Formalizing Commitments Using Action Languages

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

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

October 28, 2010

Abstract

This paper presents an action language, called $\mathcal{L}^{mt}$, for representing and reasoning about commitments in multi-agent domains. The language is an extension of the language $\mathcal{L}$, with new features motivated by the problem of representing and reasoning about commitments. These features include time, delayed effects, ir/reversible effects, concurrent actions, and multi-agents, for specifying and reasoning about narratives in multi-agent domains. The paper provides a transition-based semantics for $\mathcal{L}^{mt}$, which makes it possible to define an entailment relation between queries and multi-agent narratives with time constraints. The paper also demonstrates how features and properties of commitments can be described in this action language. In particular, it shows 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 outstanding commitments; and (*iii*) suggesting ways to satisfy outstanding commitments.

1 Introduction and Motivation

Consider the following conversation between agents A and B:

Agent A: Do you want to do something tonight?

Agent B: Sure, what do you want to do?

Agent A: Let us have a pot-luck dinner with X. I will prepare some sandwiches and call X. But can you pick her up? Also, could you bring some soft-drinks?

Agent B: Sure. How about 7pm?

Agent A: Great.

The conversation highlights a number of activities that A and B promise to perform: A needs to prepare the sandwiches and call X. These activities need to be completed before 7pm. B, on the other hand, needs to show up at A's flat by 7pm with soft-drinks and with X. These activities are referred to as *commitments* between A and B. This conversation also provides a number of

interesting questions. What happens if A fails to make the phone call to X? What happens if B does not have enough money to buy the drinks? Can B ask A for money or can B ask X to bring the soft-drinks? More generally, what does it mean for an agent to satisfy (or violate) a commitment? What does it mean for an agent to ask other agents for help in fulfilling her commitments? How and when can we say that an agent has satisfied (or violated) a commitment?

Commitments are integral parts of societies of agents. Modeling commitments has been an intense topic of research in autonomous agents.

The focus has often been on the development of ontologies for commitments [5, 13], on the identification of basic requirements for formalisms to represent and reason about commitments [11], and the development of formalisms for specifying and verifying protocols or tracking commitments [6, 17, 10].

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, B can satisfy her commitment only if she has enough money to buy the soft-drinks; A can satisfy her commitment only if she has enough materials and knows how to make sandwiches; a customer will not pay for the promised goods if the goods have not been delivered; a client will have to wait for her check if the insurance agent does not follow through with 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 before the browser times out. As such, it is natural to think that any formalization of *commitments* should be considered in conjunction with a formalization of *actions* and *changes*. This also raises the question of whether high-level *action languages*, a popular formalism for representing and reasoning about actions and changes, are adequate for representing and reasoning about commitments, and if not, what additional features are needed for this purpose.

Action languages (e.g., $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ [9]), 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.[1] 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.

In this paper, we will develop an action language, called $\mathcal{L}^{mt}$, suitable for formalizing commitments. We achieve this by extending the action language $\mathcal{L}$ [1, 2, 3] with features such as multi-agency and future effects of actions. The introduction of these features allows representing and reasoning about actions with *reversible* and *irreversible* future effects. To the best of our knowledge, a combination of these types of actions has not been considered in the literature. The language supports also observations and protocols. We show that several tasks related to reasoning

[1] Several implementations can be found at `potassco.sourceforge.net/labs.html`

with commitments, such as identifying satisfied, outstanding, and unsatisfied commitments, can be uniformly expressed as *queries* in $\mathcal{L}^{mt}$. Furthermore, the problem of finding a way to satisfy outstanding 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 $\mathcal{L}$ with Concurrency

In this section, we review the language $\mathcal{L}$, as developed in [2], enhanced with novel capabilities to express concurrent actions. The language $\mathcal{L}$ has been already extended in [3] with static causal laws, sensing actions, and observables for diagnosis within narratives. The presentation in this paper follows [3] and considers concurrent actions. The description of the language is divided in three components: a domain description language $\mathcal{L}_D$, a language to specify observations $\mathcal{L}_O$, and a query language $\mathcal{L}_Q$.

2.1 $\mathcal{L}_D$: The Domain Description Language

The signature of $\mathcal{L}_D$ consists of two non-empty disjoint sets of symbols: the set of fluents $\mathcal{F}$, and the set of actions, $\mathcal{A}$. 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 domain description in $\mathcal{L}_D$ consists of axioms of the following forms:

$$a \ \textbf{causes} \ \ell \ \textbf{if} \ \ \psi \tag{1}$$

$$\varphi \ \textbf{if} \ \ \psi \tag{2}$$

$$\textbf{impossible} \ A \ \textbf{if} \ \ \psi \tag{3}$$

where a is an action, ℓ is a literal, ψ and φ are sets of literals (interpreted as conjunctions), and A 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. In axioms of type (1) and (3) we will omit the **if** part if ψ is a tautology.

A domain description given in $\mathcal{L}_D$ defines a transition function, which maps a set of actions and a state to a set of states. Intuitively, given a set of actions 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 an empty set it means that A is not executable in s.

Let D be a domain description in the language of $\mathcal{L}_D$. An *interpretation* I of the fluents in $\mathcal{L}_D$ 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 connective in the usual way. For example, $f \wedge q$ is true in I iff f is true in I and q is true in I. We say that φ holds in I (or I satisfies φ), denoted by $I \models \varphi$, if φ is true in I.

Let I be an interpretation and K be a set of static causal laws of the form φ **if** ψ. We say that I is closed under K if for every rule φ **if** ψ in K, if $I \models \psi$ then $I \models \varphi$. By $Cl_K(I)$ we denote the smallest superset of I which is closed under K. If K is the set of all the static laws in a domain description D, then we denote with $Cl_D(I)$ the set $Cl_K(I)$ for K equal to the set of all causal laws in 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 $A \subset B$ and $s \models \varphi$.

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

Given the domain description D containing a set of static causal laws R, we formally define $\Phi_D(A, s)$, the set of states that may be reached by executing the set of actions A in s as follows.

- If A is not prohibited (i.e., executable) in s, then

$$\Phi_D(A, s) = \{s' \mid s' = Cl_D((s \cap s') \cup e_A(s)) \textit{ and } s' \textit{ is a state}\};$$

- If A is prohibited (i.e., not executable) in s, then $\Phi_D(A, s)$ is $\emptyset$.

The function Φ_D is extended to define $\widehat{\Phi_D}$ for reasoning about the effects of sequences of sets of actions as follows. For a state s and a sequence of sets of actions $\alpha = [A_1, \ldots, A_n]$, let $\alpha_{n-1} = [A_1, \ldots, A_{n-1}]$, we define

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

We will commonly assume $\texttt{noop} \in \mathcal{A}$, an action that is always executable and not affecting the state of the world.

2.2 $\mathcal{L}_O$: The Observation Language

The language $\mathcal{L}_O$ is built from sequences of sets of actions, fluent formulas, and a set of *situation constants* **S**, containing two special constants, s_0 and s_c, denoting the initial situation and the current situation. Observations in $\mathcal{L}_O$ are axioms of the forms:

$$\varphi \textbf{ at } \mathsf{s} \tag{4}$$

$$\alpha \textbf{ between } \mathsf{s}_1, \mathsf{s}_2 \tag{5}$$

$$\alpha \textbf{ occurs\_at } \mathsf{s} \tag{6}$$

$$\mathsf{s}_1 \prec \mathsf{s}_2 \tag{7}$$

where φ is a fluent formula, α is a (possibly empty) sequence of sets of actions, and $\mathsf{s}, \mathsf{s}_1, \mathsf{s}_2$ are situation constants which differ from s_c.

Axioms of the forms (4) and (7) are called *fluent facts* and *precedence facts*, respectively. (4) states that φ is true in the situation s. (7) says that s_1 occurs before s_2. Axioms of the forms (5) and (6) are referred to as *occurrence facts*. (6) indicates that α starts its execution in the situation s. On the other hand, (5) states that α starts and completes its execution in s_1 and s_2, respectively.

2.3 Narratives

A *narrative* is a pair (D, Γ) where D is a domain description and Γ is a set of observations of the form (4)-(7) such that $\{\mathsf{s}_0 \prec \mathsf{s},\ \mathsf{s} \prec \mathsf{s}_\mathsf{c} \mid \mathsf{s} \in \mathbf{S}\} \subseteq \Gamma$. In our examples, we will often omit the set of precedence facts related to s_0 and s_c when we define Γ.

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 sequences to interpretations, 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, where $[]$ is the empty sequence of sets of actions. A *causal model* of D is a causal interpretation Ψ such that:

(i) $\Psi([])$ is a state of D; and
(ii) 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 sets of actions of D that satisfy the following properties:

(i) $\Sigma(\mathsf{s}_0) = []$;
(ii) for every $\mathsf{s} \in \mathbf{S}$, $\Sigma(\mathsf{s})$ is a prefix of $\Sigma(\mathsf{s}_\mathsf{c})$.

An *interpretation M* of (D, Γ) is a pair (Ψ, Σ), where Ψ is a causal model of D, Σ is a situation assignment of $\mathbf{S}$, and $\Sigma(\mathsf{s}_\mathsf{c})$ belongs to the domain of Ψ. For an interpretation $M = (\Psi, \Sigma)$ of (D, Γ):

(i) α **occurs_at** s is true in M if the sequence $\Sigma(\mathsf{s}) \circ \alpha$ is a prefix of $\Sigma(\mathsf{s}_\mathsf{c})$;
(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 $\Sigma(\mathsf{s}_2)$.

[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.

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 `noop`. An interpretation $M = (\Psi, \Sigma)$ is a *model* of a narrative (D, Γ) if:

(i) facts in Γ are true in M;
(ii) there is no other interpretation $M' = (\Psi, \Sigma')$ such that M' satisfies condition (i) above and $\Sigma'(\mathsf{s_c})$ is a subsequence of $\Sigma(\mathsf{s_c})$.

Observe that these models are minimal in the sense that they exclude extraneous actions. A narrative is *consistent* if it has a model.

2.4 $\mathcal{L}_Q$: The Query Language

Let us define a query language $\mathcal{L}_Q$ for narratives. Queries in $\mathcal{L}_Q$ are of the form:

$$\varphi \textbf{ after } \alpha \textbf{ at } \mathsf{s} \tag{8}$$

where α is a sequence of sets of actions. A query q of the form (8) is true in a model $M = (\Psi, \Sigma)$ of a narrative (D, Γ), denoted by $(D, \Gamma) \models_M q$, if φ is true in $\widehat{\Phi_D}(\alpha, \Psi(\Sigma(\mathsf{s}))$. A query q is entailed by a narrative (D, Γ), denoted by $(D, \Gamma) \models q$, if q is true in every model of (D, Γ).

3 $\mathcal{L}^m$ for Multiagent Domains

The action language $\mathcal{L}$ can be extended for specifying and reasoning about multiagent narratives. We will denote the new language with $\mathcal{L}^m$. To this end, we assume that each agent will have its own set of fluents and actions. Formally, a multiagent domain is defined over a signature $\langle \mathcal{AG}, \{\mathcal{F}_i, \mathcal{A}_i\}_{i \in \mathcal{AG}} \rangle$ where $\mathcal{AG}$ is a set of agent identifiers and $\mathcal{F}_i$ and $\mathcal{A}_i$ are the set 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 multiagent domain specification is a set of axioms (1), (2), and (3) where $a \in \bigcup_{i \in \mathcal{AG}} \mathcal{A}_i$ in each axiom of the form (1), and for each set of actions A in an axiom (3), A is a finite set $A \subseteq \bigcup_{i \in \mathcal{AG}} \mathcal{A}_i$.

Example 1 Let us consider the Netbill example presented in [14] modeling a protocol between a *merchant* and a *customer*. The customer can make a request for a quote, accept a quote, or send a payment. The merchant can send a quote, the goods, or the receipt. In this example, the agents are the **m***erchant* and the **c***ustomer*, i.e., $\mathcal{AG} = \{m, c\}$. The signatures of the agents are:

merchant	$\mathcal{F}_m = \left\{ \begin{array}{l} request, pay, \\ goods, receipt, \\ quote, accept \end{array} \right\}$	$\mathcal{A}_m = \left\{ \begin{array}{l} sendQuote, \\ sendGoods, \\ sendReceipt \end{array} \right\}$
customer	$\mathcal{F}_c = \left\{ \begin{array}{l} request, pay, \\ goods, receipt, \\ quote, accept \end{array} \right\}$	$\mathcal{A}_c = \left\{ \begin{array}{l} sendRequest, \\ sendAccept, \\ sendPayment \end{array} \right\}$

The description of the actions for the two agents is encoded by the following axioms:

D_c	D_m
sendRequest **causes** $request$	*sendGoods* **causes** $goods$
sendPayment **causes** pay	*sendReceipt* **causes** $receipt$
sendAccept **causes** $accept$	*sendQuote* **causes** $quote$
impossible $\{$*sendAccept*$\}$ **if** $\neg quote$	**impossible** $\{$*sendReceipt*$\}$ **if** $\neg pay$
	impossible $\{$*sendGoods*$\}$ **if** $\neg accept$

The last two laws state that the *Merchant* cannot execute the action $sendReceipt$ if $\neg pay$ 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. □

The semantics of a multiagent domain is defined by the transition function Φ_D 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. 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 and $s_i \in \Phi_D(s_{i-1}, \beta_{i-1})$ for $1 \leq i \leq n$.

The above extension is sufficient to allow us to consider a multiagent narrative. In the presence of multiple agents, some extensions to the observation language, the query language, and the notion of a narrative are necessary:

- Instead of a sequence of sets of actions in (5) or (6), we consider a sequence of action snapshots.
- A multiagent narrative is a pair (D, Γ) where D is a multi-agent domain specification defined over a signature $\langle \mathcal{AG}, \{\mathcal{F}_i, \mathcal{A}_i\}_{i \in \mathcal{AG}} \rangle$, and Γ is a set of observations with the above changes.
- The query language also consider sequences of action snapshots instead of sequences of sets of actions.

Example 2 Given the domain in Example 1, $N = (D, \Gamma)$ is a narrative where

- D is the domain description described in Example 1;
- Γ consists of the precedence facts $\mathsf{s}_0 \prec \mathsf{s}_1 \prec \mathsf{s}_2 \prec \mathsf{s}_3 \prec \mathsf{s}_\mathsf{c}$ and the following observations:

$$\neg pay \wedge \neg accept \wedge \neg quote \wedge \neg goods \ \textbf{at} \ \mathsf{s}_0$$
$$[\{sendRequest, \texttt{noop}\}] \ \textbf{occurs\_at} \ \mathsf{s}_0$$
$$[\{sendAccept, \texttt{noop}\}] \ \textbf{occurs\_at} \ \mathsf{s}_2$$
$$[\{sendQuote, \texttt{noop}\}] \ \textbf{between} \ \mathsf{s}_1, \mathsf{s}_2$$

where s_0, s_1, s_2, s_3, s_c are situation constants.

Let $M = (\Psi, \Sigma)$ where

- $\Sigma(\mathsf{s}_0) = [\,]$,
- $\Sigma(\mathsf{s}_1) = \{sendRequest, \mathtt{noop}\}$,[3]
- $\Sigma(\mathsf{s}_2) = [\{sendRequest, \mathtt{noop}\}, \{\mathtt{noop}, sendQuote\}]$, and
- $\Sigma(\mathsf{s}_3) = \Sigma(\mathsf{s}_c) = \Sigma(\mathsf{s}_2) \circ [\{sendAccept, \mathtt{noop}\}]$.

and $\Psi([\,]) = \{\neg f \mid f \in \mathcal{F}_c \cup \mathcal{F}_m\}$ which can be easily extended to allow M to be a model of N. □

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

Example 2 shows that specifying and reasoning about multiagent narratives can be effectively done using $\mathcal{L}^m$. The language, however, does not allow for the specification of durative actions. As a result, it is not possible to specify deadlines or time constraints within $\mathcal{L}$ (or $\mathcal{L}^m$). For example, the following statements cannot be represented:

- The customer is interested in the quote only within 2 hours from the completion of the request.
- The price on the quote is valid only for one day.
- The delivery only takes three days.

In this section, we propose an extension of $\mathcal{L}^m$, called $\mathcal{L}^{mt}$, that supports the representation of durative actions and time constraints in the observation and query languages. Before moving to the definition of the language, let us discuss a few issues that arise when actions with duration, time, and deadlines are considered. Consider the execution of an action a; we can observe the following situations:

- An effect of a might be delayed. For example, sending the goods to the customer causes the goods to be delivered after three days;
- An effect of a could be override by the execution of another action. For example, consider two actions: 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.

In the following, we will propose a way to deal with these types of issues. 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.

[3]The first and second action are from the customer and the merchant, respectively.

4.1 Syntax of $\mathcal{L}^{mt}$

We assume an arbitrary but fixed multiagent signature $\langle \mathcal{AG}, \{\mathcal{F}_i, \mathcal{A}_i\}_{i\in\mathcal{AG}}\rangle$ as in the previous section. For simplicity, we assume that `noop` belongs to every $\mathcal{A}_i$. The signature of $\mathcal{L}^{mt}$ contains also 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:

$$\varphi \textbf{ starts } \ process\_id \ [\textbf{reversible}|\textbf{irreversible}] \ \ell^{\hat{t}} \tag{9}$$

$$\varphi \textbf{ stops } \ process\_id \tag{10}$$

$$a \textbf{ starts } \ process\_id \ [\textbf{reversible}|\textbf{irreversible}] \ \ell^{\hat{r}} \textbf{ if } \ \varphi \tag{11}$$

$$a \textbf{ stops } \ process\_id \textbf{ if } \ \varphi \tag{12}$$

where φ is a set 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 $0 \leq r_1 \leq r_2$,[4] 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 and will materialize their effects.

Example 3 Consider the domain from Example 1. Let us assume that if the customer sends the payment, then the payment will be completed within 3 to 5 working days. The customer, however, can cancel the payment before it is completed. This can be represented in $\mathcal{L}^{mt}$ by the laws

$$\textit{sendPayment} \textbf{ starts } \ payment\_process \ \textbf{reversible} \ pay^{\vee[3,5]} \tag{13}$$

$$\textit{cancelPayment} \textbf{ stops } \ payment\_process \tag{14}$$

$$\textbf{impossible } \{cancelPayment\} \textbf{ if } \ pay \tag{15}$$

The first law creates a reversible process, since its effect can be reversed by the second law. The third law states a non-executability condition. □

[4]For simplicity of the presentation, 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 $t_1 \leq t_i \leq t_2$.

4.2 Transition Function for $\mathcal{L}^{mt}$

The notion of a state in a $\mathcal{L}^{mt}$ 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, the action $sendPayment$ in (13) states that the action starts a process named *payment_process* whose effect is to make pay 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. Intuitively, 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 $ER = \emptyset$.

Let us discuss the transitions between extended states. In presence of future effects encoded by the processes, the world changes due to (*i*) the completion of a process; or (*ii*) action occurrences. Figure 1 illustrates this. On the left, we have 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 state of the world $s \setminus \{\neg p\} \cup \{p\}$, which happens to have no more future effects. On the right, for the same extended state, action a, whose effect is to make q true in the next moment of time, occurs. We expect the next world state to be $s \setminus \{\neg p, \neg q\} \cup \{p, q\}$.

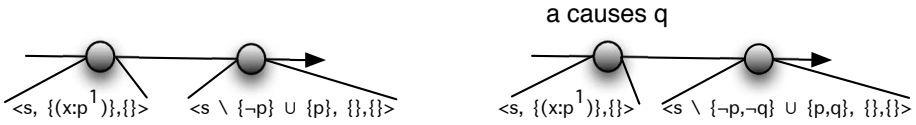

Figure 1: Transitions Between Extended States (Inertia vs. Dynamic)

The above discussion leads us to define the semantics of $\mathcal{L}^{mt}$ domains in 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.

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 (9) 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 (9) with the option **reversible** such

that $s \models \varphi\}$, and $P_2(s) = \{process\_id \mid$ there exists a law of the form (10) 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, by $X - d$ we denote the set $\{(x : \ell^{t-d}) \mid (x : \ell^t) \in X\}$. Intuitively, s' is a state that satisfies the effects that need to be true one unit from the current state. For an integer $t > 0$, let $\hat{s} + t = \bigcup_{\hat{u} \in update(\hat{s}+t-1)} update(\hat{u})$ where $\hat{s} + 0 = \hat{s}$.

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

The effects of the processes starting by α in s, denoted by $procs_\alpha(s)$, 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 $a_i \in \alpha$ 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 $a_i \in \alpha$, 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 α in s, denoted by $stop_\alpha(s)$, is a set of process names and is defined as $\{p_{id} \mid$ (a_i **stops** p_{id} **if** φ) is in D and $s \models \varphi\}$. Intuitively, each (IR', RE') encodes a possible set of effects that the snapshot α can create given the current state of the world is s. $stop_\alpha(s)$, on the other hand, is the set of processes that needed to be stopped. We are now ready to define transition function Φ^t_D for $\mathcal{L}^{mt}$ domains which maps extended states and action snapshots to sets of extended states. We need some additional notations. 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^t_D is then defined by

$$\Phi^t_D(\alpha, \hat{s}) = \bigcup_{(I,R) \in procs_\alpha(s)} update((s, IR \cup I \cup \oplus(e_\alpha(s)), (RE \cup R) \setminus stop_\alpha(s))$$

if α is executable in s, and $\Phi^t_D(\hat{s}, \alpha) = \emptyset$ otherwise. Intuitively, $\Phi^t_D(\hat{s}, \alpha)$ encodes the possible trajectories of the world given that α is executed in $\hat{s}$. We extend Φ^t_D to $\widehat{\Phi^t}_D$ 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 α is executed t

units of time from the current state of the world. We overload Φ^t_D and define

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

We also write $\Phi^t_D(\hat{s}, \alpha, t)+t_1$ to denote

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

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

The next example illustrates the computation of Φ^t_D.

Example 4 Let us consider the domain in Example 1 with the changes proposed in Example 3. Consider he state $s_0 = \{request, quote, accept, \neg pay, \neg receipt, \neg goods\}$. Let $\hat{s}_0 = (s_0, \emptyset, \emptyset)$ and $\alpha_1 = \{\texttt{noop}, sendGoods\}$. We have that α_1 is executable in s_0. We have that $e_{\alpha_1}(s_0) = \{goods\}$. So, $\Phi^t_D(\hat{s}_0, \alpha_1) = update((s_0, \{(\top : goods^1)\}, \emptyset)) = \{(s'_0, \emptyset, \emptyset)\}$ where $s'_0 = \{request, quote, accept, \neg pay$
Let $\hat{u} = (s'_0, \emptyset, \emptyset)$ and $\alpha_2 = \{sendPayment, \texttt{noop}\}$. It is easy to see that

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

Therefore,

$$\Phi^t_D(\hat{u}, \alpha_2) + 3 = \{(u', \emptyset, \emptyset)\} \cup \{update((s'_0, \emptyset, \{(payment\_process : pay^i)\})) \mid i = 1, 2\}$$

where $u' = \{request, quote, accept, pay, \neg receipt, goods\}$. It is easy to see that
$\Phi^t_D(\hat{u}, \alpha_2) + 5 = \{(u', \emptyset, \emptyset)\}$. □

Let us define a *timed action snapshot* to be a pair (α, t) where α is an action snapshot and t is a time reference. $\widehat{\Phi^t_D}$ can also be extended to a transition function that operates on sequences of timed action snapshots $A = [(\alpha_1, t_1), \ldots, (\alpha_n, t_n)]$ where $t_1 < t_2 < \ldots < t_n$ and α_i's are action snapshots as follows:

- For $n = 0$: $\widehat{\Phi^t_D}(\hat{s}, A) = \hat{s}$; and
- For $n > 0$: $\widehat{\Phi^t_D}(\hat{s}, A) = \bigcup_{\hat{u}\in\Phi^t_D(\hat{s},\alpha_1,t_1)} \widehat{\Phi^t_D}(\hat{u}, B)$
  where B=$[(\alpha_2, t_2 - t_1), \ldots, (\alpha_n, t_n - t_1)]$ if $\widehat{\Phi^t_D}(\hat{u}, B) \neq \emptyset$ for every $\hat{u}$∈$\Phi^t_D(\hat{s}, \alpha_1, t_1)$; otherwise, $\widehat{\Phi^t_D}(\hat{s}, A) = \emptyset$.

For a state s and a sequence of timed action snapshot A, we define $\widehat{\Phi^t_D}(s, A) = \widehat{\Phi^t_D}((s, \emptyset, \emptyset), A)$.

Example 5 Let us continue with Example 4. Let $\alpha_3 = \{\texttt{noop}, cancelPayment\}$ and the sequence $A = [(\alpha_2, 0), (\alpha_3, 1)]$. Consider the extended state $\hat{u}$. We have that

$$\begin{array}{rcl} \widehat{\Phi^t_D}(\hat{u}, A) & = & \bigcup_{\hat{v}\in\Phi^t_D(\hat{u},\alpha_2)} \Phi^t_D(\hat{v}, \alpha_3) \\ & = & \bigcup_{i\in\{2,3,4\}} \Phi^t_D((s'_0, \emptyset, \{(payment\_process : pay^i) \mid i = 2, 3, 4\}) \\ & = & \{(s'_0, \emptyset, \emptyset)\} \end{array}$$

In other words, the payment was canceled before pay becomes true. On the other hand, for $A = [(\alpha_2, 0), (\alpha_3, 3)]$ we have that $\widehat{\Phi^t_D}(\hat{u}, A) = \emptyset$ since there exists an extended state, namely $(u', \emptyset, \emptyset)$ in $\Phi^t_D(\hat{u}, \alpha_2)$ in which α_3 is not executable. □

4.3 The Observation Language $\mathcal{L}^{mt}_O$ and $\mathcal{L}^{mt}$ Narratives

To accommodate time constraints about the observations, we extend $\mathcal{L}_O$ with an additional type of observations of the form

$$\mathsf{s} \ \textbf{at} \ t \tag{16}$$

and refer to the new language as $\mathcal{L}^{mt}_O$. We will also require that α in (5)-(6) is a sequence of timed action snapshots.

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 (4)-(7) and (16).

Example 6 Let D_1 be the domain description described in Example 1 with the changes in Example 3 and Γ_1 be the set of observations consisting of the observations in Example 3 and the observation (s_2 **at** 3). Then, $N_1 = (D_1, \Gamma_1)$ is a $\mathcal{L}^{mt}$ narrative. □

In the following, we will extend the notion of a model of a narrative in the previous section to $\mathcal{L}^{mt}$ domains. 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 $\hat{u} \in \hat{s} + t$.

Given a $\mathcal{L}^{mt}$ narrative (D, Γ), the notions of a causal interpretation, causal model, and situation assignment defined in Section 2 can be carried over with minor changes: a causal interpretation Ψ is a mapping from sequences of action snapshots to extended states of D; a situation assignment is a mapping from the set of situation **S** into the set of sequences of action snapshots. The notion of *interpretation* needs to take into consideration time constraints and is modified as follows.

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 **S**, and Δ is a *time assignment* which is a mapping from the set of prefixes of $\Sigma(\mathsf{s_c})$ into the set of non-negative integers such that $\Delta([\,]) = 0$ and $\Delta(\beta) \leq \Delta(\gamma)$ for every $\beta \sqsubseteq \gamma \sqsubseteq \Sigma(\mathsf{s_c})$;[5] and Ψ satisfies the following conditions:

- $\Psi([\,])$ is a complete extended state of D, and
- for every β, α such that $\beta \circ \alpha \sqsubseteq \Sigma(\mathsf{s_c})$, $\Psi(\beta \circ \alpha)$ belongs to $\widehat{\Phi^t_D}(\Psi([\,]), (\beta, 0) \circ (\alpha, \Delta(\beta)))$.

Given an interpretation M of (D, Γ) and an observation $o \in \Gamma$, we say that o holds in M, denoted by $M \models o$, if

- for o of the form (4)-(7), $M \models o$ iff o is true in M as defined in Sect. 2;
- $M \models (\mathsf{s} \ \textbf{at} \ t)$ if $\Delta(\Sigma(\mathsf{s})) = t$.

[5] The notation $\beta \sqsubseteq \gamma$ denotes that β is a prefix of γ.

With this extension, we can define a model of a narrative (D, Γ) as an interpretation (Σ, Ψ, Δ) of (D, Γ) that satisfies Γ such that there exists no other interpretation $M' = (\Psi, \Sigma', \Gamma')$ such that M' satisfies Γ and $\Sigma'(\mathsf{s_c})$ is a subsequence of $\Sigma(\mathsf{s_c})$.

Example 7 Consider the narrative $N_1 = (D_1, \Gamma_1)$ in Example 2 with the following modification to D_1: the law related to $sendQuote$ is replaced by

$$sendQuote \textbf{ starts } quote\_proc \textbf{ irreversible } quote^{\vee[2,2]}$$

Let $M = (\Psi, \Sigma, \Delta)$ where Ψ and Σ are defined as in Example 3 and Δ is defined as follows. $\Delta(\Sigma(\mathsf{s}_0)) = 0$, $\Delta(\Sigma(\mathsf{s}_1)) = 1$, $\Delta(\Sigma(\mathsf{s}_2)) = 3$, $\Delta(\Sigma(\mathsf{s}_3)) = \Delta(\mathsf{s_c}) = 4$. We can show that M is a model for N_1.

Now, let us consider $N_2 = (D_2, \Gamma_2)$, where D_2 is obtained from D_1 by including the changes of Example 3 and the modification about the action $sendQuote$. Γ_2 is the union of Γ_1 and the following observations:

$$[\{\mathtt{noop}, sendGoods\}] \textbf{ occurs\_at } s_3 \qquad pay \textbf{ at } \mathsf{s}_4 \qquad \mathsf{s}_3 \textbf{ at } 5$$

where s_4 is a new situation constant satisfying $\mathsf{s}_3 \prec \mathsf{s}_4 \prec \mathsf{s_c}$. It can be shown that in any model of N_2, $\Sigma(\mathsf{s}_4)$ would have to contain the action $sendPayment$. □

As $\mathcal{L}^{mt}$ extends $\mathcal{L}$, queries of the form (8) can still be considered and the entailment relation between a narrative (D, Γ) and a query φ **after** α **at** s is defined as in Section 2.4. 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

$$\varphi^{\vee[t_1,t_2]} \textbf{ at } \mathsf{s} \tag{17}$$

We say that a query q of form (17) holds w.r.t. (D, Γ), denoted by $(D, \Gamma) \models q$, if, for every model $M = (\Psi, \Sigma, \Delta)$ of (D, Γ), there exists some t, $t_1 \leq t \leq t_2$, and φ is true in $\Psi(\Sigma(\mathsf{s})) + t$. As an example, for the narrative (D_2, Γ_2), we can show that $(D_2, \Gamma_2) \models pay^{\vee[4,6]}$ **at** s_3.

5 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 [13]. 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. A commitment where we do not care when the property is made true can be expressed using a disjunctive annotation. As an example, the statement "I will come in three hours," told by agent A to agent B, conveys the commitment $c(A, B, arrived, 3, 3)$ made by A to B.

We would like to stress that we can still 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 $\bigcap_{i \in \mathcal{AG}} \mathcal{F}_i$. Similarly, we assume that, to enable a meaningful communication, if $c(x, y, \varphi)$ is a commitment fluent, then φ is a fluent formula which uses fluents from $\mathcal{F}_x \cup \mathcal{F}_y$.

Activities are identified to enable the manipulation of commitments (x is the debtor and y is the creditor):

- *Creation of a commitment:* $create(x, y, \varphi, t_1, t_2)$ 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, and at any point in time, no two active commitments are identical;
- *Discharge of a commitment:* $discharge(x, y, \varphi)$ indicates that agent x discharges a commitment towards agent y (by satisfying the request);
- *Release of a commitment:* the syntax $release(x, y, \varphi)$ describes the fact that agent y releases x from its obligation;
- *Assign a commitment:* $assign(x, y, k, \varphi, t_1, t_2)$ indicates that agent y transfers the commitment to a different creditor (with a new time frame);
- *Delegate a commitment:* $delegate(x, y, k, \varphi, t_1, t_2)$ indicates that agent x delegates the commitment to another debtor (with a new time frame);
- *Cancel a commitment:* $cancel(x, y, \varphi, \psi, t_1, t_2)$ describes the fact that agent x modifies the terms of the commitment (by canceling the previous one and generating a new one with a new time frame).

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

$$[\varphi | a] \quad \textbf{triggers} \quad c\_activity \tag{18}$$

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 can be

$$pay \quad \textbf{triggers} \quad create(m, c, receipt, 1, 3) \tag{19}$$

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

$$sendAccept \quad \textbf{triggers} \quad create(c, m, pay, 1, 5) \tag{20}$$

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 [6],

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

for $k \geq 3$, which represents the agreement between the service provider (s) and a customer (c) which states 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.

Definition 1 *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. For example, let D_n be the domain in Example 1 and C_1 be the two statements (19) and (20), we have that (D_n, C_1) is a domain with commitments. Let us also represent another example from the literature [6].

Example 8 *A customer has signed a service agreement with a printer supplier: if a printer breaks down, the supplier guarantees to send a technician on site. The technician must intervene within three days from the call. Any delay in the intervention will incur from the supplier's side an obligation to pay a \$10 penalty per day of delay, as of the fourth day.*

Intuitively, the domain can be encoded by the action

$$repair \ \textbf{causes} \ \neg broken$$

The collection of trigger statements contains a single statement

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**

  Assume that a **triggers** $c\_activity$ belongs to C. In this case,

  - if $c\_activity = create(x, y, \varphi, t_1, t_2)$, then the laws

    $$a \ \textbf{causes} \ c(x, y, \varphi) \quad \text{and} \quad a \ \textbf{starts} \ c(x, y, \varphi) \ \textbf{reversible} \ done(x, y, \varphi)^{\vee[t_1, t_2]}$$

    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

$$a \textbf{ stops } c(x, y, \varphi) \textbf{ if } c(x, y, \varphi) \qquad a \textbf{ causes } \neg c(x, y, \varphi) \textbf{ if } c(x, y, \varphi)$$
$$a \textbf{ starts } discharging(x, y, \varphi) \textbf{ irreversible } \varphi \textbf{ if } c(x, y, \varphi)$$

  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 (the dynamic law).
- if $c\_activity = release(x, y, \varphi)$ then

$$a \textbf{ stops } c(x, y, \varphi) \textbf{ if } c(x, y, \varphi) \quad \text{and} \quad a \textbf{ causes } \neg c(x, y, \varphi) \textbf{ if } c(x, y, \varphi)$$

  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

$$a \textbf{ stops } c(x, y, \varphi) \textbf{ if } c(x, y, \varphi) \qquad a \textbf{ causes } \neg c(x, y, \varphi) \textbf{ if } c(x, y, \varphi)$$
$$a \textbf{ causes } c(x, k, \varphi) \qquad a \textbf{ starts } c(x, k, \varphi) \textbf{ reversible } done(x, k, \varphi)^{\vee[t_1,t_2]}$$

  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

$$a \textbf{ stops } c(x, y, \varphi) \textbf{ if } c(x, y, \varphi) \qquad a \textbf{ causes } \neg c(x, y, \varphi) \textbf{ if } c(x, y, \varphi)$$
$$a \textbf{ causes } c(k, y, \varphi) \qquad a \textbf{ starts } c(k, y, \varphi) \textbf{ reversible } done(k, y, \varphi)^{\vee[t_1,t_2]}$$

  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

$$a \textbf{ stops } c(x, y, \varphi) \textbf{ if } c(x, y, \varphi) \qquad a \textbf{ causes } \neg c(x, y, \varphi) \textbf{ if } c(x, y, \varphi)$$
$$a \textbf{ causes } c(x, y, \psi) \qquad a \textbf{ starts } c(x, y, \psi) \textbf{ reversible } done(x, y, \psi)^{\vee[t_1,t_2]}$$

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

- **Fluent Triggers**

  Assume that ψ **triggers** $c\_activity$ is in C. In this case,

- if $c\_activity = create(x, y, \varphi)$, then the static law $(c(x, t, \varphi) \textbf{ if } \psi)$ is an element of D';
- if $c\_activity = create(x, y, \varphi, \lambda)$, then the static law $(c(x, y, \varphi, \lambda) \textbf{ if } \psi)$ is an element of D';
- if $c\_activitiy = discharge(x, y, \varphi)$ then $\neg c(x, y, \varphi) \textbf{ if } \psi, \varphi$ belongs to D'.
- if $c\_activity = release(x, y, \varphi)$ then $\neg c(x, y, \varphi) \textbf{ if } \psi$ is an element of D'.

- if $c\_activity = assign(x, y, k, \varphi)$ then D' contains
$$\neg c(x, y, \varphi) \textbf{ if } c(x, y, \varphi), \psi \qquad c(x, k, \varphi) \textbf{ if } c(x, y, \varphi), \psi$$
- if $c\_activity = delegate(x, y, k, \varphi)$ then D' contains
$$\neg c(x, y, \varphi) \textbf{ if } c(x, y, \varphi), \psi \qquad c(k, y, \varphi) \textbf{ if } c(x, y, \varphi), \psi$$
- if $c\_activity = cancel(x, y, \varphi, \lambda)$ then D' contains
$$\neg c(x, y, \varphi) \textbf{ if } c(x, y, \varphi), \psi \qquad c(x, y, \lambda) \textbf{ if } c(x, y, \varphi), \psi$$

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 $\mathcal{M}$.

Definition 2 *Let $\mathcal{M} = (D, C)$ be a domain with commitments. The transition function $\Phi_{\mathcal{M}}$ for $\mathcal{M}$ is defined to be the function $\Phi^t_{\tau(\mathcal{M})}$.*

Example 9 *Consider the domain with commitments $\mathcal{M}_1 = (D_n, C_2)$:*

- *D_n is the domain description described in Example 1;*
- *C_2 is the set of statements consisting of (19), (20), and the following statements*
$$request \textbf{ triggers } create(m, c, quote, 1, 1)$$
$$accept \textbf{ triggers } create(m, c, goods, 1, 1)$$

So, the set of fluents in $\tau(\mathcal{M}_1)$, denoted by $\mathcal{F}_1$, consists of $\mathcal{F}$ (the setof fluents of D_1) and 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 $done(x, y, \varphi)$ 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 $u = s_0 \setminus \{\overline{request, c(m, c, quote)}\} \cup \{request, c(m, c, quote)\}$ 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 of the statement
$$request \textbf{ triggers } create(m, c, quote, 1, 1)$$
into laws in $\tau(\mathcal{M}_1)$. □

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. We will now define the notion of (un)/satisfaction of commitment.

Definition 3 *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*
- outstanding *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.

Definition 4 *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 $\widehat{\Phi^t}_{\tau(\mathcal{M})}(s_0, A)$.*
- outstanding *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 $\widehat{\Phi^t}_{\tau(\mathcal{M})}(s_0, A)$.*

We illustrate the above definition in the next example.

Example 10 *Let us consider the domain $\mathcal{M}_1$ and the state s_0 in Example 9. Since*

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

we have that $c(m, c, quote)$ is violated after the execution of $sendRequest$ at s_0. On the other hand, 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 $v' = u \setminus \{\neg done(m, c, quote), \neg quote, c(m, c, quote)\} \cup \{done(m, c, quote), quote, \neg c(m, c, quote)\}$. This implies that the commitment $c(m, c, quote)$ is satisfied after the execution of A in s_0. □

Definition 4 shows that domains with commitments are adequate for representation and hypothetical reasoning about commitments. In practice, the status of a commitment often depends on the real state of the world and the actions that have been executed in the past. As such, to represent and reason about commitments, it is necessary to consider the narrative that leads to the current state of the world.

In the rest of this section, we will show that the transition function based semantics of domains with commitments is also suitable for various tasks, such as verifying (un)satisfied commitments and identifying outstanding commitments. To this end, we define the notion of a narrative with commitments.

Definition 5 *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 of the form (4)–(7).*

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 with commitments. We illustrate this through the next example.

Example 11 *Consider the narrative* $N_1 = (D_n, \Gamma, C_2)$ *where* $\mathcal{M}_1 = (D_n, C_2)$ *is the domain description in Exp. 9 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:*

$$\neg pay \wedge \neg accept \wedge \neg quote \wedge \neg goods \ \ \textbf{at} \ \ \mathsf{s}_0$$
$$sendRequest \ \ \textbf{occurs\_at} \ \ \mathsf{s}_0$$
$$sendAccept \ \ \textbf{occurs\_at} \ \ \mathsf{s}_2$$

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* $\Sigma(\mathsf{s}_0)$=[ ], $\Sigma(\mathsf{s}_1)$=[$\{sendRequest\}$], $\Sigma(\mathsf{s}_2)$=[$\{sendRequest\}, \{sendQuote\}$], *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' *identical to* Σ*;*
- $\Psi'([\,])$ *must satisfy* $\{\neg pay, \neg accept, \neg quote, \neg goods\}$.

This allows us to conclude that $N_1 \models (\neg pay \ \ \textbf{at} \ \ \mathsf{s})$ *for* $\mathsf{s} \in \mathbf{S}$ *and* $N_1 \models c(c, m, pay) \wedge \neg done(c, m, pay) \ \ \textbf{at} \ \ \mathsf{s}_c$. □

Let us now define the notion of satisfaction of a commitment given a narrative.

Definition 6 *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** $\mathsf{s_c}$.
- outstanding *w.r.t. M if $M \models \neg done(c, y, \varphi) \wedge c(x, y, \varphi)$* **at** $\mathsf{s_c}$.

Example 12 *For the narrative $N_1 = (D_n, \Gamma, C_2)$ from Example 11, we can show that the commitment $c(m, c, quote)$ is satisfied, the commitment $c(c, m, pay)$ is outstanding, and there are no violated commitments.* □

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.

6 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".

Definition 7 (Protocol) *A protocol is a formula (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; Case (5) is the

usual conditional selection and Case (6) is the 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, $(p_0, \textit{sendGoods} < \textit{sendPayment} < \textit{sendReceipt})$ 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:

$$P_{id} \quad \textbf{at} \quad \mathsf{s} \tag{21}$$

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 $a_i \neq \textit{any}$ 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 (21), 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:

(i) $s_0 = \Psi(\Sigma(\mathsf{s}))$;

(ii) $\Sigma(\mathsf{s}) \circ [\alpha_1, \ldots, \alpha_k]$ is a prefix of $\Sigma(\mathsf{s_c})$;[6]

[6]We use $\circ$ to denote append between lists.

(iii) For every $1 \leq j \leq k$, $\Psi(\Sigma(\mathsf{s}) \circ [\alpha_1, \ldots, \alpha_j]) = s_j$.
Def. 6 can be used unchanged for narratives with protocols.

Example 13 *Let $N_2 = (D_n, \Gamma, C_2)$ where D_n is defined as in Exp. 11, C_2 is defined as in Exp. 11 with the addition of the protocol $(p_0, \mathit{sendGoods} < \mathit{sendPayment} < \mathit{sendReceipt})$ 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* $\Sigma(\mathsf{s}_\mathsf{c}) = [\{\mathit{sendGoods}\}, \{\mathit{sendPayment}\}, \{\mathit{sendReceipt}\}]$*;*
- $\Psi(\mathsf{s}_0) = s_0$ *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*) pay must be true for sendReceipt to be executed; and (*ii*) sendReceipt is executed at time* 2. □

7 Related Works

$\mathcal{L}^{mt}$ is an evolution of a classical action languages, drawing features like static causal laws from the language $\mathcal{B}$ [9], narrative and observations from the language $\mathcal{L}$ [2, 3], and time and deadlines from the language $\mathcal{ADC}$ [4]. 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*. $\mathcal{L}^{mt}$ is similar to the planning language PDDL 2.1 (and its successors) in that it provides a means for describing systems with durative 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 while event calculus does not.

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 outstanding commitments. The treatment of commitments and the ontology for commitments adopted in this paper is largely inspired by [11, 13].

With respect to [17], 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 [17]. Similar differences are present w.r.t. [10], which builds on dynamic temporal logic.

Our approach has some relations to [6]; 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 [6] does not. The same authors, in [15], 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, 15].

[7, 8] also makes use of an action language in dealing with commitments and protocols. While we focus on formalizing commitments, the works [7, 8] use C+ in specifying protocols. A protocol in our definition is similar to a protocol defined in [7, 8] 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 [16]. 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 [7] and the ability to handle nested commitments as in [8].

8 Conclusion and Future Work

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 outstanding 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.

We would like to observe that our framework (Def. 6) provides a way to identify outstanding, violated, and satisfiable commitments given a narrative (D, Γ, C). A natural question that arises is what should the agents do to satisfy the outstanding 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 outstanding 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 outstanding commitments is one of our main goals in this research in the near future.

References

[1] M. Balduccini and M. Gelfond. Diagnostic Reasoning with A-Prolog. *Theory and Practice of Logic Programming*, 3(4,5):425–461, 2003.

[2] C. Baral, M. Gelfond, and A. Provetti. Representing Actions: Laws, Observations and Hypothesis. *Journal of Logic Programming*, 31(1-3):201–243, May 1997.

[3] Chitta Baral, Sheila McIlraith, and Tran Cao Son. Formulating diagnostic problem solving using an action language with narratives and sensing. In *Proceedings of the Seventh International Conference on Principles of Knowledge and Representation and Reasoning (KR'2000)*, pages 311–322, 2000.

[4] Chitta Baral, Tran Cao Son, and Le-Chi Tuan. A transition function based characterization of actions with delayed and continuous effects. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR'2002)*, pages 291–302. Morgan Kaufmann Publisher, 2002.

[5] Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In Victor R. Lesser and Les Gasser, editors, *Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA*, pages 41–48. The MIT Press, 1995.

[6] Federico Chesani, Paola Mello, Marco Montali, and Paolo Torroni. Commitment tracking via the reactive event calculus. In *21th International Joint Conference on Artificial Intelligence (IJCAI), Pasadena, CA, USA*, pages 91–96, July, 14-17 2009.

[7] Amit K. Chopra and Munindar P. Singh. Contextualizing commitment protocol. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, pages 1345–1352. ACM, 2006.

[8] Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Representing and reasoning about commitments in business processes. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1328–1333. AAAI Press, 2007.

[9] M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3(6), 1998.

[10] L. Giordano, A. Martelli, and C. Schwind. Specifying and Verifying Interaction Protocols in a Temporal Action Logic. *Journal of Applied Logic*, 5(2), 2007.

[11] Ashok U. Mallya and Michael N. Huhns. Commitments among agents. *IEEE Internet Computing*, 7(4):90–93, 2003.

[12] Ashok U. Mallya, Pinar Yolum, and Munindar P. Singh. Resolving commitments among autonomous agents. In Frank Dignum, editor, *Advances in Agent Communication, International Workshop on Agent Communication Languages, ACL 2003, Melbourne, Australia, July 14, 2003*, volume 2922 of *Lecture Notes in Computer Science*, pages 166–182. Springer, 2003.

[13] Munindar P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.

[14] M. A. Sirbu. Credits and debits on the internet. In M. N. Huhns and M. P. Singh, editors, *Readings in Agents*, pages 2990–305. Morgan Kaufmann, San Francisco, 1998.

[15] Paolo Torroni, Federico Chesani, Marco Montali, and Paola Mello. Social commitments in time: satisfied or compensated. In *DALT*, 2009.

[16] Paolo Torroni, Pinar Yolum, Munindar P. Singh, Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, and Paola Mello. Modelling interactions via commitments and expectations. In Virginia Dignum, editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages 263–284. IGI Global, Hershey, Pennsylvania, March 2009. Chapter 11.

[17] Pinar Yolum and Munindar P. Singh. Flexible protocol specification and execution: applying event calculus planning using commitments. pages 527–534. ACM, 2002.