Constraints among Commitments: Regulative Specification of Interaction Protocols

Matteo Baldoni, Cristina Baroglio, and Elisa Marengo

Dipartimento di Informatica — Università degli Studi di Torino
c.so Svizzera 185, I-10149 Torino (Italy)
{`baldoni,baroglio,emarengo`}`@di.unito.it`

Abstract. Interaction protocols play a fundamental role in multi-agent systems. In this work, after analysing the trends that are emerging not only from research on multi-agent interaction protocols but also from neighbouring fields, like research on workflows and business processes, we propose a novel definition of commitment-based interaction protocols, that is characterized by the decoupling of the *constitutive* and the *regulative* specifications and that explicitly foresees a representation of the latter based on *constraints among commitments*. A clear distinction in the two representations has many advantages, that are explained in the paper, mainly residing in a greater openness of multi-agent systems, and an easier re-use of protocols and of action definitions. A language, named 2CL, for writing regulative specifications is also given.

1 Introduction

Open systems are systems made of heterogeneously designed and pre-existing parties that are assembled with some aim, none of them can pursue alone. In order to allow for a fruitful cooperation, the interaction that each agent carries on with the others, in the context of the assembled system, must respect some rules. The term "interaction protocol" refers to a pattern of behavior that allows a set of agents to become a multi-agent system, engaging the expected cooperations with one another. Particularly relevant are *commitment-based protocols*, introduced by Singh [30, 37, 36]. A commitment can be seen as a fluent which can hold in the social state of the system. It represents the fact that a debtor commits to a creditor to bring about some condition. All the agents that interact according to a commitment-based protocol share the semantics of a set of actions, which affect the social state by creating new commitments, canceling commitments, and so forth. The greatest advantages of the commitment-based protocols, w.r.t. to other approaches to interaction, are that they *do not over-constrain* the behavior of the agents by imposing an ordering on the execution of the shared actions, and that by giving a shared meaning to the social actions, they allow working on actual knowledge on what happened (or what is likely to happen), rather than on beliefs about each others' mental state. Nevertheless, commitment protocols *do not yet suit well* all those situations where the evolution of the social state is constrained by laws, preferences, habits, and the like, due to the fact that they do not allow the specification of legal patterns of execution, e.g. a merchant may wish to make clear that shipping will be done only after payment. This kind of constraints makes sense in many practical situations, as noticed also in [32].

In this work, we face this issue by taking on Chopra and Singh's [12] distinction between the *constitutive* and *regulative* specifications of the interaction, deriving from the seminal work of Searle [28]: roughly speaking, constitutive rules give the semantics of actions, while regulative rules rule the flow of execution, thus building new, possibly context-dependant behaviors. In other words, regulative rules capture some important characteristics of how things should be carried on in *specific contexts* of interaction [7]. An actual separation of the constitutive from the regulative specification would bring many advantages in the construction of multi-agent systems. The main one is a direct effect of the obtained *modularity*: an *easier re-use* of actions in different contexts, an *easier customization* on the protocol, an *easier composition* of protocols. For instance, currently interaction protocols are often considered as simply made by a set of shared actions, and this obliges the introduction of additional effects and preconditions in the definition of actions themselves, whenever certain (partial) orderings are desired, e.g. [35]. Thus, *actions result to be strongly dependent from the context* they were thought for. If, instead, the *context* were given by an explicit regulative specification, it would not be necessary to *over-specify* actions, in the spirit of the commitment approach to protocol definition. Actions would be simpler and easier to understand because the constitutive part would correspond to the definition of the action *per se* and not of the action in a context of reference.

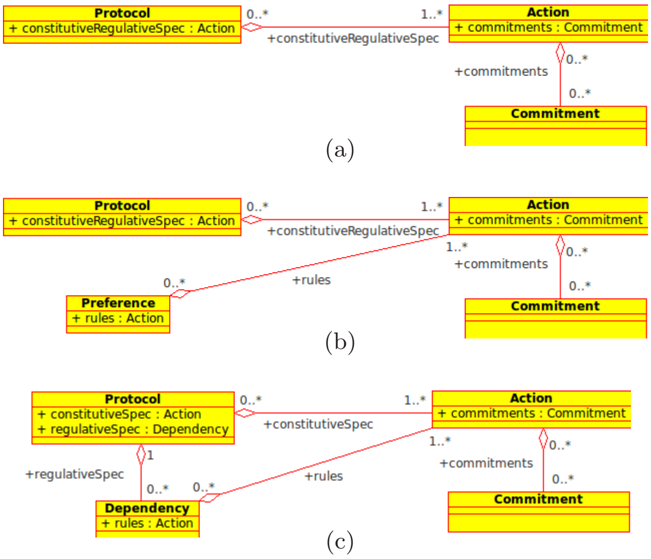
As a consequence, multi-agent systems would gain greater *openness*, *interoperability*, and *modularity* of design. In particular, interoperability would be better supported because it would be possible to verify it w.r.t. specific aspects (e.g. interoperability at the level of actions [12, 10, 13] or at the level of regulation rules [5]). Protocols would be more open in the sense that their modularity would allow designers to easily adapt them to different contexts. Moreover, the decoupling would make it easier for agents to enter a system due to the increased probability of re-using their actions. Agents could also check, individually (against the protocol specification) if they have actions that, when executed individually or according to some pattern, *match* with the constitutive rules, independently from the context of use given by the regulative specification.

In the light of the distinction between constitutive and regulative rules, this work analyzes alternative commitment-based protocol models, that can be found in the literature (namely [12, 23, 19, 20, 36, 35, 27, 22, 3, 31], Section 2), showing that, despite the fact that it is possible recognize various attempts to capture both specifications, these proposals still miss the degree of modularity postulated in [28, 7] and described above. In particular, we show that none allows the specification of both parts (1) *in a decoupled way*, (2) *by means of first-class languages*, (3) *which allow flexible representations* – either one of the two specifications is disregarded or it is too strict or the two representations are to some extent mixed. Section 3, then, proposes a model for commitment-based interaction protocols that separates the constitutive and the regulative parts, and supplies first-class languages for representing both in a flexible way. In particular, for the constitutive specification we adopt [12, 10], while for what concerns regulative specification we propose the use of *constraints among commitments*, and propose a language, named 2CL, that allows the specification of different kinds of such constraints. The language, graphically and in the way the graphical notation is used, inherits from [27, 22] but it is very different from it in its basic principles. In fact, it builds on *commitments* and not on events (actions). Section 4 shows how it is easy to tailor an interaction protocol, expressed by means of 2CL, to different contexts of usage, by producing variants of it working on the regulative specifications only. For the sake of simplicity we chose the well-known Contract Net Protocol (CNP) [15]. In the Conclusions we conclude the comparison with the models in Section 2 showing that the proposed model includes the others as a special case or overcomes their limits.

2 Actions and Protocols: Constitutive and Regulative Specifications

Let us consider commitment-based protocols. Commitments are directed from a debtor to a creditor. The notation $C(x, y, r, p)$ denotes that the agent x commits to an agent y to bring about the condition p when the condition r holds. All commitments are conditional. An unconditional commitment is merely a special case where r equals *true*. Whenever this is the case, we use the short notation $C(x, y, p)$. Agents share a social state that contains commitments and other fluents that are relevant to their interaction. Every agent can affect the social state by executing actions, whose definition is given in terms of modifications to the social state (e.g. adding a new commitment, releasing another agent from some commitment, satisfying a commitment, etc. see [36]). Commitment protocols are interaction patterns given in terms of commitments. Usually a commitment protocol is made of a set of actions (messages), whose semantics is known to – and agreed upon by – all of the participants [36, 37, 10].

There are many definitions for actions in the literature. In *UML* and in the literature about workflows, actions are atomic executions. They are considered to take zero time, and cannot be interrupted, while activities represent more complex behaviors, that may run for a long time, and may be interrupted by events. Most of works on agents adopt, instead, a *precondition-effect* view of actions, independently from the time they take to complete or from possible interruptions. *Preconditions* can be of two kinds: preconditions to the action execution, and preconditions to some effect. The former are fluents that must hold in the social state to make the action executable, the latter are additional conditions that, when holding, allow the production of the specific effect that they control. For instance, in order to pay by credit card it is necessary to own a credit card (precondition to the action). If a credit card owner uses it for paying, the payment will be done only if the card is valid (conditional effect). For example, in [12, 10] actions have no preconditions of any kind, in [11, 21] actions have both preconditions to the executability and conditional effects, while [35] uses only preconditions to the execution of actions. Given these basic notions, let us, now, focus on *regulative rules* and overview the most relevant works in the context of *commitment-based interaction protocols*, in order to compare and discuss the proposed models, which are graphically summarized in Fig. 1 and Fig. 2.



(a)

(b)

(c)

Fig. 1. (a) Chopra and Singh's implementation model: regulative specifications based on actions; (b) Mallya and Singh's model: adding preferences on actions; (c) Singh's dependencies among events.

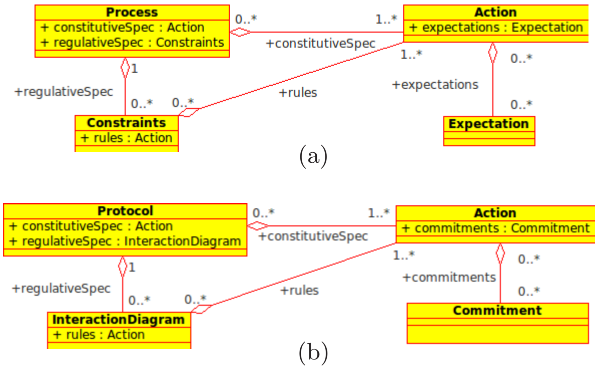
Chopra and Singh. ([12], Fig. 1(a)) Chopra and Singh introduce the distinction between constitutive and regulative specifications in the definition of commitment-based protocols. Each agent is publicly described by the *effects* of the messages they can send, which make the *constitutive specification* of the agent. Such specifications allow agents to agree on the meaning of their communications. Instead, the *regulative specification* rules the data flow among messages. For instance, the constitutive specification of the action *buy* could be the commitment to pay the merchant, while the regulative specification may require that goods are sent only after the payment has been done. In that work (personal communication [9]) and in [11] the regulative specification is based *on the actions themselves*; in particular, the flow is controlled by the *preconditions to the (non-)executability of the actions*. So, in order to impose that sending goods should follow payment, the action *send-goods* should have as a precondition a fluent that is made true as an effect of the action *pay*.

This solution (which is adopted also by other works, like [20, 36, 37, 10, 35]) is characterized by a *strong localization* of the regulative specification. Both the constitutive and the regulative specifications are indistinguishable (being both based on actions, see Fig. 1(a)). *The problem is that by doing so the definition of an action becomes dependant on the protocol where it is used.* This limits the openness of the system and in particular complicates the re-use of software (the agents' actions). Actions, in fact, are defined not only for what concerns their effects (constitutive specification) but also taking into account their context of use. When changing context (protocol), the regulative specification inside the actions is to be updated or new, specific actions are to be defined. Analogously, when adding a new action, it is necessary to enrich it with the correct regulative specification. In our view, a greater decoupling between the actions and the regulative specification would have the advantage of facilitating the re-use of actions because it would allow the avoidance of the over-specification that is necessary to impose an ordering among actions.

Preferences and dependencies. ([23], Fig. 1(b) and [31], Fig. 1(c), respectively) Mallya and Singh [23] propose to order the possible executions according to a set of preferences that take into account the policies of the various parties. No execution is strictly forbidden but a preference criterion is specified. Differently than above, here the constitutive specification is given in terms of commitments but the preference rules are given in terms of actions. Preferences do not precisely correspond to regulatives rules because they specify selection policies, rather than constraining the execution flow, nevertheless, giving them in terms of actions makes the specification less flexible and less easily adaptable or open. The same limits, Fig. 1(c), can be ascribed to the work to which [23] is inspired, i.e. [31], although in this work it is possible to recognize the introduction of a regulative specification, based on the *before* relation applied to events.

ConDec. ([27, 26, 8, 22], Fig. 2(a)) Pesic and van der Aalst propose an approach that is radically opposite to the one by Chopra and Singh [27]. In this approach, which totally lacks of a constitutive component (and does not build on commitments nor is set in the agents framework), the declarative language ConDec is proposed for representing business processes which, though not exactly interaction protocols, specify the expected behavior of a set of interacting parties by constraining the execution of their tasks. The regulative rules are a first-class element of the protocol which are given by means of an ad hoc *declarative* language. They are not local to single actions as in Fig. 1(a), rather they are constraints that rule the flow of activity execution (activity in UML sense). In [26, 8, 22], the authors use this approach to specify interaction protocols and service choreographies. To this aim, they integrate ConDec with SCIFF thus giving a semantics to actions that is based on *expectations*.

Still, however, in these proposals there is a too tight connection between the regulative rules and actions because such rules define temporal constraints over



(a)

(b)

Fig. 2. (a) ConDec model: regulative specification given by means of constraints on actions, an extension supplies an expectation-based semantics for actions; (b) Fornara and Colombetti's model: regulative specification given by interaction diagrams defined on actions.

actions (events), see Fig. 2(a). This, in our opinion, clashes with the openness of multi-agent systems. Let us explain our view with an example. Let us suppose that payment should occur before sending the goods, and that the protocol foresees the actions *pay-by-credit-card* and *send-goods*. Then, it will specify that *pay-by-credit-card* must occur before *send-goods*. Now, if a client arrives which can pay cash, it will not be in condition to take part to the interaction unless the regulative specification is changed by adding a rule that says that paying cash should occur before sending the goods. This should be done even though the action has the same semantics of *pay-by-credit-card* in terms of commitments. The need of modifying the regulative specification (even in the case when actions have the same semantics!), gives *an undesired rigidity to the protocol*. Problems arise also in the case an agent can execute a sequence of actions which *altogether* implement one of those foreseen by the protocol. The problem is that the regulative specification is given in terms of *actions*, so, when changing the actions names we need to change regulative specifications as well. It is also easy to make mistakes by forgetting to update the regulative part when a new action is changed or when its semantics is changed.

Fornara and Colombetti. ([16, 19, 18], Fig. 2(b)) Fornara and Colombetti define a commitment-based semantics for the speech acts of agent communication languages, like FIPA, and then use *interaction diagrams* to define agent interaction protocols. In this proposal, the social actions are represented by the speech acts and the constitutive specification is given in terms of commitments. The choice of relying on interaction diagrams is, however, very strong because it forces the ordering of action executions, loosing, in our opinion the flexibility aimed at by the adoption of commitments.

Summary. The distinction between a regulative and a constitutive specification is surely interesting but the current proposals still show some limits in the realization of this model, each with its pros and cons. Fornara and Colombetti propose a too rigid model: the use of interaction diagrams conflicts with the desirable flexibility of commitments. In this respect, ConDec's use of constraints is better: the declarative approach that is proposed is aligned with the declarative nature of commitments. The problem is that constraints are defined in terms of performing actions rather than on bringing about conditions. Also Chopra and Singh [9] propose an implementation where the regulative specification is given on top of actions themselves. Again, while commitments are given on conditions and not on the actions that should bring them about, constraints are posed on the action execution, with the result that modularity is not obtained. The same holds for [35, 20, 36, 37].

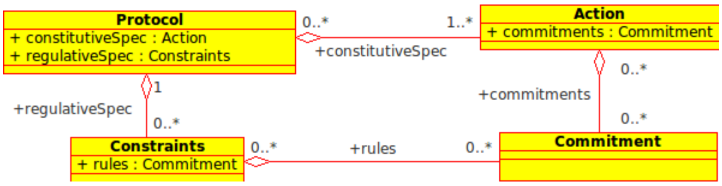


Fig. 3. Our proposal: decoupling between constitutive (actions) and regulative (constraints) specifications.

Our proposal aims at overcoming the listed limits. As in [23, 19] we propose the use of commitments to give constitutive specifications. As in [27, 26] we propose the use of a declarative language, 2CL, for capturing constraints that rule the execution flow. The difference is that in our proposal such constraints relate commitments and not actions (see Fig. 3). Doing so allows us the achievement of a greater modularity, which brings along the mentioned advantages: allowing an easier re-use of actions in different contexts, allowing an easier re-use of protocols with different actors, simplifying the modification of protocols, greater openness, better support to interoperability checks. The next sections illustrates our proposal for the representation of regulative specifications.

3 Commitment Protocols: A Decoupled Approach

In this work, we propose an approach to the definition of commitment-based interaction protocols which includes a *constitutive* specification, that defines the meaning of actions for all the agents in the system, and a *regulative* specification, which constrains the possible evolutions of the social state. Both are defined based on *commitments*.

Definition 1 (Interaction protocol). *An interaction protocol* P *is a tuple* $\langle Ro, F, A, C\rangle$, *where* Ro *is a set of roles, identifying the interacting parties,* F *is* *a set of fluents (including commitments) that can occur in the social state, A is a set of actions, and C is a set of constraints.*

The set of social actions A, defined on F and on *Ro*, forms the *constitutive specification* of the protocol, while the set of constraints C, defined on F and on *Ro* too, forms the *regulative specification* of the protocol.

The *constitutive specification* of an action, similarly to [10], defines its meaning in terms of how it affects the social state by adding or removing fluents or by performing operations on the commitments (the usual create, discharge, release, delete, etc., see [29, 37]). The constitutive specification follows the grammar:

$A \rightarrow (\mathit{Action}\ \mathbf{means}\ \mathit{Operation})^{+}$
$\mathit{Action} \rightarrow \mathit{protocolAction}([\mathit{paramList}])$
$\mathit{Operation} \rightarrow \mathit{Op}(\mathit{commitment}) \mid fact \mid \mathit{Operation} \wedge \mathit{Operation}$
$\mathit{Op} \rightarrow \textsc{create} \mid \textsc{delete} \mid \textsc{discharge} \mid \textsc{release} \mid \textsc{delegate} \mid \textsc{assign}$

where *protocolAction* is the name of an interactive action of the protocol; *paramList* denotes the possible parameter list of the action; *Op* is one of the operations on commitments; *commitment* is a commitment of form $C(x, y, r, p)$, as specified in Section 2 (see also [10, page 49]), where x and y are roles in *Ro* and r and p are formulas in disjunctive normal form of propositional literals in F; and *fact* is a literal, i.e. a positive or negative proposition that does not concern commitments and which contributes to the social state (they are the conditions that are brought about). For instance, the action *cfp* of the contract net protocol (which is used as an example below) is given in this way: $cfp(i, p)$ **means** $\textsc{create}(C(i, p, assigned\_task))$, i.e. its effects is to add to the social state the commitment $C(i, p, assigned\_task)$ by which the initiator (role i) commits to a participant (role p) to assign a task of interest to someone. Not necessarily the task will, in the end, be assigned to the p at issue; if many participants propose to solve the task, the choice will depend on the decision criteria implemented by the specific initiator, that are not modeled by the protocol.

In order to represent the *regulative specification*, we propose a *constraint-based representation* following the grammar:

$C \rightarrow (\mathit{Disj}\ op\ \mathit{Disj})^{+}$
$\mathit{Disj} \rightarrow \mathit{Conj} \vee \mathit{Disj} \mid \mathit{Conj}$
$\mathit{Conj} \rightarrow fluent \wedge \mathit{Conj} \mid fluent$

C, see Def. 1, is a set of constraints of the form A *op* B, where A and B are formulas of fluents in disjunctive normal form and *op* is one of the operators in Table 1; *fluent* can be either a commitment or a fact. Such constraints rule the evolution of the social state by imposing specific patterns on how states can progress. For instance, $C(i, p, assign\_task) \mathrel{-\!\!\triangleright\!\bullet} (refused\_task \vee C(p, i, solve\_task))$ expresses the fact that a participant cannot refuse a task nor it is allowed to commit to solve it before the initiator has taken a commitment, stating its intention to assign the task to that participant. Notice that the constraint *does not* specify *which* actions should bring these conditions about, in fact, constraints do not rule the occurrence of events. Moreover, the declarative nature of the specification adds flexibility w.r.t. an algorithmic specification, in fact, while the latter specifies *all* the allowed evolutions, declarative constraints allow *any* evolution that respects the relations involving the specified fluents.

Relation	*Positive*	*LTL meaning*	*Negative*	*LTL meaning*
Correlation	$a \bullet\!\!- b$	$\Diamond a \supset \Diamond b$	$a \bullet\!\!\not{-}\, b$	$\Diamond a \supset \neg\Diamond b$
Co-existence	$a \bullet\!\!-\!\!\bullet b$	$a \bullet\!\!- b \wedge b \bullet\!\!- a$	$a \bullet\!\!\not{-}\!\!\bullet b$	$a \bullet\!\!\not{-}\, b \wedge b \bullet\!\!\not{-}\, a$
Response	$a \bullet\!\!-\!\!\triangleright b$	$\Box(a \supset \Diamond b)$	$a \bullet\!\!\not{-}\!\!\triangleright b$	$\Box(a \supset \neg\Diamond b)$
Before	$a -\!\!\triangleright\!\!\bullet b$	$\neg b \mathsf{U} a$	$a \not{-}\!\!\triangleright\!\!\bullet b$	$\neg a \mathsf{U} b$
Cause	$a \bullet\!\!-\!\!\triangleright\!\!\bullet b$	$a \bullet\!\!-\!\!\triangleright b \wedge a -\!\!\triangleright\!\!\bullet b$	$a \bullet\!\!\not{-}\!\!\triangleright\!\!\bullet b$	$a \bullet\!\!\not{-}\!\!\triangleright b \wedge a \not{-}\!\!\triangleright\!\!\bullet b$
Premise	$a \triangleright\!\!\triangleright\!\!- b$	$\Box(\bigcirc b \supset a)$	$a \triangleright\!\!\triangleright\!\!\not{-}\, b$	$\Box(\bigcirc b \supset \neg a)$
Immediate response	$a -\!\!\triangleright\!\!\triangleright b$	$\Box(a \supset \bigcirc b)$	$a \not{-}\!\!\triangleright\!\!\triangleright b$	$\Box(a \supset \bigcirc\neg b)$

Table 1. 2CL operators and their semantics in LTL.

We named the language for representing the regulative specification 2CL(the acronym stands for "Constraints among Commitments Language") . The names of the operators and in the graphical format, used in Section 4, are inspired by ConDec [27]. We remark again that the main difference is that constraints are defined over commitments and facts, while in ConDec they are defined on actions. In order to allow the application of reasoning techniques, e.g. to check if the on-going interaction is respecting the protocol, to build sequences of actions that respect the protocol, or to verify properties of the system, it is necessary to give the operators a semantics that can be reasoned about. To this aim, in this work we use *linear temporal logic* (LTL, [14]), which includes temporal operators such as next-time ($\bigcirc$), eventually ($\Diamond$), always ($\Box$), weak until (U). Let us describe the various operators. For simplicity the description are given on single fluents rather than formulas.

Correlation: this operator captures the fact that in an execution where a occurs, also b occurs but there is no temporal relation between the two. Its negation means that if a occurs in some execution, b must not occur.

Co-existence: the mutual correlation between a and b. Its negation captures the mutual exclusion of a and b. Notice that in LTL the semantics of negated co-existence is equivalent to the semantics of negated correlation.

Response: this is a temporal relation, stating that if a occurs b must hold at least once afterwards (or in the same state). It does not matter if b already held before a. The negation states that if a holds, b cannot hold in the same state or after.

Before: this a temporal relation, stating that b cannot hold until a becomes true. Afterwards, it is not necessary that b becomes true. The negation of $a -\!\!\triangleright\!\!\bullet b$ is equivalent to $b -\!\!\triangleright\!\!\bullet a$.

Cause: this operator states that if a occurs, after b must occur at least once and b cannot occur before a. The negation states that if a occurs, b cannot follow it and if b occurs, a is not allowed to occurre before.

Premise: is a stronger temporal relation concerning *subsequent* states, stating that a must hold in all the states immediately preceding one state in which b holds. The negation states that a must never hold in a state that immediately precedes one where b holds.

Immediate Response: it concerns *subsequent* states, stating that b must occur in all the states immediately following a state where a occurs. The negation states that b does not have to hold in the states immediately following a state where b holds.

Notice that the negated operators semantics (column 5) not always corresponds to the negation of the semantics of the corresponding positive operator (column 3). This is due to the intention of capturing the intuitive meaning of negations. We show this need by means of a couple of examples. For what concerns correlation, the negation of the formula in column 3, which is $\Diamond a \land \neg \Diamond b$, is too strong because it says that a must hold sooner or later while b cannot hold. What we mean by negated coexistence, instead, that *if a becomes true* then b must not occur in the execution. For completeness, the semantics of negated correlation is not equivalent to the semantics of $a \bullet\!\!- \neg b$.

For what concerns immediate response, by negating the semantics in column 3 we obtain $\Diamond(\bigcirc b \land \neg a)$ which says that b occurs in some state and a does not occur in the previous state. Instead, the intended meaning of the negation is that a does not have to hold in the states that precede those in which b holds (but b does not necessarily have to hold). Analogous considerations can be drawn for the other operators. The choice of sticking to the intuitive semantics of the operators is done to give the user only *seven* basic operators. Had we defined the negated operators semantics by negating the semantics of the positive operators, we would have given the user *forteen* different operators.

4 Tailoring Protocols to different needs

In this section, we show the use of the proposed model by, first, representing the well-known Contract Net Protocol (CNP for short) [15] and, then, by showing how easy it is to produce variants by playing with its regulative specification, separately from the constitutive specification of its actions. Briefly, CNP includes two roles, the initiator (i in the following) and a participant (p). The initiator calls for proposals. The participant may send a proposal or refuse to do it. When a proposal is received, the initiator may either reject or accept it. Notice that, for the sake of simplicity, we do not model the exchange of information concerning the proposal itself but only the interaction concerning the task assignment and solution. We report the CNP as represented according to our proposal, by giving its constitutive specification followed by its regulative specification.

Constitutive specification of CNP. The actions of CNP, as expressed according to the grammar in Section 3, are:

(a) $cfp(i, p)$ **means** $\textsc{create}(C(i, p, assigned\_task))$
(b) $propose(p, i)$ **means** $\textsc{create}(C(p, i, solved\_task))$
(c) $refuse(p, i)$ **means** $refused\_task \wedge \textsc{release}(C(i, p, assigned\_task))$
(d) $accept(i, p)$ **means** $assigned\_task$
(e) $reject(i, p)$ **means** $rejected\_proposal \wedge \textsc{delete}(C(i, p, assigned\_task)) \wedge$
$\textsc{release}(C(p, i, solved\_task))$
(f) $inform\_done(p, i)$ **means** $solved\_task$
(g) $failure(p, i)$ **means** $failed\_task \wedge \textsc{delete}(C(p, i, solved\_task))$

Since such definitions are quite straightforward, we get into the details of just a couple of them. The effect of the action *cfp* is to create the commitment $C(i, p, assigned\_task)$. Intuitively, this commitment states the resolution of the initiator to assign a task to a participant because it needs someone to solve it. This does not mean that, at the end, the task will be assigned to *that* participant. Indeed, during the execution the participant may refuse to solve the task or the initiator may reject its proposal because, for example, it is not convenient. The action $refuse(p, i)$ (the participant refuses to solve a task), instead, has, as effect, the action $\textsc{release}(C(i, p, assigned\_task))$, by which the participant releases the initiator from the commitment of assigning a task to it, and the fact $refused\_task$, whose meaning is clear.

Regulative specification of CNP. The regulative rules of CNP, as expressed according to the grammar in Section 3, are:

c1: $C(i, p, assigned\_task) \bullet\!\!-\!\!\triangleright\!\!\bullet\, C(p, i, solved\_task)$ XOR $refused\_task$
c2: $C(p, i, solved\_task) \bullet\!\!-\!\!\triangleright\!\!\bullet\, rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \bullet\!\!-\!\!\triangleright\!\!\bullet\, solved\_task$ XOR $failed$

Fig. 4 reports them as a graph, whose nodes (the rectangles) contain fluents that should be in the social state at some point of the execution, while the arrows are operators from Table 1. The initiator declares its intention to assign a task (node $n1$, $C(i, p, assigned\_task)$). If this happens, afterwards the participant takes its decision and alternatively refuses or states its intention to solve the task. This is represented by the fact that the node $n1$ is connected to the nodes

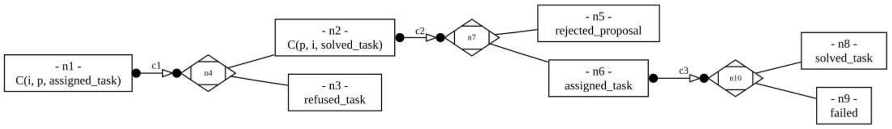


Fig. 4. Regulative specification of the Contract Net Protocol.

$n2$ ($C(p, i, solved\_task)$), and $n3$ ($refused\_task$): $n2$ and $n3$ are alternative evolutions of the social state after $n1$. The connector $n4$ denotes the *exclusive or* of the two. It is a graphical simplification of the and-or formula implementing the "exclusive or". The arrow used (of the kind $a \bullet\!\!\!\rightarrow\!\!\bullet\; b$) represents the fact when the initiator has a task to assign, the participant it is interacting with necessarily has to either refuse the task or take the commitment to solve it. It is not obliged to do it as the next step of its execution but sooner or later it must take one of the two ways. The specification foresees that the participant cannot take the initiative of proposing to solve a task (or of refusing to do something) if the initiator has not declared that there is a task to solve. This is the intuitive meaning of the circles at the two sides of the arrow $c1$.

Notice that we have not mentioned which actions should be executed in order to change the social state. Actually, we do not care. Any action, whose effects are compatible with the schema of evolution of the social state reported above is feasible. In the same way it is not necessary, in commitment protocols, to say which action to take in order to satisfy a commitment. Moreover, the transition from one state to one of its next states (according to the description given by regulative specification) may actually require the application of many actions (not necessarily one). The regulative specification does *not* give any *procedure* for achieving the social state change, that it captures. In fact, constraints on the evolution of the social state are independent from the actions that are used by the agents. Both, however, are specified on top of the fluents in the social state.

If the interaction continues because the participant has proposed to solve the task, the initiator must either reject the proposal or accept it and assign the task to the participant, which, in this case, will try to solve the task and give back to the initiator an answer (the solution or the information that it has failed). The arrows in the graph between nodes $n2$ and the alternative between $n5$ and $n6$, on a side, and between $n6$ and the alternative between $n8$ and $n9$ are again of the kind $\bullet\!\!\!\rightarrow\!\!\bullet$ (causality operator).

4.1 Tailoring the Contract Net Protocol

Let us now show the versatility of the proposed representation by showing how a designer can easily modify the specification of the contract net protocol given above in order to build new protocols which adapt to different conditions. All the variantions are produced by working exclusively on the regulative specification without modifying the actions. Of course, it is possible to do the opposite or to modify both parts if necessary.

Lazy and zealous participant. (Fig. 5(a), Fig. 5(b)) Let us consider, for a start, two simple variants of the allowed behavior, obtained by changing a single arrow with another operator from Table 1. For instance if, see Fig. 5(a) (only the modified part of the CNP regulative specification is reported), we use a *before* relation ($\rightarrow\!\!\bullet$), the participant *would not be obliged* to answer (it is allowed to have a lazy behavior). In constraints the whole variant is:

c1: $C(i, p, assigned\_task) \rightarrow\!\!\bullet\; C(p, i, solved\_task)$ XOR $refused\_task$

c2: $C(p, i, solved\_task) \; {\bullet\!\!-\!\!\triangleright\!\!\bullet} \; rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \; {\bullet\!\!-\!\!\triangleright\!\!\bullet} \; solved\_task$ XOR $failed$

Instead, see Fig. 5(b), if a *response* (${\bullet\!\!-\!\!\triangleright}$) is used, the participant *can*, for instance, *also take the initiative* to volunteer to solve a task even though the initiator has not made any request (zealous participant):

c1: $C(i, p, assigned\_task) \; {\bullet\!\!-\!\!\triangleright} \; C(p, i, solved\_task)$ XOR $refused\_task$
c2: $C(p, i, solved\_task) \; {\bullet\!\!-\!\!\triangleright\!\!\bullet} \; rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \; {\bullet\!\!-\!\!\triangleright\!\!\bullet} \; solved\_task$ XOR $failed$

These two variants correspond to protocols that differ from CNP but that can easily be obtained by working at the level of constraints among commitments.
Contract Net with Immediate Answer. (Fig. 5(c)) A stricter regulative specification, produced for a context, which differs from the original contract net one (Fig. 4) only for what concerns the constraint $c2$, is reported in Fig. 5(c). On the whole, the regulative specification of the protocol states that the participant is expected to answer (as in the original CNP) and, when the answer is the commitment to solve the task, the initiator will reply without any delay. This amounts to require that the initiator evaluates the proposal and gives the outcome of its evaluation back to the participant *immediately*. The specification is as follows:

c1: $C(i, p, assigned\_task) \; {\bullet\!\!-\!\!\triangleright\!\!\bullet} \; C(p, i, solved\_task)$ XOR $refused\_task$
c2: $C(p, i, solved\_task) \; {-\!\!\triangleright\!\!\!\triangleright} \; rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \; {\bullet\!\!-\!\!\triangleright\!\!\bullet} \; solved\_task$ XOR $failed$

Call for Bids. (Fig. 5(d)) The next context that we consider is a *call for bids*, where an initiator publishes an open call, e.g. in an official gazette, that does not require the subscribers to the gazette to answer. Fig. 5(d) shows the new protocol: the fact that the participant is not obliged to send a bid is captured by the constraint $c1$, which is a *before* (${-\!\!\triangleright\!\!\bullet}$) instead of being a *cause* (${\bullet\!\!-\!\!\triangleright\!\!\bullet}$, in Fig. 4). We have further modified the CNP by changing the constraint $c2$ in Fig. 4, which is now an *immediate response* (${-\!\!\triangleright\!\!\!\triangleright}$), in this way the initiator is obliged to answer immediately to any participant which sends a bid, either rejecting the proposal or assigning the task. The new specification in rules is:

c1: $C(i, p, assigned\_task) \; {-\!\!\triangleright\!\!\bullet} \; C(p, i, solved\_task)$ XOR $refused\_task$
c2: $C(p, i, solved\_task) \; {-\!\!\triangleright\!\!\!\triangleright} \; rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \; {\bullet\!\!-\!\!\triangleright\!\!\bullet} \; solved\_task$ XOR $failed$

In the same context of the "Call for Bids" protocol, a designer may need to express the fact that the participant can notify a failure in the task solution also in the case in which the task has not been assigned to it yet but, for some reason, it has found out that it has become impossible for it to proceed with the solution, in case the task is assigned to it. Instead, it is not allowed to communicate the solution until the task is assigned to it. The new protocol can be obtained by modifying the regulative specification of Fig. 5(d) as in Fig. 5(e). In rules:

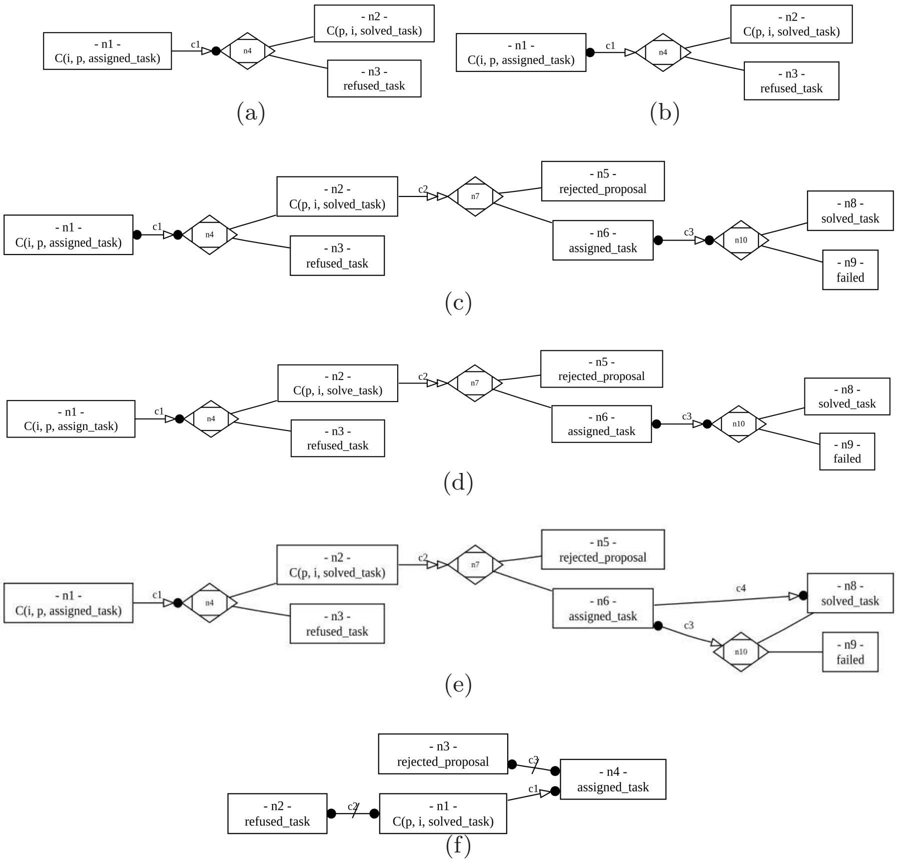


Fig. 5. (a) Lazy Participant; (b) Zealus Participant; (c) Contract Net with Immediate Answer; (d) Call for Bid; (e) Call for Bids with Anticipated Failure; (f) Soft Call for Bids.

c1: $C(i, p, assigned\_task) \relbar\!\triangleright\!\bullet\; C(p, i, solved\_task)$ XOR $refused\_task$
c2: $C(p, i, solved\_task) \relbar\!\triangleright\!\!\triangleright\; rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \;\bullet\!\relbar\!\triangleright\; solved\_task$ XOR $failed$
c4: $assigned\_task \relbar\!\triangleright\!\bullet\; solved\_task$

The changes concern the constraints after node $n6$. In the new version, instead of having simply a *cause* constraint, we have a *response* ($\bullet\!\relbar\!\triangleright$). Response is a softer constraint because it does not forbid to the alternatives specified by $n10$ to hold before *assigned_task*. For this reason, in order to enforce that the solution is communicated only after the assignment, another constraint is to be added ($c4$). In this way, failure can be notified at any moment.

Soft Call for Bids. (Fig. 5(f)) The last example is a very soft interaction protocol that, differently than the previous ones, expresses just a few regulative constraints, leaving a much greater freedom of behavior to the initiator and to the participant.

c1: $C(p, i, solved\_task) \mathrel{-\!\!\triangleright\!\bullet} assigned\_task$
c2: $refused\_task \mathrel{\bullet\!\!\not\!\!-\!\!\bullet} C(p, i, solved\_task)$
c3: $rejected\_proposal \mathrel{\bullet\!\!\not\!\!-\!\!\bullet} assigned\_task$

This example also shows the use of *negative* constraints. The only constraint that is imposed on the evolution of the social state is that the assignment of a task cannot be done if the participant has not committed to solve the task. Moreover, there are two negative constraints (of the kind $\bullet\!\!\not\!\!-\!\!\bullet$) stating that the rejection of a proposal is mutually exclusive to its assignment ($c3$), and that the refusal of a task is mutually exclusive to the commitment to solve it ($c2$). So, for instance, it is possible for the participant to express its intention to solve a task, for which no call has been made and it is also possible for it to give a solution before any assignment of the task has been made to it. On the other hand, the initiator can ignore the participant even though it has committed to solve the task by avoiding to answer to it. It can call for proposals even if it already has a commitment by the participant, and it can reject a participant even though it has not made any proposal. It is not even necessary that the initiator commits to assign the task. In rules:

5 Conclusion and future work

Constitutive and regulative specifications have been recognized as fundamental components of the interaction based on communication starting from Searle [28, 7], and including various authors in the Multi-Agent community, e.g. [12, 10, 6]. In this paper we have presented a model of commitment-based interaction protocols that includes an explicit representation of both constitutive and regulative specifications. From a graphical point of view, the language 2CL is inspired to [27, 4]. The semantics of the operators is based on linear temporal logic due to the fact that this logic is well-known and simple and opens the way to possible integrations with model checkers like SPIN. We mean, however, to study alternatives offered, for instance, by CTL*.

The proposal includes as special cases some of the representation models that are discussed in Section 2. Specifically, we can model the proposal by Chopra and Singh [12] as well as the models adopted in [20, 37, 35] which follow the same principles, by introducing for each action a fluent that is univocally associated to it, as an effect of the action, and, then, to define constraints (typically of kind *premise*, $\triangleright\!\triangleright\!\!-$) among these fluents.

In case the designer wishes to specify strict sequences of action executions, as it may happen in [17, 19, 16], our proposal allows to do it in a straightforward way. One can introduce for each action a fluent that is univocally associated to it, as in the above case. As a difference, the designer must use the *immediate response* operator ($-\!\!\triangleright\!\triangleright$) to create sequences, which can further be combined with other constraints.

Last but not the least, the proposal overcomes the limits of those in [27, 31, 26, 8, 22] because the regulative specification rules the evolution of the social state and not the execution of actions/events. In case the designer wishes to constrain the execution of specific actions, again he/she can introduce a fluent for each action, univocally produced as an action effect.

An approach similar to commitment-based protocols is the one introduced in [3], where *expectation*-based protocols are presented. Expectations concern events expected to happen (or not to happen) and can be associated to time points. Protocols are specified by constraining the times at which events occur. As for the previous works, the limit of this approach is that it works directly on events (i.e. actions); by constraining actions the approach lacks the openness discussed in the Introduction and in the discussion about ConDec in Section 2. On the other hand, our proposal does not handle time explicitly so we cannot yet represent and handle timeouts and also compensation mechanisms. Our intention with this paper was, however, to present the idea of an explicit, declarative, and decoupled representation of both the constitutive and the regulative specifications. We mean to tackle also issues concerning time, faults and compensation, like in [33] (where commitments are implemented by means of expectations), in future work.

The adaptation of commitment-based protocols to different contexts of usage has been tackled in [11]. The authors show how a declarative approach is particularly suitable to this aim. Our proposal is set along this line. In fact, not only the constitutive rules are given in a declarative way but also the regulative specification is made of declarative constraints and it is possible to contextualize it by adding or removing constraints. The advantage w.r.t. [11], however, is the modularity of the two specifications discussed along the paper.

The work in [25] contains a comparison of various approaches to interaction protocols, including but not limited to commitment-based protocols. Specifically, also normative systems, algebraic-operational approaches (like $\mathcal{RASA}$ [24]), and Petri nets are considered. The comparison is done along many directions. The authors confirm our opinion that declarative approaches (like commitment-based ones) are very flexible. However, they claim that they are less readable (and sometimes more verbose) than algorithmic approaches. To support this consideration they cite some of the major existing tools for the designer (like AgentUML), which are algorithmic. For verbosity, they cite the CNP representation in [3] which consists of seventeen rules. We underline that our regulative representation of CNP consists instead of *three* rules only. The constitutive specification is made of *seven* rules (because there are seven actions). For what concerns commitment protocols, the difficulty in reading declarative specifications is, in our opinion, due to the lack of separation between the constitutive and the regulative specifications that many approaches show. Moreover, as [34] notices, there is a lack of graphical intuitive representations oriented to designers. We have tried to overcome these problems by decoupling the regulative and the constitutive specifications and by giving a graphical representation. This representation has the advantage of giving the *perception of a flow* in the execution, remaining however at a *what* rather than at a *how* level (*no-flow in-flow*). This representation also supports the compositionality of the protocols. In fact, to put it simply, in order to produce a new protocol starting from existing ones, it is sufficient to draw together the sets of constraints of interest and produce a bigger graph without any effort. Protocols can, then, be designed bottom-up. This aspects will further be studied as future work.

Acknowledgements

The authors would like to thank the reviewers for the helpful comments. This research has partially been funded by "Regione Piemonte" through the project ICT4LAW.

References

1. *Proc. of the 2nd Multi-Agent Logics, Languages, and Organisations Federated Workshops*, volume 494 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.
2. *Declarative Agent Languages and Technologies VII, 7th Int. Workshop, DALT 2009*, volume 5948 of *Lecture Notes in Computer Science*. Springer, 2010.
3. M. Alberti, D. Daolio, P. Torroni, M. Gavanelli, E. Lamma, and P. Mello. Specification and verification of agent interaction protocols in a logic-based system. In *Proc. of the SAC 2004*, pages 72–78. ACM, 2004.
4. M. Baldoni, C. Baroglio, I. Brunkhorst, N. Henze, E. Marengo, and V. Patti. Constraint Modeling for Curriculum Planning and Validation. *Int. J. of Interactive Learning Environments*, 2009.
5. M. Baldoni, C. Baroglio, and E. Marengo. Commitment-based Protocols with Behavioral Rules and Correctness Properties of MAS. In *Proc. of International Workshop on Declarative Agent Languages and Technologies, DALT 2010*, Toronto, Canada, May 2010.
6. G. Boella and L. W. N. van der Torre. Regulative and constitutive norms in normative multiagent systems. In *Proc. of KR*, pages 255–266. AAAI Press, 2004.
7. C. Cherry. Regulative rules and constitutive rules. *The Philosophical Quarterly*, 23(93):301–315, 1973.
8. F. Chesani, P. Mello, M. Montali, and P. Torroni. Verifying a-priori the composition of declarative specified services. In *MALLOW* [1].
9. A. K. Chopra. Personal communication. December 2009.
10. A.K. Chopra. *Commitment Alignment: Semantics, Patterns, and Decision Procedures for Distributed Computing*. PhD thesis, North Carolina State University, Raleigh, NC, 2009.
11. A.K. Chopra and M. P. Singh. Contextualizing commitment protocol. In *Proc. of AAMAS'06*, pages 1345–1352. ACM, 2006.
12. A.K. Chopra and M. P. Singh. Constitutive interoperability. In *Proc. of AAMAS'08*, pages 797–804, 2008.
13. A.K. Chopra and M. P. Singh. Multiagent commitment alignment. In *Proc. of AAMAS'09*, pages 937–944, 2009.
14. E. A. Emerson. Temporal and Modal Logic. In *Handbook of Theoretical Computer Science*, volume B, pages 997–1072. Elsevier, 1990.

15. Foundation for Intelligent Physical Agents. *FIPA Contract Net Interaction Protocol Specification*, December 2002.
16. N. Fornara. *Interaction and Communication among Autonomous Agents in Multiagent Systems.* PhD thesis, Università della Svizzera italiana, Facoltà di Scienze della Comunicazione, June 2003.
17. N. Fornara and M. Colombetti. Defining interaction protocols using a commitment-based agent communication language. In *Proc. of AAMAS'03*, pages 520–527, 2003.
18. N. Fornara and M. Colombetti. Protocol Specification Using a Commitment Based ACL. In *ACL 2003*, volume 2922 of *LNCS*, pages 108–127. Springer, 2003.
19. N. Fornara and M. Colombetti. A Commitment-Based Approach To Agent Communication. *Applied Artificial Intelligence*, 18(9-10):853–866, 2004.
20. L. Giordano, A. Martelli, and C. Schwind. Specifying and verifying interaction protocols in a temporal action logic. *J. Applied Logic*, 5(2):214–234, 2007.
21. Ö. Kafali and P. Yolum. Detecting exceptions in commitment protocols: Discovering hidden states. In *MALLOW* [1].
22. Montali M., M. Pesic, W.M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari. Declarative specification and verification of service choreographies. *ACM Transactions on the Web*, 2009.
23. A. U. Mallya and M. P. Singh. Introducing preferences into commitment protocols. In *AC*, volume 3859 of *LNCS*, pages 136–149. Springer, 2006.
24. T. Miller and P. McBurney. Annotation and matching of first-class agent interaction protocols. In *Proc. of the 7th AAMAS*, pages 805–812, 2008.
25. T. Miller and J. McGinnis. Amongst first-class protocols. In *Proc. of Eng. Societies in the Agents World VIII*, volume 4995 of *LNCS*, pages 208–223. Springer, 2008.
26. M. Montali. *Specification and Verification of Declarative Open Interaction Models - A Logic-based framework.* PhD thesis, University of Bologna, 2009.
27. M. Pesic and W. M. P. van der Aalst. A Declarative Approach for Flexible Business Processes Management. In *Proc. of. Business Process Management Workshops*, volume 4103 of *LNCS*, pages 169–180. Springer, 2006.
28. J. Searle. *Speech Acts.* Cambridge University Press, 1969.
29. M. P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
30. M. P. Singh. A social semantics for agent communication languages. In F. Dignum and M. Greaves, editors, *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 31–45. Springer, 2000.
31. M. P. Singh. Distributed enactment of multiagent workflows: temporal logic for web service composition. In *AAMAS*, pages 907–914. ACM, 2003.
32. M. P. Singh and A. K. Chopra. Correctness properties for multiagent systems. In *DALT* [2], pages 192–207.
33. P. Torroni, F. Chesani, P. Mello, and M. Montali. Social commitments in time: Satisfied or compensated. In *DALT* [2], pages 228–243.
34. W. M. P. van der Aalst, M. Dumas, A. H. M. ter Hofstede, N. Russell, H. M. W. Verbeek, and P. Wohed. Life after BPEL? In *Proc. of WS-FM*, volume 3670 of *LNCS*, pages 35–50, 2005.
35. M. Winikoff, W. Liu, and J. Harland. Enhancing commitment machines. In *Proc. of DALT*, volume 3476 of *LNCS*, pages 198–220, 2004.
36. P. Yolum and M. P. Singh. Commitment machines. In *Proc. of ATAL*, volume 2333 of *LNCS*, pages 235–247. Springer, 2001.
37. P. Yolum and M. P. Singh. Designing and executing protocols using the event calculus. In *Agents*, pages 27–28, 2001.