) = \{[s_0, u, v]\}$. It is easy to verify that for $A = [(sendRequest, 0), (sendQuote, 1)]$, $\Phi^t_{\tau(\mathcal{M}_1)}(s_0, A) = \{[s_0, u, v']\}$ where $v' = u \backslash \{\neg done(m, c, quote), \neg quote, c(m, c, quote)\} \cup \{done(m, c, quote), quote, \neg c(m, c, quote)\}$. This implies that the commitment $c(m, c, quote)$ is satisfied after the execution of A in s_0. □

4 Observations and Narratives

4.1 Observation Language

We consider an extension of the action language by enabling the representation of *observations*. We extend the signature of the language $\mathcal{L}^{mt}$ with a set of *situation constants* $\mathbf{S}$, 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) $\quad \alpha$ **occurs_at** s (12) $\quad \mathsf{s}$ **at** t (13)

α **between** $\mathsf{s}_1, \mathsf{s}_2$ (14) $\quad \mathsf{s}_1 \prec \mathsf{s}_2$ (15)

where φ is a fluent formula, α is a (possibly empty) sequence of timed action snapshots, and $\mathsf{s}, \mathsf{s}_1, \mathsf{s}_2$ are situation constants which differ from s_c. Axioms of the forms (11) and (15) are called *fluent facts* and *precedence facts*, respectively. (11) states that φ is true in the situation s. (15) 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* (a *narrative*, for short) 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 $\{\mathsf{s}_0 \prec \mathsf{s},\ \mathsf{s} \prec \mathsf{s}_\mathsf{c} \mid \mathsf{s} \in \mathbf{S}\} \subseteq \Gamma$.

Observations are interpreted with respect to a domain description. While a domain description defines a transition function that characterizes what states *may* be reached when an action is executed in a state, a narrative consisting of a domain description together with a set of observations defines the possible situation histories of the system. This characterization is achieved by two functions, Σ and Ψ. While Σ maps situation constants to sequences of sets of actions, Ψ picks one among the various transitions given by $\Phi_D(A, s)$ and maps sequences of sets of actions to a unique state.

More formally, let (D, Γ) be a narrative. A *causal interpretation* of (D, Γ) is a partial function Ψ from action snapshots sequences to extended states, whose domain is nonempty and prefix-closed.[2] By $Dom(\Psi)$ we denote the domain of a causal interpretation Ψ. Notice that $[] \in Dom(\Psi)$ for every causal interpretation Ψ. A *causal model* of D is a causal interpretation Ψ such that $\Psi([])$ is an extended state of D and, for every $\alpha \circ [A] \in Dom(\Psi)$, $\Psi(\alpha \circ [A]) \in \Phi_D(A, \Psi(\alpha))$.

A *situation assignment* of $\mathbf{S}$ with respect to D is a mapping Σ from $\mathbf{S}$ into the set of sequences of action snapshots of D that satisfy the following properties: $\Sigma(\mathsf{s}_0) = []$ and, for every $\mathsf{s} \in \mathbf{S}$, $\Sigma(\mathsf{s})$ is a prefix of $\Sigma(\mathsf{s_c})$.

An *interpretation M* of (D, Γ) is a triple (Ψ, Σ, Δ), where Ψ is a causal model of D, Σ is a situation assignment of $\mathbf{S}$ such that and $\Sigma(\mathsf{s_c})$ belongs to the domain of Ψ, and Δ is a *time assignment* which maps prefixes of $\Sigma(\mathsf{s_c})$ to the set of non-negative numbers, with the following restrictions: $\Delta([\,]) = 0$ and $\Delta(\beta) \leq \Delta(\gamma)$ for every $\beta \sqsubseteq \gamma \sqsubseteq \Sigma(\mathsf{s_c})$. Additionally, for every α, β s.t. $\beta \circ \alpha \sqsubseteq \Sigma(\mathsf{s_c})$, $\Psi(\beta \circ \alpha)$ belongs to $\widehat{\Phi^t_D}(\Psi([\,]), (\beta, 0) \circ (\alpha, \Delta(\beta)))$.
For an interpretation $M = (\Psi, \Sigma, \Delta)$ of (D, Γ):

(i) α **occurs_at** s is true in M if the sequence $\Sigma(\mathsf{s}) \circ \alpha$ is a prefix of $\Sigma(\mathsf{s_c})$;
(ii) α **between** $\mathsf{s}_1, \mathsf{s}_2$ is true in M if $\Sigma(\mathsf{s}_1) \circ \alpha = \Sigma(\mathsf{s}_2)$;
(iii) φ **at** s is true in M if φ holds in $\Psi(\Sigma(\mathsf{s}))$;
(iv) $\mathsf{s}_1 \prec \mathsf{s}_2$ is true in M if $\Sigma(\mathsf{s}_1)$ is a prefix of $\Sigma(\mathsf{s}_2)$;
(v) s **at** t is true in M if $\Delta(\Sigma(\mathsf{s})) = t$.

Given two sequences of sets of actions $\alpha = [A_1, \ldots, A_n]$ and $\alpha' = [B_1, \ldots, B_m]$, we say that α is a subsequence of α', denoted by $\alpha \ll \alpha'$, if α can be obtained from α' by (*i*) deleting some B_i from α'; and (*ii*) replacing some action $a \in \mathbf{A}$ in the remaining B_i by $\mathtt{noop}$. An interpretation $M = (\Psi, \Sigma, \Delta)$ is a *model* of a narrative (D, Γ) if all facts in Γ are true in M, and there is no other interpretation $M' = (\Psi, \Sigma', \Delta')$ such that M' satisfies condition (i) above and $\Sigma'(\mathsf{s_c})$ is a subsequence of $\Sigma(\mathsf{s_c})$. These models are minimal, as they exclude extraneous actions. A narrative is *consistent* if it has a model.

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 $\Psi(\Sigma(\mathsf{s}))$. In the presence of time, given a narrative (D, Γ) and a fluent formula φ, we are also interested in knowing whether φ^t is true (resp. false) in a situation s for some $t_1 \leq t \leq t_2$. This is expressed using a query of the form

$$\varphi^{\vee[t_1, t_2]} \textbf{ at } \mathsf{s} \qquad (16)$$

We say that a query q of form (16) 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 $\Psi(\Sigma(\mathsf{s})) + t$.

4.2 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). To save space, we omit the specific details on the semantics of narratives

[2] A set X of action sequences is prefix-closed if for every sequence $\alpha \in X$, every prefix of α is also in X. The symbol $\circ$ denotes list concatenation.

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:

$$\neg paid \wedge \neg accept \wedge \neg quote \wedge \neg goods \ \textbf{at} \ \mathsf{s_0}$$
$$sendRequest \ \textbf{occurs\_at} \ \mathsf{s_0} \quad \text{and} \quad sendAccept \ \textbf{occurs\_at} \ \mathsf{s_2}$$

where $\mathsf{s_0}$, $\mathsf{s_1}$, $\mathsf{s_2}$, $\mathsf{s_3}$, $\mathsf{s_c}$ 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 $\Sigma(\mathsf{s_0})=[\,]$, $\Sigma(\mathsf{s_1})=[\{sendRequest\}]$, $\Sigma(\mathsf{s_2})=[\{sendRequest\}, \{sendQuote\}]$, and $\Sigma(\mathsf{s_3})=\Sigma(\mathsf{s_c})=[\{sendRequest\}, \{sendQuote\}, \{sendAccept\}]$.
- $\Psi([\,])$ 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.

The presence of the action $sendQuote$ can be explained by the fact that $quote$ is the precondition for $sendAccept$. We can show that M is a model of the narrative N_1.

The minimality condition of models of a narrative also allows us to prove that, for every model $(\Psi', \Sigma', \Delta')$ of $\mathcal{M}_1$, the situation assignment Σ' is identical to Σ and $\Psi'([\,])$ must satisfy $\{\neg paid, \neg accept, \neg quote, \neg goods\}$. This allows us to conclude that $N_1 \models (\neg paid \ \textbf{at} \ \mathsf{s})$ for $\mathsf{s} \in \mathbf{S}$ and $N_1 \models c(c, m, paid) \wedge \neg done(c, m, paid) \ \textbf{at} \ \mathsf{s_c}$. We can show that the commitment $c(m, c, quote)$ is satisfied, the commitment $c(c, m, paid)$ is pending, and there are no violated commitments. □

5 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".

A *protocol* is a pair (P_{id}, P) where P_{id} is a unique identifier and P is of the form:

1. a set $\{a_i\}_{i \in \mathcal{AG}}$, where $a_i \in \mathcal{A}_i \cup \{any\}$;
2. $?\varphi$ where φ is a formula;
3. $p_1; \ldots; p_n$ where p_i's are protocols;
4. $p_1 | \ldots | p_n$ where p_i's are protocols;
5. **if** φ **then** p_1 **else** p_2 where p_1, p_2 are protocols and φ is a formula;
6. **while** φ **do** p where p is a protocol and φ is a formula;
7. $p_1 < p_2$ where p_1 and p_2 are protocols.

Intuitively, Case (1) describes a request for execution of certain specific actions by certain agents (*any* indicates that we do not care about what that agent is doing); Case (2)

is a test action, which tests for the condition φ in the world state; Case (3) sequentially composes protocols, i.e., it requires first to meet the requirements of p_1, then those of p_2, etc.; Case (4) requires any of the protocols $p_1, \ldots, p_n$ to be satisfied, i.e., it represents a non-deterministic choice; Cases (5) and (6) are the usual conditional selection and iteration over protocols; Case (7) is a partial ordering among protocols, indicating that p_1 must be completed sometime before the execution of p_2. According to this definition, $(p_0, \mathit{sendGoods} < \mathit{sendPayment} < \mathit{sendReceipt})$ is a protocol.

The language can be extended to allow statements that trigger complex commitments, analogously to the case of basic commitments:

$$[a \mid \varphi] \quad \textbf{triggers} \quad \textit{complex commitment}$$

A narrative can be extended with the following type of observation:

$$P_{id} \quad \textbf{at} \quad \mathsf{s} \tag{17}$$

where P_{id} is a protocol identifier. This observation states that the protocol referred to by P_{id} has started execution at situation s. A narrative is a triple (D, Γ, C) where Γ can contain also protocol observations.

For a trajectory $h = s_0\alpha_1 s_1 \ldots \alpha_k s_k$, s_0 is called the start of h and is denoted by $start(h)$. $h[i, j]$ denotes the sub-trajectory $s_i\alpha_{i+1} \ldots \alpha_j s_j$. For every state s, $traj(s)$ denotes a set of trajectories whose start state is s. Given a protocol P and a trajectory $h = s_0\alpha_1 \ldots \alpha_k s_k$, we say that h is an *instance* of (P_{id}, P) if

- If $P = \{a_i\}_{i \in \mathcal{AG}}$ then $k = 1$ and, if $\alpha_1 = \{a_i^1\}_{i \in \mathcal{AG}}$, then for each $a_i \neq \mathit{any}$ we have $a_i = a_i^1$.
- If $P = \varphi$ then $k = 0$ and $s_0 \models \varphi$.
- If $P = p_1; \ldots; p_n$ then there exists some sequence of indices $i_0 = 0 \leq i_1 \leq \ldots \leq i_n \leq i_{n+1} = k$ such that $h[i_{i_t}, i_{i_{t+1}}]$ is an instance of p_t.
- If $P = p_1 | \ldots | p_n$ then there exists some $1 \leq i \leq n$ such that h is an instance of p_i.
- If $P =$ **if** φ **then** p_1 **else** p_2 and $s_0 \models \varphi$ then h is an instance of P if it is an instance of p_1; otherwise, h must be an instance of p_2.
- If $P =$ **while** φ **do** p and $s_0 \not\models \varphi$ then h is an instance of P if $k = 0$; otherwise, there is an index $0 \leq i \leq k$ s.t. $h[0, i]$ is an instance of p and $h[i, k]$ is an instance of P.
- If $P = p_1 < p_2$ then there exists $0 \leq i \leq j \leq k$ such that $h[0, i]$ is an instance of p_1 and $h[j, k]$ is an instance of p_2.

$(P_{id}, P) \models h$ denotes that h is an instance of (P_{id}, P).

We will now complete the definition of a model of a narrative with protocols. The notion of interpretation and the entailment relation between interpretations and observations, except for the observations of type (17), are defined as in the previous section. For an interpretation $M = (\Psi, \Sigma, \Delta)$ of a narrative (D, Γ, C) and a protocol observation $(P_{id}$ **at** $\mathsf{s}) \in C$, we say that $M \models (P_{id}$ **at** $\mathsf{s})$ if there exists some instance $s_0\alpha_1 s_1 \ldots \alpha_k s_k$ of (P_{id}, P) where: $s_0 = \Psi(\Sigma(\mathsf{s}))$, $\Sigma(\mathsf{s}) \circ [\alpha_1, \ldots, \alpha_k]$ is a prefix of $\Sigma(\mathsf{s_c})$;[3] and For every $1 \leq j \leq k$, $\Psi(\Sigma(\mathsf{s}) \circ [\alpha_1, \ldots, \alpha_j]) = s_j$. The remaining definitions related to narratives can be used unchanged for narratives with protocols.

Example 5. Let $N_2 = (D_n, \Gamma, C_2)$ where D_n is defined as in Exp. 4, C_2 is defined as in Exp. 4 with the addition of the protocol $(p_0, \mathit{sendGoods} < \mathit{sendPayment} <$

[3] We use $\circ$ to denote concatenation of lists.

sendReceipt) and Γ consists of the precedence facts $\mathsf{s}_0 \prec \mathsf{s}_\mathsf{c}$ and the single observation p_0 **at** s_0. Observe that any instance of p_0 contains the actions *sendGoods*, *sendPayment*, and *sendReceipt*, in this order. The executability condition of *sendGoods* implies that $accept$ has to be true at the time it is executed. Together with the minimality condition of models of N_2, we have that for every model $M = (\Psi, \Sigma, \Delta)$ of N_2, $\Psi(\mathsf{s}_0) \models accept$. We construct one model as follows:

- $\Sigma(\mathsf{s}_0) = [\,]$ and $\Sigma(\mathsf{s}_\mathsf{c}) = [\{\textit{sendGoods}\}, \{\textit{sendPayment}\}, \{\textit{sendReceipt}\}]$;
- $\Psi(\mathsf{s}_0) = s_0$ where $accept \in s_0$, and $\Psi(\mathsf{s}_\mathsf{c}) \in \widehat{\Phi^t_{\tau(\mathcal{M}_2)}}(s_0, \Sigma(\mathsf{s}_\mathsf{c}))$;
- $\Delta(\mathsf{s}_0) = 0$ and $\Delta(\mathsf{s}_\mathsf{c}) = 3$.

Observe that we can also infer that, in the above model, the customer must have paid right after he/she received the goods (at time 1), since (*i*) $paid$ must be true for *sendReceipt* to be executed; and (*ii*) *sendReceipt* is executed at time 2. □

6 Related Works

Our proposal is related to several works on reasoning with commitments. The main differences between our work and previous works lie in our use of an action language and in our formulation of various problems as a query in our language; this also allows the use of planning to satisfy pending commitments. The treatment of commitments and the ontology for commitments adopted in this paper is largely inspired by [13, 15]. Space limitations allow us to highlight only some representative cases.

With respect to [18], our formalization of basic commitments embedded in a domain with commitments and in a narrative of a multi-agent system allows also for a protocol specification that subsumes that of [18]. Similar differences are present w.r.t. [11], which builds on dynamic temporal logic.

Our approach has some relations to [7]; using a reactive event calculus, they provide a notion similar to narratives. Besides being different from each other in the use of an action language, our approach considers protocols and [7] does not. The same authors, in [16], propose a new language for modeling commitments in which existential quantifier of time points are used. The use of disjunctive time specification in annotating fluent formulas in our work allows us to avoid the issues raised in [12, 16].

[8, 9] also makes use of an action language in dealing with commitments and protocols. While we focus on formalizing commitments, the works [8, 9] use C+ in specifying protocols. A protocol in our definition is similar to a protocol defined in [8, 9] in that it restricts the evolution of the system to a certain sets of trajectories. In this sense, our definition of protocols provides the machineries for off-line verification of properties of protocols [17]. By introducing the observation of the from "P_{id} **at** s" we allow for the possible executions of a protocol in different states and hence different contexts. However, we do not have the notion of a transformer as in [8] and the ability to handle nested commitments as in [9].

The use of complex protocols in commitments has also been explored in [1].

The language $\mathcal{L}^{mt}$ is an evolution of a classical action languages, drawing features like static causal laws from $\mathcal{B}$ [10], narrative and observations from $\mathcal{L}$ [3, 4], and time and deadlines from $\mathcal{ADC}$ [5]. To the best of our knowledge, $\mathcal{L}^{mt}$ is the first action language with all these features, *embedded in the context of modeling multi-agent domains*. $\mathcal{L}^{mt}$ has similarities to the language PDDL 2.1 in that it can describe systems with du-

rative actions and delayed effects. $\mathcal{L}^{mt}$ has a transition function based semantics and considers observations, ir/reversible processes and multiple agents, while PDDL 2.1 does not. It should also be mentioned that $\mathcal{L}^{mt}$ differs from the event calculus in that it allows representing and reasoning with static causal laws and considers ir/reversible fluents while event calculus does not. These are also the differences between $\mathcal{L}^{mt}$ and situation calculus based approaches to dealing with duration [14].

7 Discussion and Conclusion

In this paper, we show 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 show how the problem of verifying commitments or identifying pending commitments can be posed as queries to a narrative with commitments. We show how the language can also be easily 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 that arises 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 and thus can be solved by planning techniques. An investigation of the application of multi-agent planning techniques in generating plans to satisfy pending commitments is one of our main goals in this research in the near future.

References

1. M. Baldoni et al. Commitment-based Protocols with Behavioral Rules and Correctness. *DALT*, Springer, 2010.
2. M. Balduccini and M. Gelfond. Diagnostic Reasoning with A-Prolog. *TPLP*, 3(4,5), 2003.
3. C. Baral et al. Representing Actions: Laws, Observations and Hypothesis. *JLP*, 31(1-3).
4. C. Baral et al. Formulating diagnostic problem solving using an action language with narratives and sensing. *KR*, 311–322, 2000.
5. C. Baral et al. A transition function based characterization of actions with delayed and continuous effects. *KR*, 291–302. 2002.
6. C. Castelfranchi. Commitments: From individual intentions to groups and organizations. *Int. Conf. on Multiagent Systems*, pages 41–48. The MIT Press, 1995.
7. F. Chesani et al. Commitment tracking via the reactive event calculus. *IJCAI*, 2009.
8. A.K. Chopra and M.P. Singh. Contextualizing commitment protocol. *AAMAS*, pages 1345–1352. ACM, 2006.
9. N. Desai et al. Representing and reasoning about commitments in business processes. *AAAI*, 1328–1333. 2007.
10. M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3(6), 1998.
11. L. Giordano et al. Specifying and Verifying Interaction Protocols in a Temporal Action Logic. *Journal App. Logic*, 5(2), 2007.
12. A. Mallya et al. Resolving Commitments Among Autonomous Agents. *ACL*, 2003.
13. A. Mallya and M. Huhns. Commitments among agents. *IEEE Internet Comp.*, 7(4), 2003.
14. R. Reiter. Knowledge in Actions. MIT Press. 2001.
15. M.P. Singh. An ontology for commitments in multiagent systems. *Artif. Int. Law*, 7(1), 1999.
16. P. Torroni et al. Social Commitments in Time: Satisfied or Compensated. In *DALT*, 2009.
17. P. Torroni et al. Modelling interactions via commitments and expectations. *Handbook of Research on Multi-Agent Systems*, 263–284. IGI Global, 2009.
18. P. Yolum and M.P. Singh. Flexible protocol specification and execution: applying event calculus planning using commitments. *AAMAS*, pages 527–534. ACM, 2002.