Social Commitments in Time: Satisfied or Compensated

Paolo Torroni, Federico Chesani, Paola Mello, and Marco Montali

DEIS, University of Bologna. V.le Risorgimento 2, 40136 Bologna, Italy

Abstract. We define a framework based on computational logic technology and on a reactive axiomatization of the Event Calculus to formalize the evolution of commitments in time. We propose a new characterization of commitments with time that enables a rich modeling of the domain, various forms of reasoning, and run-time and static verification.

1 Introduction

Social commitments are commitments made from an agent to another agent to bring about a certain property. In broad terms, a social commitment represents the commitment that an agent, called *debtor*, has towards another agent, called *creditor*, to bring about some property or state of affairs, which is the *subject* of the commitment. In some instantiations of this idea, such as [7,16], the subject of a commitment is a temporal logic formula.

Commitments are a well-known concept in Multi-Agent Systems (MAS) research [2,14]. Representing the commitments that the agents have to one another and specifying constraints on their interactions in terms of commitments provides a principled basis for agent interactions [15]. From a MAS modelling perspective, a role can be modelled by a set of commitments. For example, a seller in an online market may be understood as committing to its price quotes and a buyer may be understood as committing to paying for goods received. Commitments also serve as a natural tool to resolve design ambiguities. The formal semantics enables verification of conformance and reasoning about the MAS specifications [6] to define core interaction patterns and build on them by reuse, refinement, and composition.

Central to the whole approach is the idea of manipulation of commitments: their creation, discharge, delegation, assignment, cancellation, and release, since commitments are stateful objects that change in time as events occur. Time and events are, therefore, essential elements. Some authors distinguish between *base-level* commitments, written $\mathsf{C}(x, y, p)$, and *conditional* commitments, written $\mathsf{CC}(x, y, p, q)$ (x is the debtor, y is the creditor, and p/q are properties). $\mathsf{CC}(x, y, p, q)$ signifies that if p is brought out, x will be committed towards y to bring about q.

In this work we give emphasis to temporal aspects of commitments. We build from previous research by Mallya et al. [12,11]. In our opinion, they represent the best articulated research on time-enhanced commitments to date. The main

M. Baldoni et al. (Eds.): DALT 2009, LNAI 5948, pp. 228–243, 2010.
© Springer-Verlag Berlin Heidelberg 2010

idea in these articles is to extend a commitment framework with a way to describe time points and intervals, and alternative outcomes due to commitments extending into the uncertain future. The perspective on commitment-related temporal issues proposed by [12] mainly aims to capture the idea of validity of a commitment. Thus the previous notation $\mathsf{C}(x, p, y)$ is extended with existential and universal *temporal quantifiers*, which become prefixes of p. There are two types of quantification. By an existential quantification, $[t_1, t_2]p$, we mean that p is true at one or more moments in the interval beginning at t_1 and ending at t_2. By a universal quantification, $\overline{[t_1, t_2]}p$, we indicate instead that p is true at every moment in the interval beginning at t_1 and ending at t_2. Nestings are possible. For example, a commitment from x to y that q is going to hold at every moment in a week beginning on some day between the 1st and 20th of February could be written as follows: $\mathsf{C}(x, y, [01.02.2009, 20.02.2009](\overline{[t_{start}, t_{start} + 7days]}q))$.

This is an elegant approach which decouples the temporal quantification from the proposition, enabling reasoning about the temporal aspect without regard to the propositions' meaning. However, there are still some cases in which such a characterization is difficult to use in practical applications. The main problems are due to the lack of variables in temporal logic expressions, and from the separation between such expressions and the other parts of the represented knowledge. Among the aims of this work there is our intention to identify such cases and discuss them.

Along with a notation to express commitments, we need a language to express operations on commitments. For example, Yolum and Singh propose a notation based on the Event Calculus temporal representation language to describe commitment manipulation inside an operational framework [16]. Moreover, from a design perspective, we need an architecture in which a commitment notation, a temporal representation language and a specification and verification framework are given a specific role.

In this paper, we discuss our ongoing research about commitment frameworks. We start by introducing some issues regarding social commitment modeling, and define a number of desiderata for social commitment frameworks. We then propose a new notation for commitments and commitment specification programs: the Commitment Modeling Language ($\mathcal{CML}$). Finally, we outline an abstract commitment framework architecture and a concrete instance of it that supports $\mathcal{CML}$. In such an instance, temporal reasoning with commitments is operationalized using a reactive implementation of the Event Calculus and various verification tasks can be accomplished thanks to an underlying declarative, computational logic-based framework.

2 Some Issues Regarding Modeling

The following informal discussion is example-driven. Examples are mainly taken from the literature. We start by observing that in some cases Mallya et al.'s notation can be simplified, considering that to represent an existentially quantified time interval it is sufficient to represent a time point using a variable with a

domain. We then sketch a new possible notation that accommodates variables with domains and temporal modeling in several dimensions. Again, based on literature examples, we demonstrate the possible usage of rules to model conditional commitments. Finally we discuss time associated with commitment state changes and the issue of compensation.

2.1 Time Variables, Rules and Constraints

Let us analyze the scenario proposed by Mallya et al. in [12].

Example 1. A travel agent wishes to book an airline ticket to a certain destination, a rental car to use while there, and a hotel room at which to stay. Consider four situations:

- *Situation 1.1.* The travel agent wants the passenger to fly on a particular day while still reserving the right to choose any flight on that day. If the airline offers such a deal, it becomes committed to maintaining a condition—a booked ticket—over an extended time period.
- *Situation 1.2.* The car rental company offers a one-week free rental in January.
- *Situation 1.3.* A hotel offers an electronic discount coupon that expires today, but text on the coupon states that it can only be used during a future spring break. Note that in this case the commitment violates a constraint about time. In fact, the coupon expires before it can be used.
- *Situation 1.4.* The car rental company guarantees that its cars will not break down for at least two days, promising an immediate replacement if one does. However, if the company is closed on weekends, then a customer who rents a car on a Friday would not benefit from the warranty if the car broke down on Saturday. Thus in this case the car rental company offers a warranty that cannot be used during the period in which the warranty is valid. □

Following [12], we use the symbols h for hotel, g for guest, r for rental company, c for customer, a for airline and p for the proposition, subject of the commitment. How do we model the situations above using commitments?

Situation 1.1. Let p represent that a ticket booking is guaranteed. Thus, using an existential temporal quantifier, $[t_1, t_1 + 24hrs]p$, we express that a ticket booking is guaranteed for 1 day, as of t_1 [12].

However, in practical applications, it may be interesting to explicitly model the time at which the commitment is satisfied (e.g., when the ticket is issued). To this end, we could use an alternative notation, which associates p with a variable, and binds such a variable to a domain interval: $[T]p, T \in [t_1, t_1 + 24hrs]$. We write the commitment as follows:

$$\mathsf{C}(a, g, [T]p), t_1 \leq T, T \leq t_1 + 24hrs. \tag{1}$$

In this way, the commitment is satisfied if there is a possible value of T which falls in the range $[t_1, t_1 + 24hrs]$, and such a value can be used for further inferences.

Situation 1.2. Let p denote free rental, and t_1 January 1st. Thus, using a universal temporal quantifier, guaranteed free rental for 7 days as of time t_3 is denoted by $\overline{[t_3, t_3 + 7days]}p$. Then to express that such an interval $\overline{[t_3, t_3 + 7days]}$ is inside January, Mallya et al. [12] use a condition on t_3, namely $t_1 \leq t_3 \leq t_1 + 24days$, and they attach an existential temporal quantifier outside of the quantifier above: $[t_1, t_1 + 31days](\overline{[t_3, t_3 + 7days]}p), t_1 \leq t_3 \leq t_1 + 24days$.

Let us now use the notation introduced above, instead of existential temporal quantification. We obtain $[T, T + 7days]p, t_1 \leq T, T \leq t_1 + 24days$. Note that we simplified the notation. In particular, we do not need do distinguish any more between existentially/universally quantified time intervals, because all intervals are universally quantified, and we can drop the over-line notation. The resulting commitment is:

$$\mathsf{C}(r, c, [T, T + 7days]p), t_1 \leq T, T \leq t_1 + 24days. \tag{2}$$

Situation 1.3. Mallya et al. propose the following solution:

$$\mathsf{C}(h, c, [t_1, t_1 + 24hrs](\overline{[t_3, t_3 + 7days]}p)), t_1 + 24hrs < t_3,$$

where $t_1, t_1 + 24hrs$ is "today" (before spring break) and spring break starts on t_3 and lasts one week. In this way, we obtain two disjoint intervals. The commitment should be resolved before the end of the first interval in order not to be breached, however it can only be resolved during the second interval, which implies that it will be necessarily breached. An alternative notation for the same commitment is the following:

$$\mathsf{C}(h, t, [T_s, T_e]p), t_1 \leq T_s, T_e \leq t_1 + 7days, t_3 \leq T_s, T_e \leq t_3 + 24hrs. \tag{3}$$

In this way, we eliminate the need for nested intervals, and unresolvability can automatically be discovered by basic CLP inference [9].

Situation 1.3 shows that in a specific case, we can do away with nesting. In general, all existential temporal quantifiers can be mapped onto CLP domain restrictions, so the need for nesting intervals is only due to nested universal temporal quantifiers. An example of such a situation is the following:

Example 2. The car rental company offers a one-week free rental every month, for the whole 2009. □

In this case, we cannot do away with nested intervals. It is possible to extend Mallya et al.'s Solution 2 and write

$$\overline{[t_1, t_1 + 12months]}(\overline{[t_3, t_3 + 7days]}p), t_1 \leq t_3 \leq t_1 + 24days,$$

however that does not capture the "every month" concept, due to lack of domain variables. A different notation is needed. For example, we may use nested commitments, instead of nested intervals. Alternatively, if the "every month" concept is often used in the domain, we could define a new (non-binary) constraint and a dedicated propagation algorithm which ensures a very compact

notation and an efficient inference process. Non-binary (global) constraints are one of the prominent features of constraint programming frameworks. It may well be that a global constraints that we can use is already available off-the shelf.[1]

Situation 1.4. Mallya et al. propose the following solution:

$$\mathsf{C}(r, c, (\overline{[t_1, t_1 + 2days]}great\_car \vee [t_1, t_2]replace\_car)), t_2 < t_1 + 2days,$$

where *great_car* means that the car has not broken down, and *replace_car* represents the warranty that the rental company gives on the quality of the car, t_1 denotes the instant at which the car is rented on Friday and t_2 denotes the closing of the rental company on Friday. Using the framework presented in [12] is it possible to reason on this "warranty paradox" using CTL and realize that the warranty cannot be enjoyed if the car is rented on a Friday and it breaks down on Saturday.

Note that this modeling, however intuitive, may give rise to some counter-intuitive results. For example, c may decide to satisfy the commitment by replacing a perfectly functioning car with a broken car.

If we wish to follow Situation 1.4's description more literally, we should opt for a different formalization. For example, the commitment about the replacement car should only be a consequence of the car breaking down:

$$\mathsf{C}(r, c, [T]replace\_car) \leftarrow t_1 \leq T, T \leq t_2, \mathbf{H}(break\_down, T), T \leq t_1 + 2days \quad (4)$$

where by $\mathbf{H}(break\_down(T))$ we denote an event occurred ("**H**appened") at time T. Again, it is possible to reason on the "warranty paradox" using basic CLP inference. The result of such a reasoning would be a "real" validity interval for the warranty, which excludes Saturday.

Thus using a rule-based notation we can express many situations in a faithful way. In particular, it would be possible to express conditional commitments. However, there is another possible solution, based on the concept of *compensation*. A compensation is an action to be taken to recover from a situation of violation. To this end, we need to explicitly denote the state of commitments. For instance, we can write $[t]viol(\mathsf{C}(x, y, p))$ to indicate that a commitment has been violated at time t (due to an event occurring at time t which falsifies p, or due to the elapsing at time t of a time interval in which p was supposed to be verified). We obtain:

$$\mathsf{C}(r, c, [t_1, t_2]great\_car). \quad (5)$$

$$\mathsf{C}(r, c, [T_r]replace\_car) \leftarrow [T_b]viol(\mathsf{C}(r, c, [T_s, T_e]great\_car)), \quad T_s \leq T_b, T_b \leq T_s + 2days, T_b \leq T_e, T_b \leq T_r. \quad (6)$$

[1] See Beldiceanu and Carlsson's global constraints catalog, `http://www.emn.fr/x-info/sdemasse/gccat/`

(T_b is the time of break down, T_r is the time of replacement). Alternatively, using an extended notation, we could write:

$$compensate(r, c, [T_r]replace\_car, \mathsf{C}(r, c, [T_s, T_e]great\_car), T_b)) \leftarrow \\ T_s \leq T_b, T_b \leq T_s + 2days, T_b \leq T_e, T_b \leq T_r. \tag{7}$$

More on compensations below.

Note that we can easily refine the rules above to specify what "immediate replacement" means (1 day? 3 hours?), by posing another constraint between T_b and T_r, other than $T_b \leq T_r$.

2.2 Time of Commitments

Another temporal dimension could be interesting in many applications. It is the time of the commitment itself. To the best of our knowledge, this dimension has not been considered by previous research.

Example 3. Consider a university-like agent organization, in which agent x and faculty f belong to. There are roles with social responsibilities, which we can express by way of commitments. One such role is that of director of studies (*dos*). x has been appointed *dos* at Faculty f on October 29, 2008.

We can express that x has been appointed *dos* for 2009 at Faculty f, using a notation like:

$$\mathsf{C}(x, f, [01.01.2009, 31.12.2009]dos). \tag{8}$$

But how can we express that x has been appointed *dos* on October 29th 2008? This could be an important element of the domain. Consider a regulation that says that *a Faculty member that has been appointed director of studies cannot take more commitments in the Faculty.* The notation above does not permit to reason at this level. The closest approximation is probably: *a Faculty member cannot take more commitments in the Faculty while he is director of studies.* Or, we could resort to an additional commitment to express the appointment, beside the *dos* commitment. But this would complicate the model by increasing the number of commitments. A simple solution is to attach the duration of the commitment to the commitment itself:

$$[29.10.2008, T_{end}]active(\mathsf{C}(x, f, [01.01.2009, 31.12.2009]dos)). \tag{9}$$

2.3 Compensations

Contracts often involve deadlines and compensations. Usually, compensation actions are possibilities given to recover from a situation of violation. In a typical setting, a commitment not satisfied in time will not become satisfied by actions taken after the deadline, but it will instead incur in a further commitment from the debtor's side to take a compensation action. The extent of the compensation required may be subject to context-dependent conditions and be directly related to the time spent after the deadline before the compensation action is taken.

Example 4. According to the Italian civil code, the owners of real estate must pay taxes to the municipality (I.C.I.) between June 1 and June 16 of every tax year, for the property owned during the previous solar year. The Law allows the debtor who did not pay by the deadline to recover, by their own initiative, from such a violation by a procedure called *spontaneous revision*. The spontaneous revision procedure is permitted only if the debtor has not yet been officially notified about ongoing investigations related to such a violation. A spontaneous revision's compensation of a previous violation amounts to 3% of the amount not paid, which counts as a sanction, plus the legal interests on the amount not paid, which depend on the time elapsed between the I.C.I. payment deadline and the payment by spontaneous revision. □

To model this example, we can resort to a *compensate* notation like we did above. Let t_1 be June 1st, t_2 be June 16th, c a citizen, m a municipality, and let domain-dependent knowledge such as the interest rate $IRate$ and the amount of taxes to be paid by a citizen be defined by rules or facts such as $interest\_rate(0.025)$ and $ici(c, 100euro)$. A possible solution of Example 4 is the following:

$$\mathsf{C}(c, m, [T]pay\_ICI(Amt)), t_1 \leq T, T \leq t_2 \leftarrow ici(c, Amt). \tag{10}$$

$$compensate(c, m, [T_p]pay\_ICI(Amt), \mathsf{C}(c, m, [T_r]s\_rev(Amt_{new}))) \leftarrow$$
$$interest\_rate(IRate), Amt_{new} = Amt \times (1.03 + IRate \times (T_r - T_p)), \tag{11}$$
$$\neg\mathbf{H}(official\_notification(pay\_ICI(Amt)), T_n), T_n < T_r.$$

(s_rev stands for spontaneous revision, Amt for amount, and "=" is a CLP equality constraint).

Such examples are ubiquitous in legislation bodies, and in many domains in which contracts are used to establish rights and obligations of interacting parties. To be able to model such situations and reason about them, a notation should accommodate variables inside commitments and allow us to relate such variables with domains and expressions containing other variables.

Note that in this case *compensate* is syntactic sugar for an alternative and equally expressive notation. One could resort for instance to conditional commitments, and include original content (pay_ICI) and compensating content (s_rev) in one single CC-like fact. However, compensations could be defined in different ways depending on the domain. For example, one can introduce various degrees of violation, such as mild/serious violation, depending on whether an official notification has been issued or not. A commitment modeling framework should be flexible enough to accommodate all these needs. This notation helps to abstract away from the specific compensation semantics.

3 Desiderata for a Commitment Modeling Framework

The considerations made above suggest a number of desiderata for a commitment modeling framework that enables reasoning with time. The first two desiderata are taken from [12].

Time intervals. Contracts often involve time bounds. It should be possible to express such time bounds, in order to enable reasoning about satisfaction or breach of commitments in general.

Achievement and maintenance. Two kinds of commitment conditions are possible: achievement conditions (a ticket will be issued by the end of the day), and maintenance conditions (the car will work without breaking down for 2 days). They should both be accommodated.

Degrees of violation. It should be possible to reason about the extent of breach of a commitment, to capture ideas such as partial or mild violation of a contract.

Compensation. The language should enable associating commitments with compensation actions.

Time of commitment state changes. It should be possible to reason about the time a commitment assumes a particular state, e.g., the time a commitment is created, violated or discharged. The framework should enable reasoning about the state of commitments along time.

Meta-level reasoning. There could be commitments about commitments (and further nesting). The notation should accommodate contracts in which a commitment is about another commitment that will be created at some later point, or about some existing commitment.

Simplicity. The notation should be easy and at the same time rigorous. It should be possible to run automated reasoning tasks on commitment-based contract specifications. Some interesting reasoning tasks are: contract analysis, commitment tracking, and compliance verification.

Modularity. It should be possible to extend the commitment notation or modify the underlying theories and reasoning procedures in a modular way. Moreover, it should be possible to integrate a commitment framework with other domain knowledge, so as to enable reasoners and agents using commitments to reason using all available knowledge, possibly including ontological knowledge. Such an integration should preserve the modularity principle.

4 A New Notation for Social Commitments: $\mathcal{CML}$

We propose a new notation for social commitments. We call it $\mathcal{CML}$ (Commitment Modeling Language). To enable reasoning, we consider commitments as a part of a knowledge base. In particular, social commitments are specified inside $\mathcal{CML}$ programs ($CPrograms$), which could describe for example a contract.

A $\mathcal{CML}$ program is made of rules. A rule in the form

$$CRuleHead \leftarrow CRuleBody. \tag{12}$$

is used to define effects of events on the state of commitments. More specifically, the user can use such rules to define for instance which events create, discharge,

or break which commitments, in the style of [16]. The body defines the context, i.e., the conditions that must hold in order for an event to have an effect on the state of a commitment. If no context is defined, the rule is said to be a fact.

Atoms in the body of rules may be external predicates, not defined in the commitment specification program (this should be allowed because of the modularity principle), or they can be fluents modeling the state of commitments. Such a state can be associated with existentially quantified temporal variables or with universally quantified intervals.

The $\mathcal{CML}$ syntax, shown in Figure 1, is easily extensible to accommodate various models of commitment evolution. For example, *p_viol* has been introduced alongside *viol* and *active* as a possible new commitment state, and a new type of commitment operation, e.g., *compensate*, could be added to the language to provide the user with such a high-level abstraction.

$$
\begin{aligned}
CMLProgram &::= CRules \\
CRules &::= CRule[CRules] \\
CRule &::= CRuleHead\text{"."} \mid CRuleHead\text{" } \leftarrow \text{ "}CRuleBody\text{"."} \\
CRuleHead &::= OP_{\mathsf{C}}\text{"("}Terms\text{", "}Commitment\text{")"} \\
OP_{\mathsf{C}} &::= =\text{ "}create\text{"} \mid \text{"}discharge\text{"} \mid \text{"}cancel\text{"} \mid \text{"}release\text{"} \mid \text{"}assign\text{"} \mid \text{"}delegate\text{"} \mid \ldots \\
CRuleBody &::= CRuleBodyElem \mid [\text{", "}CRuleBody] \\
CRuleBodyElem &::= holds\text{"("}Interval\ State\text{"("}Commitment\text{", "}Time\text{"))"} \mid \\
&\quad\ Atom \mid Constraint \\
Commitment &::= \mathsf{C}\text{"("}Agent\text{", "}Agent\text{", "}[Interval]CAtom\text{")"} \\
Interval &::= \text{"["}TExpr[\text{", "}TExpr]\text{"]"} \\
TExpr &::= Time\ OP_T\ Time \mid Time\ OP_T\ Duration \\
OP_T &::= \text{ " + "} \mid \text{" } - \text{ "} \\
Time &::= Date \mid Numeral \mid Variable \mid TExpression \\
Duration &::= Numeral\ Granularity \\
Granularity &::= \text{"}hrs\text{"} \mid \text{"}days\text{"} \mid \text{"}weeks\text{"} \mid \text{"}months\text{"} \mid \ldots \\
Agent &::= Term \\
Atom &::= Ident \mid Ident\text{"("}Terms\text{")"} \\
Term &::= Atom \mid Numeral \mid Variable \\
Terms &::= Term[\text{", "}Term] \\
CAtom &::= Atom \mid Commitment \\
Constraint &::= Variable\text{" } \in \text{ "}Domain \mid Variable\ OP_{CLP}\ TExpr \\
State &::= \text{"}viol\text{"} \mid \text{"}p\_viol\text{"} \mid \text{"}active\text{"} \mid \ldots \\
OP_{CLP} &::= \text{ " = "} \mid \text{" } \neq \text{ "} \mid \text{" } \leq \text{ "} \mid \text{" } \geq \text{ "} \mid \text{" } < \text{ "} \mid \text{" } > \text{ "} \\
Domain &::= Interval \mid Set
\end{aligned}
$$

Fig. 1. Syntax of $\mathcal{CML}$ programs

A sample $CProgram$, modeling Situation 1.4, is the following:

$$create(rent\_a\_car(T_c, T_e), \mathsf{C}(r, c, [T_c, T_c + 2days]great\_car)). \tag{13}$$

$$create(car\_broken(T_b), \mathsf{C}(r, c, [T_r]replace\_car)) \leftarrow$$
$$T_r \leq T_b + 24hours, holds([T_b]viol(\mathsf{C}(r, c, [T_s, T_e]great\_car), T_b)). \tag{14}$$

Renting a car at time T_c until T_e creates a commitment that for 2 days as of T_c the car does not break down. The car breaking down at a time T_b creates a commitment that the car must be replaced within 24 hours of the incident, if the breakdown has caused a breach of commitment.

While $\mathcal{CML}$ provides very expressive constructs such as variables with domains, constraints and rules, on the other hand it does not explicitly accommodate temporal logic expressions, such as $p\mathcal{U}q$ or $\bigcirc p$. We are currently investigating whether and how temporal logic formulae can be mapped onto $\mathcal{CML}$ expressions.

Two fundamental aspects of commitment frameworks are manipulation and reasoning [15]. Manipulation operates on the state of commitments.

4.1 States of Commitments

Recently, many authors proposed different possible evolutions of the commitment state, from an initial state after creation, down to its satisfaction through discharge, delegation or cancellation operations, or else to its violation due to the occurrence of events that contradict the subject of the agreement represented by the commitment itself. For instance, in [7], Fornara and Colombetti propose the following set of states for a commitment: empty (e), cancelled (c), precommitment (p), conditional (cc), active (a), fulfilled (f), and violated (v). The states and their transitions are depicted in Figure 2.

5 Commitment Manipulation and Reasoning

Usually, once the conditions specified in the commitment are either satisfied (for an achievement commitment) or violated (for a maintenance commitment), the commitment assumes a final state. However, as discussed above, if we consider also relevant temporal aspects, such as deadlines, we could define a finer-grained

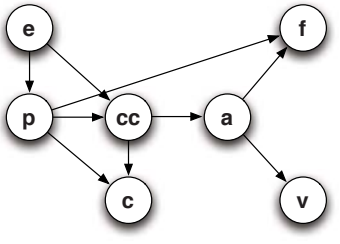

Fig. 2. Fornara & Colombetti's commitment state transitions [7]

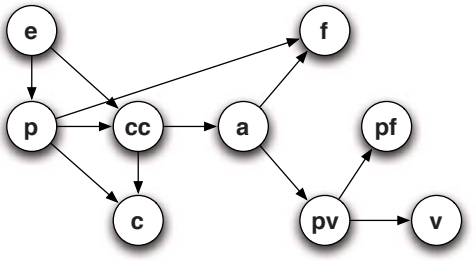

Fig. 3. A possible extension to a commitment state transitions accounting for partial violation (pv) and partial fulfillment (pf) of commitments

characterization of the state of a commitment. For example, after a deadline has passed, the debtor may still opt for a belated action that partially makes up for the violation. It may be interesting to distinguish among (1) commitment satisfied in time, (2) commitment "violated" before the deadline but "satisfied" after the deadline (partial violation/partial satisfaction), and (3) violated commitment. Such a distinction is depicted in Figure 3, which is an extended version of Figure 2.

This issue is discussed in depth by Fornara and Colombetti in [8], where the authors propose a new commitment life-cycle accommodating two states after *violated*: *extinguished* and *irrecoverable*. Our contribution here is not in the theoretical underpinning of sanctions and violations, but rather in building a framework where different theories of violation and sanction may be instantiated and operationalized.

5.1 Reasoning about Commitments

Yolum and Singh [16] propose to reason about commitments using the Event Calculus ($\mathcal{EC}$) [10]. The $\mathcal{EC}$ was introduced by Kowalski and Sergot as a logic programming framework for representing and reasoning about events and their effects. Basic concepts are that of *event*, happening at a point in time, and *property* (or *fluent*), holding during time intervals. Fluents are initiated/terminated by occurring events. There are many different formulations of the $\mathcal{EC}$ axioms. A simple one, taken from [5], is the one below (F stands for *Fluent*, Ev for *Event*).

$$\begin{aligned} holds\_at(F,T) \leftarrow\ & initiates(Ev,F,T_{Start}) \\ & \wedge T_{Start} < T \wedge \neg clipped(T_{Start},F,T). \end{aligned} \qquad (ec_1)$$

$$\begin{aligned} clipped(T_1,F,T_3) \leftarrow\ & terminates(Ev,F,T_2) \\ & \wedge T_1 < T_2 \wedge T_2 < T_3. \end{aligned} \qquad (ec_2)$$

$$\begin{aligned} initiates(Ev,F,T) \leftarrow\ & happens(Ev,T) \wedge holds(F_1,T) \\ & \wedge \ldots \wedge holds(F_N,T). \end{aligned} \qquad (ec_3)$$

$$\begin{aligned} terminates(Ev,F,T) \leftarrow\ & happens(Ev,T) \wedge holds(F_1,T) \\ & \wedge \ldots \wedge holds(F_N,T). \end{aligned} \qquad (ec_4)$$

Axioms ec_1 and ec_2 are the general ones of $\mathcal{EC}$, whereas ec_3 and ec_4 are user-defined, domain-specific axiom schemas. The $\mathcal{EC}$ is a suitable formalism to specify the effects of commitment manipulation, and reason from such operations. As a sample fragment of Yolum and Singh's formalization, consider a *create* operation, whose purpose is to establish a commitment, and can only be performed by the debtor. To express that an event $e(x)$ carried out by x at time t creates a commitment $\mathsf{C}(x, y, p)$, Yolum and Singh define the operation $create(e(x), \mathsf{C}(x, y, p))$ in terms of $happens(e(x), t) \wedge initiates(e(x), \mathsf{C}(x, y, p), t)$.

In the same way, the semantics of $\mathcal{CML}$ can be given in terms of $\mathcal{EC}$ programs. This helps *simplicity*, because the language of $\mathcal{EC}$ is very simple, and *modularity*, because for different domains we can define different theories of commitments.

The $\mathcal{EC}$ is an effective framework for temporal reasoning. It has been extensively used in the literature to carry out two main reasoning tasks: deductive *narrative verification*, to check whether a certain fluent holds given a narrative (set of events), and abductive *planning*, to simulate a possible narrative which satisfies some requirements [13]. Chittaro and Montanari [4] proposed a way to use the $\mathcal{EC}$ for run-time monitoring and verification. It is based on a mechanism to cache the outcome of the inference process every time the knowledge base is updated by a new event. In a nutshell, the *Cached Event Calculus* ($\mathcal{CEC}$) computes and stores fluents' *maximum validity intervals* (MVIs), which are the maximum time intervals in which fluents hold, according to the known events. The set of cached validity intervals is then extended/revised as new events occur or get to be known. Therefore, the $\mathcal{EC}$ can be used as a basis for reasoning on commitments in many ways, including not only planning and static verification, but also tracking, depending on the $\mathcal{EC}$ implementation used (abductive, deductive, reactive).

6 Social Commitment Framework Architecture

We propose an abstract, layered architecture that enables modeling and reasoning with social commitments. It consists of:

- a user application layer;
- a commitment modeling layer;
- a temporal representation and reasoning layer;
- a reasoning and verification layer.

On the top layer, the user can define contracts or agent social interaction rules using commitments. Such definitions are based on a language provided by the layer below. The commitment modeling language is implemented using a temporal representation and reasoning framework, which is in turn built on top of a more general reasoning and verification framework, which lies at the bottom layer. It is important to rely on a formal framework that accommodates various forms of verification, because in this way commitments can be operationalized and the user can formally analyze commitment-based contracts, reason on the state of commitments, plan for actions needed to reach states of fulfillment, and track the evolution of commitments at run-time.

Indeed, the underlying reasoning and verification layer must be powerful enough to implement a temporal representation and reasoning layer. We propose a concrete instance of such an architecture, represented in Figure 4.

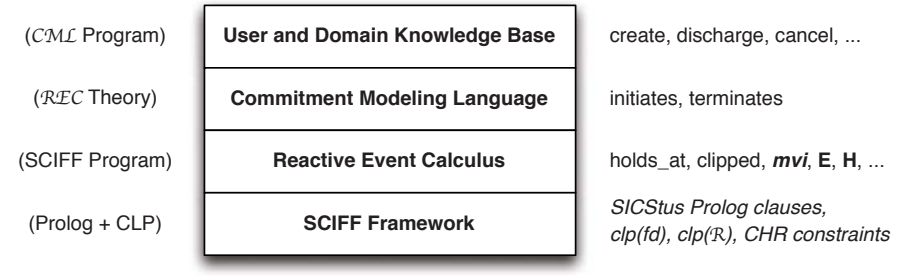

Fig. 4. Social commitment framework architecture

At the bottom layer, we find a number of Prolog+CLP modules which implement the SCIFF family of proof-procedures and provide the SCIFF language to the layer above [1]. The SCIFF framework is based on abductive logic programming and it consists of a declarative specification language and a family of proof-procedures for reasoning from SCIFF specifications. Some kinds of reasoning are: deduction, hypothetical reasoning, static verification of properties, compliance checking and run-time monitoring. In general, SCIFF comes in hand for a number of useful tasks in the context of agent interaction. A high-level description of SCIFF and of its usage is given in [15], also in relation with commitments. The CLP solvers integrated in SCIFF can work with discrete and dense domains, depending on the application needs, and they are particularly useful for reasoning along the temporal dimension.

On top of the SCIFF layer there is a SCIFF implementation of the $\mathcal{EC}$, which uses ideas taken from $\mathcal{CEC}$ and thus enables runtime verification. It is called the Reactive Event Calculus ($\mathcal{REC}$). This layer provides to the layer above the $\mathcal{REC}$ language, which consists of domain-dependent axioms with schemas ec_3 and ec_4.

In the third layer, the constructs that define the Commitment Modeling Language ($\mathcal{CML}$), i.e., the notation proposed above, are written by way of $\mathcal{REC}$ theories. Thus this layer will provide the top layer with the language to write a $CProgram$. The top layer consists of user and domain-dependent knowledge encoded into a $CProgram$. An example of a program for the top layer was given in Section 4.

We believe that such an architecture, and its instantiation based on SCIFF, $\mathcal{REC}$, and the $\mathcal{CML}$, can successfully address the desiderata identified above. Modularity is achieved in two directions: in the vertical direction, by making $\mathcal{CML}$ programs, $\mathcal{EC}$ theory, and commitment theories independent of each other, and in the horizontal direction, by letting the user refer to external inputs by way of simple atoms. Atoms can be mapped into function calls via suitable interfaces such as those available in most Prolog engines. $\mathcal{CML}$ is a simple and easily

extensible language, which consists of primitives such as *create*, *discharge*, etc., in the style of [16]. The language is expressive enough to express time intervals, achievement and maintenance conditions, and time of commitment state change. Thanks to the expressivity of the language and to the modularity of the architecture, it is possible to extend the framework to model different kinds of violation and powerful new constructs such as compensation. In fact, the states of commitments and manipulation operations are not hard-wired in the architecture, but they can be (re)defined by the user. Finally, $\mathcal{CML}$ accommodates meta-lavel reasoning on commitments, and the underlying $\mathcal{REC}$ engine can reason about commitments at all levels by treating a commitment state as a fluent which holds for a period of time.

7 Conclusion

We identified some issues regarding the representation of commitments which are still open, and we formulated a number of desiderata for a commitment modeling framework. To the best of our knowledge, in the state of the art there is no framework that satisfies all the desiderata. We believe that a possible answer could come from a commitment framework organized into four layers.

On top of the stack, at the user level, contracts can be specified by way of commitment programs. We identified in SCIFF a potential candidate for the bottom layer and we defined a notation for top-level programs. A prototypical implementation exists of all the four layers.[2] Its usage for commitment tracking purposes is demonstrated in a companion paper [3].

Our discussion was informal and example-driven. We gave emphasis to temporal aspects of commitments in relation with deadlines. However, deadlines are only a special case of temporal constraints and CLP constraints in general. Surely there are many other types of constraint that could be very useful for modeling the domain correctly and compactly. In particular, global constraints capture common patterns and help specify complex and recurring constraints in a simple way. Each global constraint comes with an effective and efficient propagation algorithm capable of powerful inference. A useful activity could be to isolate a subset of CLP constraints of interest for commitment-related domains. A concrete implementation of the commitment modeling framework should include a library of such CLP constraints. To the best of our knowledge, the inference potential of such a technology, unlocked by the architecture we propose, is unprecedented in the domain of commitments.

Currently we are evaluating the $\mathcal{CML}$ language and framework empirically on a number of case studies. Some of the formal properties of the $\mathcal{CML}$ framework, in particular insofar as reasoning is concerned, are discussed in [3]. We plan to focus not on the expressivity of the top-level commitment programming language and on the notion of sanctions. To this extent, we found more recent work by Fornara and Colombetti [8] inspiring and very relevant to our approach. Along this line, we also intend to investigate how $\mathcal{CML}$ and the abstract architecture

[2] `http://lia.deis.unibo.it/research/sciff/`

fit into the concrete application domain of electronic institutions. Commitments or obligations are also included in their modeling, and often they work with deadlines of events taking place instead of time.

Acknowledgments. We thank the anonymous reviewers for their useful comments. This work has been partially supported by the FIRB project *TOCAI.IT*.

References

1. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Verifiable agent interaction in abductive logic programming: the SCIFF framework. ACM Transactions on Computational Logic 9(4), 1–43 (2008)
2. Castelfranchi, C.: Commitments: From individual intentions to groups and organizations. In: Lesser, V.R., Gasser, L. (eds.) Proceedings of the First International Conference on Multi-Agent Systems, pp. 41–48. The MIT Press, Cambridge (1995)
3. Chesani, F., Mello, P., Montali, M., Torroni, P.: Commitment tracking via the Reactive Event Calculus. In: Boutilier, C. (ed.) Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI), pp. 91–96. AAAI Press, Menlo Park (2009)
4. Chittaro, L., Montanari, A.: Efficient temporal reasoning in the cached event calculus. Computational Intelligence 12(2), 359–382 (1996)
5. Chittaro, L., Montanari, A.: Temporal representation and reasoning in artificial intelligence: Issues and approaches. Annals of Mathematics and Artificial Intelligence 28(1-4), 47–106 (2000)
6. Fisher, M., Bordini, R.H., Hirsch, B., Torroni, P.: Computational logics and agents: A road map of current technologies and future trends. Computational Intelligence 23(1), 61–91 (2007)
7. Fornara, N., Colombetti, M.: Operational specification of a commitment-based agent communication language. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 536–542. ACM Press, New York (2002)
8. Fornara, N., Colombetti, M.: Specifying and enforcing norms in artificial institutions. In: Normative Multi-Agent Systems. Dagstuhl Seminar Proceedings, vol. 07122 (2007), `http://drops.dagstuhl.de/opus/volltexte/2007/909`
9. Jaffar, J., Maher, M.: Constraint logic programming: a survey. Journal of Logic Programming 19-20, 503–582 (1994)
10. Kowalski, R.A., Sergot, M.: A logic-based calculus of events. New Generation Computing 4(1), 67–95 (1986)
11. Mallya, A.U., Huhns, M.N.: Commitments among agents. IEEE Internet Computing 7(4), 90–93 (2003)
12. Mallya, A.U., Yolum, P., Singh, M.P.: Resolving commitments among autonomous agents. In: Dignum, F.P.M. (ed.) ACL 2003. LNCS (LNAI), vol. 2922, pp. 166–182. Springer, Heidelberg (2004)
13. Shanahan, M.: An abductive event calculus planner. Journal of Logic Programming 44(1-3), 207–240 (2000)

14. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law 7, 97–113 (1999)
15. Torroni, P., Yolum, P., Singh, M.P., Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P.: Modelling interactions via commitments and expectations. In: Dignum, V. (ed.) Handbook of Research on MAS: Semantics and Dynamics of Organizational Models, Hershey, Pennsylvania, March 2009, pp. 263–284. IGI Global (2009)
16. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: applying event calculus planning using commitments. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 527–534. ACM Press, New York (2002)