1

Modelling Interaction Protocols as Modular and Reusable 1st Class Objects

Esteban León-Soto

German Research Center for Artificial Intelligence, DFKI GmbH
Multi-agents and Simulated Reality Department,
Campus. D3.2, 66123 Saarbrücken, Germany
esteban.leon@dfki.de
http://www.dfki.de

Summary. Interoperability involves solutions at different levels, from concrete data representation to coordination of actions. These solutions are even more difficult to handle when the systems being integrated expect to remain autonomous. This is frequently the case in business processes between different organizations. In order to work together several technical and organizational issues have to be solved. One of the most important aspects is coordination and efficient communication. Interaction protocols have been introduced as conversation contracts between participants in order to solve this. Interaction protocols are very difficult to develop and maintain. The present work provides a consolidating model that enables modularity for interaction protocols design and reuse of solutions by composing them to fit the different scenarios.

Key words: Interaction Protocols, Interoperability, Service Contracts, Business Process Modelling, Modularity

1.1 Introduction

Interoperability requires that organizations that desire to cooperate in a distributed process solve problems at different levels. Among concrete problems like data grounding is the issue of participants coordination in order to achieve a goal. This is the objective for interaction protocols. Experience has shown[17] that during the development process, these protocols can become very complex and difficult to manage. The present work is focused on this concrete problem: how to specify interaction protocols from a global perspective in a way that is easy and practical to recombine and reuse them.

In this document, a model for representing interaction protocols will be presented. This model is based on several proposals and contributions of the multi-agent community. One of the intentions is to consolidate these concepts in a single model and at the same time enable a mechanism for modularity

and composition of protocols. Interaction protocols are seen as a finite state machine (FSM), where states are the different situations a conversation can have and the transitions are the different actions that the participants can perform. For each possible state a set of possible actions will be specified, which are the possible actions in the specific situation that connect the state of the conversation to the resulting state if the action is performed.

These protocols are intended to be used in complex scenarios, like business processes, as contracts, stipulating how the participants are expected to behave, similar to rules in a game, in order to be interoperable. To make conversations more predictable and consequently easier to handle, interaction protocols are introduced. In principle, these narrow down the variability of a conversation to only those sequences of messages that work towards achieving the goal of the conversation. For that reason, conversation protocols are a very important component in interoperability, in which a simple way of coordinating participants to achieve a certain objective is desired. A conversation protocol is vital, in order to keep complexity demanded from the participants' behaviours low.

Reuse demands to have different levels of abstraction, in order to use solutions in different kinds of concrete cases. The present work will provide a model that can be used at all levels of abstraction, and treats protocols as first-class objects, which represent concepts from the global point of view and can be used by designers, but also by sophisticated software agents, in case they are being used, to reason about protocols.

First, in Section 1.2 a model of the state-action space will be defined, followed by Section 1.3 which will discuss some specific aspects of propositions in this model. Based on that, in Section 1.4 the model of conversation protocols is defined. Section 1.5 specifies how composition of protocols is to be done. Finally the present work is discussed in Section 1.6 and concludes in Section 1.7.

1.2 Definition of an state-action space model

1.2.1 Model of state space and state descriptions

A model of the state space of a conversation will be defined. Since these states tend to be rather large and repetitive, a practical mechanism to refer to a group of states that share properties in common will also be introduced.

Definition 1 *Propositions Set P: The finite set of all different atomic propositions used to describe states in a conversation:*

$$P = \{p_1, p_2, p_3, \ldots, p_n\}$$

with the property

$$thep_i \in P \wedge p_j \in P \wedge i \neq j \Rightarrow p_i \neq p_j \tag{1.1}$$

It is important to remark, that all propositions are completely distinct from each other. There are no two propositions in the set P about the same fact.

Definition 2 *State σ: a specific truth value assignation to each proposition that is part of a Conversation description (to all $p_i \in P$). The set of all States is Σ. A state σ is a set of elements of the relation T:*

$$T = P \rightarrow \{true, false\}$$

a state σ is defined as:

$$\sigma = \{\langle p_i, t\rangle | \langle p_i, t\rangle \in T, i = 1, 2, \ldots, n\} \tag{1.2}$$

where:
All propositions have a truth value assigned in each state

$$\forall p_i \in P, \exists \langle p_i, t\rangle \in \sigma,\ \forall \sigma \in \Sigma \tag{1.3}$$

A state cannot have a proposition associated with a truth value and at the same time the same proposition with the opposite truth value.

$$\langle p_i, t\rangle \in \sigma \Rightarrow \langle p_i, \neg t\rangle \notin \sigma,\ \forall \sigma \in \Sigma \tag{1.4}$$

Two states $\sigma \in \Sigma$ are equal if for each of propositions $p_i \in P$ they have the exact same truth value assigned:

$$\sigma_1 = \sigma_2 \Leftrightarrow \forall \langle p_i, t_i\rangle \in \sigma_1 \wedge \forall \langle p_i, t'_i\rangle \in \sigma_2 : t_i = t'_i \tag{1.5}$$

A state is for instance:

$$\sigma_1 = \{\langle p_1, true\rangle, \langle p_2, false\rangle, \ldots, \langle p_n, false\rangle\}$$

Since different states having differences irrelevant in the context may have the same meaning or properties from a certain perspective, a way for referring to such groups of states will be defined.

Definition 3 *A state description s is an association of a set of truth value assignations to some or all propositions of P, that serve as constraints, and a set of all states that fulfill these constraints. The set of all state descriptions is called S*[1]*.*

$$S : \mathcal{P}(\mathcal{T}) \rightarrow \mathcal{P}(\Sigma) \tag{1.6}$$

elements $s \in S$ are defined as:

$$s(\langle p_a, t_a\rangle, \ldots, \langle p_b, t_b\rangle) = \tag{1.7}$$

[1] $\mathcal{P}(X)$ is the power set of set X.

$$\{\sigma \in \Sigma | \langle p_a, t_a \rangle \in \sigma, \ldots, \langle p_b, t_b \rangle \in \sigma\}$$

for some arbitrary propositions p_x *in* $[p_a, \ldots, p_b]$ *where:*

$$a \geq 1 \wedge b \leq |P|$$

$$p_x \in P$$

$$t_x \in \{true, false\}$$

In this case it is important again to remark that state descriptions cannot have contradicting arguments, such state descriptions are empty:

Lemma 1 *All state descriptions with contradicting proposition truth value assignations are empty:*

$$s(\ldots, \langle p_p, t_p \rangle, \ldots, \langle p_p, \neg t_p \rangle, \ldots) = \emptyset \tag{1.8}$$

Proof 1 *From (1.4) we know, there are no states with two contradicting propositions.* □

From now on, for brevity, when there is no chance for confusion, mentioning a proposition will be synonym to assigning true to that proposition. At the same time, if the proposition is mentioned prefixed with the ¬ *negation* operator, the value assigned to the proposition is false:

$$\begin{aligned} p_i &= \langle p_i, true \rangle \\ \neg p_i &= \langle p_i, false \rangle \end{aligned} \tag{1.9}$$

For example, the state

$$\sigma_1 = \{\langle p_1, true \rangle, \langle p_2, false \rangle, \langle p_3, true \rangle, \ldots, \langle p_7, false \rangle, \ldots\}$$

can be written as:

$$\sigma_1 = \{p_1, \neg p_2, p_3, \ldots, \neg p_7, \ldots\}$$

Also, for instance, *state description* $s(p_1, \neg p_2, \neg p_7)$ can contain, among others, the following states:

- $\sigma_1 = \{p_1, \neg p_2, p_3, \ldots, \neg p_7, \ldots\}$
- $\sigma_2 = \{p_1, \neg p_2, \neg p_3, \ldots, \neg p_7, \ldots\}$
- $\sigma_3 = \{p_1, \neg p_2, \ldots, \neg p_7, \ldots, p_9, \ldots\}$
- $\sigma_4 = \{p_1, \neg p_2, \ldots, \neg p_7, \ldots, \neg p_9, \ldots\}$

But it cannot contain for instance the following states:

- $\sigma_5 = \{p_1, p_2, p_3, \ldots, \neg p_7, \ldots\}$
- $\sigma_6 = \{\neg p_1, \neg p_2, p_3, \ldots, \neg p_7, \ldots\}$
- $\sigma_7 = \{\neg p_1, p_2, \ldots, \neg p_7, \ldots, p_9, \ldots\}$
- $\sigma_8 = \{p_1, \neg p_2, \ldots, p_7, \ldots, \neg p_9, \ldots\}$

1.2.2 Model of actions in a state space

In a finite state machine (FSM) and similar transition systems, the evolution of a system during run-time is modelled as transition over different states of the state space. The change of one state to another is performed by actions. For each state in the state space there are only some possible actions. Based on the model of state space previously defined, a model of actions using the state descriptions will be defined:

Definition 4 *A micro-operation m on a state is an association of a state σ, an operator of the set $\{+,-\}$, a proposition p and a resulting state as follows:*

$$\Omega : \Sigma \times \{+,-\} \times P \times \Sigma$$

The set of all micro-operations M is:

$$\begin{aligned} M = \{\langle \sigma, \omega, p_o, \sigma' \rangle \in \Omega | \\ \forall \langle p_i, t_i \rangle \in (\sigma \setminus \langle p_o, t_o \rangle) : \exists \langle p_i, t_i' \rangle \in \sigma' : t_i = t_i' \wedge \\ \langle p_o, t_o \rangle \in \sigma' \} \end{aligned} \tag{1.10}$$

where

$$t_o = \begin{cases} true & if\ \omega = + \\ false & if\ \omega = - \end{cases}$$

For a micro-operation $m = \langle \sigma, \omega, p, \sigma' \rangle$ the first operand σ is referred to as the starting state and σ' the target state. The target state is identical to the starting state with the exception of the element with the proposition p_o, which has a value dictated by the operand ω.

A micro-operation is the representation of the concept of bringing about a fact, making a proposition true, or removing the proposition, making it false. It can be possible that σ has already the proposition p in true, in such a case applying the operation will not produce any change and σ' would be exactly like σ.

Using the definition of a state description, an operation is defined as a set of micro-operations over the set of states defined by the state description:

Definition 5 *An operation o is an association of a state description s, a member of $\{+,-\}$, a proposition p and a set of micro-operations M' that have initial states belonging to the state description s:*

The set of all operations is called O.

$$O : S \times \{+,-\} \times P \rightarrow M' \subseteq M$$

$$o_{s,\pm p} = \{\langle \sigma, \pm, p, \sigma' \rangle \in M' | \sigma \in s\} \tag{1.11}$$

where

$\pm$: is a place holder for a member of the set $\{+,-\}$.

Moreover, the specific state σ' targeted by the operation o, if a specific initial state σ is provided, is obtained by getting the micro-operation m member of o that has as starting state σ:

$$o_{s,\pm p}(\sigma) = \sigma' \tag{1.12}$$

where

$$\langle \sigma, \pm, p, \sigma' \rangle \in o_{s,\pm p}$$

This allows to specify a single operation that applies to several states and with the operation $o_{s,\pm p}(\sigma)$ it is possible to know specific cases.

Operations applying to different propositions can be composed in a single one.

Definition 6 *A composed operation c_s is a set of different operations which all refer to the same state description s and to different propositions:*

$$c_s \subseteq O$$

where

$$o_{s\pm p} \in c_s \Rightarrow \nexists o_{s\pm p'} \in c_s : p = p' \tag{1.13}$$
$$i = 1, \ldots, |c_s|$$

Similarly to simple operations, the specific state targeted by the composed operation can be found using following definition:

$$\begin{aligned} c_s(\sigma) = \{ \langle p_i, t_i' \rangle | \ & (o_{s\omega_i p_i} \in c_s \Rightarrow \langle p_i, t_i' \rangle = \omega_i p_i) \quad \vee \\ & (o_{s\omega_i p_i} \notin c_s \Rightarrow \langle p_i, t_i' \rangle = \langle p_i, t_i \rangle) \quad \} \end{aligned} \tag{1.14}$$

where

$$\langle p_i, t_i \rangle \in \sigma$$
$$\omega_i \in \{+, -\}$$
$$i = 1, \ldots, |c_s|$$
$$\omega_i p_i = \begin{cases} \langle p_i, true \rangle \ if \ \omega_i = + \\ \langle p_i, false \rangle \ if \ \omega_i = - \end{cases}$$

The resulting state of applying the composed operation is a state which has all its corresponding propositions unchanged, except those for which an operation could be found in the composed operation, in which case, the value specified by the operation will be used.

Since all operations in a composed operation refer to the same starting state description, an abbreviated form of writing an action is:

$$\begin{aligned} c_s = \{ & o_{1\ s,\pm p_1}, o_{2\ s,\pm p_2}, o_{3\ s,\pm p_3}, \ldots, o_{m\ s,\pm p_m} \} = \\ & \{ \pm p_1, \pm p_2, \pm p_3, \ldots, \pm p_m \} \end{aligned} \tag{1.15}$$

For instance, the composed operation $c_{s_1} = \{+p_1, -p_3\}$ for any state in $s_1 = s(\neg p_2, p_4)$: performing it will change the current state of the system to another identical one with exception of p_1 which will be definitely true and p_3 which will be definitely false after the applying the composed operation.

$$\sigma_1 = \{p_1, \neg p_2, p_3\, p_4\} \in s(\neg p_2, p_4)$$
$$c_{s_1}(\sigma_1) = \{p_1, \neg p_2, \neg p_3, p_4\}$$

also:

$$\sigma_2 = \{\neg p_1, \neg p_2, p_3\, p_4\} \in s(\neg p_2, p_4)$$
$$c_{s1}(\sigma_2) = \{p_1, \neg p_2, \neg p_3, p_4\}$$

In the specific state σ_1 applying the operation c_{s1} will produce the state $\{p_1,\ \neg p_2,\ \neg p_3\ \ p_4\}$ since all propositions not mentioned in c_{s1} (p_2 and p_4) remain the same, p_1 also remains as previously, since it was already true an the operation specifies $+p_1$, the opposite case happens with σ_2, where p_1 was negative and turns positive. p_3 is the only one changing in both, since in σ_1 and σ_2 it was true and the operation says $-p_3$, therefore it will be set to false. In other words, in σ_1 the composed operation c_{s1} brings about only p_3, since p_1 is already valid and in σ_2 the composed operation c_{s1} brings about both, p_1 and p_3.

Definition 7 *The state description that describes all states targeted by the composed operation c_s can be found using the function $S(c_s)$:*

$$S(c_s) = s(\langle p, t\rangle |\ (+p \in c_s \wedge t = true) \vee (-p \in c_s \wedge t = false) \vee (\pm p \notin c_s \wedge \langle p, t\rangle \in s)) \tag{1.16}$$

The targeted state description s' can be calculated by setting all the propositions as specified in c_s and keeping all other propositions not mentioned in c_s as they were in s. Note that in this situation being $\pm p \notin c_s$ part of a conjunction, it is also interpreted as a conjunctional abbreviation meaning that neither of both cases of $\pm p$ are elements of c_s.

Lemma 2 *All calculated states from the state s using the composed operation c_s are in the state description calculated by the function $S(c_s)$:*

$$c_s(\sigma) \in S(c_s)\ \forall \sigma \in s \tag{1.17}$$

Proof 2 *That all states obtained by applying c_s on a state in s are part of the state description $S(c_s)$ can be proved by contradiction:*

$$\begin{aligned}
&\exists \sigma \in s: && c_s(\sigma) \notin S(c_s)\\
\Leftrightarrow\ &\exists \langle p, t\rangle \in c_s(\sigma): && \neg(\ (+p \in c_s \wedge t = true) \ \vee\\
& && \quad (-p \in c_s \wedge t = false) \vee\\
& && \quad (\pm p \notin c_s \wedge \langle p, t\rangle \in s)\ )\\
\Leftrightarrow\ &\exists \langle p, t\rangle \in c_s(\sigma): && \neg\ (+p \in c_s \wedge t = true) \ \wedge\\
& && \neg\ (-p \in c_s \wedge t = false) \wedge\\
& && \neg\ (\pm p \notin c_s \wedge \langle p, t\rangle \in s)
\end{aligned}$$

$+p$ and $-p$ can be rewritten as $o_{s\omega p}$ using (1.15) summarizing the first two operands of the conjunction to one that expresses both cases at the same time with the same format as done in (1.14):

$$\begin{aligned} \Leftrightarrow \exists \langle p,t\rangle \in c_s(\sigma) : \neg\, (o_{s,\omega,p} \in c_s \wedge \langle p,t\rangle = \omega p) \wedge \\ \neg\, (\pm p \notin c_s \wedge \langle p,t\rangle \in s) \end{aligned}$$

which is clearly the contradiction of Definition 6 for targeted states $c_s(\sigma)$ of a composed operation(1.14). □

In order to model the application of a sequence of composed operations, the following operator over composed operations will be defined:

Definition 8 *The binary operator "chain" represented by "$\rightarrow$" is defined as the association of 2 composed operations ($c1_s$, $c2_{s'}$) and a third resulting one ($c3_s$) such that:*

- *All states resulting of the application of the first composed operation are part of the state description s' of the second composed operation*
- *The third (resulting) composed operation $c3_s$ is the set of operations of the first composed operation* overridden *by the operations of the second composed operation: all operations of the second set of operations $c2_{s'}$ are part of the result together with all those of the first set $c1_s$ that refer to propositions not mentioned in $c2_{s'}$*

$$\begin{aligned} c1_s \rightarrow c2_{s'} = c3_s \qquad & s.t: \\ & S(c1_s) = s' \\ c3_s = \{o_{s\omega_i p_i} \in c1_s | o_{s'\omega_j p_i} \notin c2_{s'}\} \bigcup & c2_{s'} \\ where & \\ & \omega_x \in \{+,-\},\ x = 1,\ldots \end{aligned} \tag{1.18}$$

Lemma 3 *The specific state targeted by a chain operation is the same as the state targeted by the second operand of the state targeted by the first operand of the chain operation:*

$$(c1_s \rightarrow c2_{s'})(\sigma) = c2_{s'}(c1_s(\sigma)) \tag{1.19}$$

Proof 3 *By applying recursively (1.14) we obtain:*

$$c2_{s'}(c1_s(\sigma)) = \sigma'' :$$

$$\forall \langle p_i, t_i''\rangle \in \sigma'' : \begin{cases} o_{s'\omega_i' p_i} \in c2_{s'} \Rightarrow \langle p_i, t_i''\rangle = \omega_i' p_i \\ o_{s'\omega_i' p_i} \notin c2_{s'} \Rightarrow \begin{cases} o_{s\omega_i p_i} \in c1_s \Rightarrow \langle p_i, t_i''\rangle = \omega_i p_i \\ o_{s\omega_i p_i} \notin c1_s \Rightarrow \langle p_i, t_i''\rangle = \langle p_i, t_i\rangle \end{cases} \end{cases}$$

which can be rewritten as:

$$\forall\langle p_i, t_i''\rangle \in \sigma'' : \begin{cases} o_{s'\omega_i' p_i} \in c2_{s'} \Rightarrow \langle p_i, t_i''\rangle = \omega_i' p_i \\ o_{s'\omega_i' p_i} \notin c2_{s'} \wedge \\ \quad o_{s\omega_i p_i} \in c1_s \Rightarrow \langle p_i, t_i''\rangle = \omega_i p_i \\ o_{s'\omega_i' p_i} \notin c2_{s'} \wedge \\ \quad o_{s\omega_i p_i} \notin c1_s \Rightarrow \langle p_i, t_i''\rangle = \langle p_i, t_i\rangle \end{cases}$$

which can be rewritten as:

$$\forall\langle p_i, t_i''\rangle \in \sigma'' : \begin{cases} o_{x\omega_i'' p_i} \in Q \Rightarrow \ \langle p_i, t_i''\rangle = \omega_i'' p_i \\ o_{x\omega_i'' p_i} \notin Q \Rightarrow \langle p_i, t_i''\rangle = \langle p_i, t_i\rangle \end{cases}$$

where

$$x \in \{s, s'\}$$

$$Q = c2_{s'} \bigcup \{o_{s\omega_i p_i} | o_{s'\omega_i' p_i} \notin c2_{s'} \wedge o_{s\omega p_i} \in c1_s\}$$

$$\omega_i'' = \begin{cases} \omega_i' \ if \ o_{s\omega_i' p_i} \in Q \\ \omega_i \ if \ o_{s\omega_i p_i} \in Q \end{cases}$$

which can be rewritten as:

$$\forall\langle p_i, t_i''\rangle \in \sigma'' : \begin{cases} o_{x\omega_i'' p_i} \in Q \ \Rightarrow \ \langle p_i, t_i''\rangle = \omega_i'' p_i \\ o_{x,\omega_i'' p_i} \notin Q \Rightarrow \langle p_i, t_i''\rangle = \langle p_i, t_i\rangle \end{cases}$$

where

$$Q = c2_{s'} \bigcup \{o_{s\omega_i p_i} \in c1_s | o_{s'\omega_i' p_i} \notin c2_{s'}\}$$

using (1.18) it is clear that $Q = c1_s \rightarrow c2_{s'}$*:*

$$\forall\langle p_i, t_i''\rangle \in \sigma'' : \begin{cases} o_{x\omega_i'' p_i} \in c1_s \rightarrow c2_{s'} \Rightarrow \langle p_i, t_i''\rangle = \omega_i'' p_i \vee \\ o_{x\omega_i'' p_i} \notin c1_s \rightarrow c2_{s'} \Rightarrow \langle p_i, t_i''\rangle = \langle p_i, t_i\rangle \end{cases}$$

which is the definition for states that are target of $(c_{1s} \rightarrow c_{2s'})(\sigma)$ *by using (1.14).* □

Using operations a definition of an *action* is proposed based on three important aspects of an action:

- In which states it applies
- Its label
- The operations (+ or -) it applies on different propositions

Definition 9 *Action Descriptions are a mapping of a state description, a label and a set of operations over some propositions to a set of possible states. Action descriptions must follow a* principle of effectiveness*, therefore the set of targeted states cannot be the same as the starting state description. The set of all actions is called A:*

$$\Delta : S \times V \times C \times S$$

$$A = \{\langle s, v, c, s' \rangle \in \Delta | s \neq s'\} \tag{1.20}$$

Moreover, the specific state targeted by an action starting from a specific state σ *is obtained by using the corresponding* $c(\sigma)$ *operator :*

$$a_{s,c}(\sigma) = c_s(\sigma) \tag{1.21}$$

1.2.3 State-Action model extension to include Roles

A conversation involves always at least two participants. In an abstract description of a conversation like the one proposed here, these participants are represented as *roles* (r) members of a set called $r \in R$. With the intention of modelling conversations using the state-action model, the concept of roles is integrated in this section.

Every action in our model is always performed by a role and is targeted at another role. In the present model, actions represented as operations over propositions mean that an agent is performing an action of communicating information to another agent, in other words, sending a message. Therefore the definition of action will be enhanced with the sender and receiver roles of the actions:

Definition 10 *A* speech act a_r *is an association of a role, an action as defined in Definition 9 and a different second role, they are all members of the set* A_R*:*

$$A_R : R \times A \times R \tag{1.22}$$

where

$$a_i = \langle r_{xi}, \langle s_i, l_i, c_i \rangle, r_{yi} \rangle$$
$$\forall a_i \in A_R : r_{xi} \neq r_{yi}$$

For instance, the speech act

$$a_r = \langle r_1, \langle s(p_1), "respond", \{-p_1\} \rangle, r_2 \rangle$$

represents the action labeled as "respond" that can be sent by role r_1 when p_1 is true to the role r_2 and defined as removing the fact p_1.

For simplicity, the term action will be used from here on for both concepts, action description as in Definition 9 and speech act as in Definition 10, whenever there is no chance for confusion.

1.3 Special kinds of propositions

Even though the present model is intended to be of general domain, some particular kinds of propositions are frequently needed when modeling interaction protocols. In the present model three of them will be defined, timeouts, commitments and operational propositions.

1.3.1 Timeouts

Protocols are mechanisms to rule actions over time. Sequencing and turn taking are problems that are solved with the present model, but there are certain cases where concrete time-windows are to be specified. For this purpose timeouts will be defined.

Definition 11 *A Timeout $T(t_p, a) \in \mathcal{T}$, where $a \in A_R$, $\mathcal{T} \subseteq P$, is a proposition member of the set of timeout propositions $\mathcal{T}$ that states that the action a will be performed after a certain period of time t_p that starts to count after the action that brings about the timeout is performed. Action a in a timeout is not necessarily performed by the sending role mentioned in a, but instead it can be an assumption the receiver of a can make. Also, this action will always have implicitly the operation $-T(t_p, a)$ declared, hence, timeouts are removed automatically after a is performed.*

For instance, the action a says, that after performing call for proposals, M will send the message "*done*" after a period of size t_d, after which *proposition requested (p_r)* will not longer hold:

$$a = \langle M, \langle s, cfp, \{+p\_r, +T(t_d, \langle M, \langle \{p\_r\}, done, \{-p\_r\}\rangle, B\rangle)\rangle, B\rangle\},$$

1.3.2 Commitments

Singh [16] and his group have proposed an algebra for commitments [18] which provides the advantage of allowing better modularity in the design of processes [8]. Taking advantage of the similarities of this algebra and the proposed operations in the previous section, the concept of commitment will be integrated to the state-action model.

Definition 12 *Commitment $C(a_d, a_c, p, c, t) \in P$ is defined as the commitment of the* debtor *agent a_d to the* creditor *agent a_c to bring about the proposition $p \in P$ under the condition that the proposition $c \in P$ becomes true. After the condition c becomes true, agent a_d is expected by agent a_c to perform some action that produces p to be true. This action is to be performed before timeout t that represents the time limit is enabled. This timeout starts to count as soon as the condition is brought about.*

Definition 13 *Unconditional Commitment $C(a_c, a_d, p, t)$: As an abbreviation the following notation will be taken:*

$$C(a_d, a_d, p, true, t) \;=\; C(a_c, a_d, p, t) \tag{1.23}$$

Meaning simply that agent a_c expects a_d to bring about the proposition p within the time period specified in t.

Timeouts t in commitments are not restricted to any specific purpose, but the main intention to include them in the definition of commitments is to provide the semantics of what will happen if the commitment is not satisfied.

It is important to note that the commitments are part of the set of propositions P, they are part of the propositions that can also be used for specifying state descriptions and actions. They are also operands for the defined operators + and -. The detailed semantics of these two operators specifically on commitments will be defined next:

Definition 14 *Commitment creation:* $+C(a_c, a_d, p, c, t)$. *Creating a commitment means that after the creation of it, agent* a_d *is committed to agent* a_c *to bring about* p *under the condition* c. *An action specifying this operation states that after the action is performed, the specified commitment starts to exist.*

Definition 15 *Commitment canceling:* $-C(a_c, a_d, p, c, t)$. *If the commitment exists, performing the operation - on it cancels it, makes it a false proposition, meaning that it does not longer exist,* a_d *is no longer expected by* a_c *to bring about* p. *An action specifying this operation states that after the action, the specified commitment, including its timeout, does not exist anymore.*

Definition 16 *Bringing about the condition* c enables *the commitment. If the condition is true, the commitment is transformed to an unconditional commitment: the conditional commitment is canceled and the unconditional commitment is created enabling the timeout countdown:*

$$\frac{c \wedge C(a_d, a_d, p, c, t)}{c \wedge C(a_c, a_d, p, t) \wedge \neg C(a_d, a_d, p, c, t)} \tag{1.24}$$

Any state where a conditioned commitment and its condition are at the same time true are automatically transformed to a state where the condition still exists, but the commitment has been replaced by an unconditional commitment.

Definition 17 *Commitment discharge: Bringing about the commitment objective* p *before the timeout* t *has been enabled cancels created commitments that have* p *as objective automatically, including their timeouts.*

$$\frac{p \wedge C(a_d, a_d, p, c, t)}{p \wedge \neg C(a_d, a_d, p, c, t)} \tag{1.25}$$

Any state where a commitment to bring about a proposition p *and at the same time the condition* p *are true are automatically transformed to a state where the condition* p *still exists, but the commitment has been canceled and does not exist any more.*

In Singh's proposal [16], there are two more operations on commitments that will be included, but integrated as actions part of our model. These are namely, *delegation* and *assignation* of commitments. These will not be extra defined, but instead two examples of how these operations are present in the model are presented:

- Delegation: The action d, labeled *delegates*, changes the debtor of a commitment C from agent a_{d1} to a_{d2}:
  $d = \langle s(C(a_c, a_{d1}, p, c, t)), \text{“}delegates\text{”}, \{-C(a_c, a_{d1}, p, c, t), +C(a_c, a_{d2}, p, c, t)\}\rangle$
- Assignation: The action a, labeled *assigns*, changes the creditor of a commitment C from agent a_{c1} to a_{c2}:
  $a = \langle s(C(a_{c1}, a_d, p, c)), \text{“}assigns\text{”}, \{-C(a_{c1}, a_d, p, c, t), +C(a_{c2}, a_d, p, c, t)\}\rangle$

Speech acts can also be used with commitments, an example of such an action can be sending the acceptance of a role r_1 to r_2 whenever p_1 is not true (for instance p_1:box 1 is in slot A) to commit doing something to make p_1 to be true if the fact c_1 (for instance c_1: there is nothing on top of box 1) is true:
$a = \langle r_1, \langle s(\neg p_1), "accept'', \{+C(r_2, r_1, p_1, c_1, t_1)\}, r_2\rangle$

1.3.3 Operational propositions

In order to have better control over conversations and to specify specific ways a role is allowed to decide or react, some propositions will have to be more complex than simple statements about the environment. Some of these propositions are arithmetic propositions, like counters or variables holding values, others are conditional logic statements that can produce other propositions given some specific condition. Another kind of operational proposition that will be mentioned specifically is the binding of specific concepts used in the action model to variables.

In the case of variables, the most crucial aspect is to manage their scope. Variables can exist one for each conversation, or one for each instance of a role or even a single global variable that is the same instance in all conversations in the protocol.

The aspect of how to use and define propositions will be discussed in further depth later in this paper, at this point, only these aspects of representing propositions or connecting them to more concrete concepts will be presented.

An example of an action with the three different kinds of propositions is the following, where a bidder B bids to an auction manager M the value *bid*:

$$\langle B, \langle \{a\_a, v = x, \}, bid, \{bid, bid > v \Rightarrow v = bid, -C(M.W, M, pay, win, t_w), +C(B, M, pay, win, t_w), M.W = B\}\rangle, M\rangle$$

where:

a_a: auction active
v: current winning value in the auction
$M.W$: Winner role holding variable in M
bid: the bid given by B

A conditional proposition will tell how the manager is expected to react: if the bid is bigger than the current value v, v will be assigned the new value of *bid*, any previous commitment to a winner is removed and a new commitment

with the sender of the bid B is created, finally this bidder is assigned the role of the winner $M.W$.

1.4 Definition of a conversation protocol

Using the proposed model of a conversation based on actions sent between roles participating in a conversation, the following section will provide the mechanisms to compose these actions in such a way that they describe how complex conversations are to be performed.

Protocols represent specific ways that it allows over the whole state-action space. The different ways a protocols can take are known as a *runs*. In the concrete case of an agent performing a protocol with more than one participants, there will be several instances of the protocol, called *conversations*, each of which will take its own run, some of them will have the same run. Figure 1.1 illustrates the relation between these 3 concepts. In synthesis a protocol is a specification of a set of runs, each of which represent at the same time a set of conversations. The cardinality of runs is limited by the amount of actions that are enabled for the same state description. The cardinality of conversations, on the other hand is to be ruled by an amount that is associated with each action:

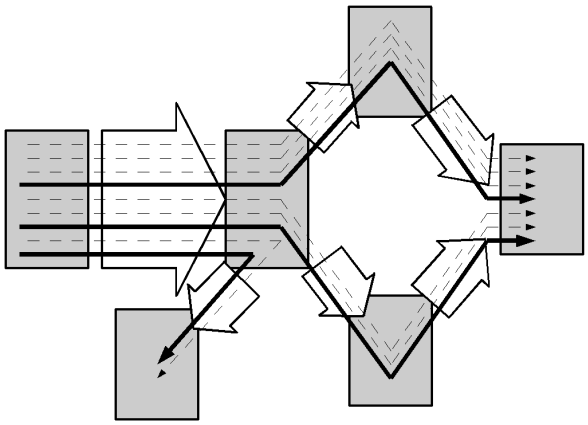

Fig. 1.1. Protocol (boxes and white arrows) composed of 3 runs (thick lines) and some conversations as instances of runs (dashed thin lines)

Definition 18 *The set C_A, the set of cardinality constraints for actions and the set C_S, the set of cardinality constraints for states, will represent the*

minimal and maximal cardinality of conversations associated to each action or state respectively:

$$C_A : \mathbb{N} \times \mathbb{N} \times A_r \tag{1.26}$$

where

$$\forall \langle p, q, a \rangle \in C_A : p < q$$
$$\forall a \in A_r : \langle p, q, a \rangle \in C_A$$

$$C_S : \mathbb{N} \times \mathbb{N} \times S \tag{1.27}$$

where

$$\forall \langle p, q, s \rangle \in C_S : p < q$$
$$\forall s \in S : \langle p, q, a \rangle \in C_S$$

For brevity and agility, the following abbreviated version of a cardinality constraint will be used:

$$\langle p, q, a \rangle =_p^q a \tag{1.28}$$

Also in cases where the cardinality of actions is free: minimum is 0 and maximum not bound, represented with N, the cardinality constraints can be omitted:

$$\langle 0, N, a \rangle = a \qquad \langle p, N, a \rangle =_p a \qquad \langle 0, q, a \rangle =^q a \tag{1.29}$$

For instance, the following cardinality association means that the action called "inform" can be sent from the role r_1 to the role r_2 a maximal of 7 times and a minimal of 1 times from states in s.

$$_1^7\langle r_1, \langle s, \text{“}inform'', o \rangle, r_2 \rangle$$

Protocols will be specified similarly as in [6], in terms of the propositions required to start it, called preconditions and propositions describing the effects it has in the context of the conversation, called post-conditions. Protocols, like actions, will be also labeled.

Definition 19 *A protocol π is an association of preconditions in the form of one or more state descriptions, a label, post-conditions in the form of one or more state descriptions and a set of speech acts. Pre-, post-conditions and speech acts are associated to cardinality constraints, the set of all protocols is called Π:*

$$\Pi : \mathcal{P}(C_S) \times V_\pi \times \mathcal{P}(C_S) \times \mathcal{P}(C_A) \tag{1.30}$$

where V_π is the set of labels for protocols.

Protocols have cardinality constraints associated to their starting and ending state descriptions as follows:

$$\begin{array}{l}\forall \pi \in \Pi : \\ where \\ \pi = \langle \mathcal{S}, l, \mathcal{E}, \mathcal{A} \rangle \\ \mathcal{A} = \{ {}^{q_1}_{p_1}\langle r_x, \langle s_1, v_1, c_1 \rangle, r_y \rangle, \ldots, {}^{q_h}_{p_h}\langle r_x, \langle s_h, v_h, c_h \rangle, r_y \rangle \}\end{array}$$

all actions sharing a starting state of the protocol ${}^{q_a}_{p_a} s_a \in \mathcal{S}$ *as precondition must satisfy the cardinality constraints, which are the cardinality constraints associated to the state description* s_a*:* p_a *and* q_a *:*

$$\begin{array}{l}\forall ({}^{q_i}_{p_i}\langle r_x, \langle s_i, v_i, c_i \rangle, r_y \rangle \in \mathcal{A} | s_a = s_i) : p_i \leq p_a \wedge q_i \geq q_a \\ where \\ i = 1, \ldots, |\mathcal{S}|\end{array} \tag{1.31}$$

all actions that result in a state description that matches an end state description of the protocol ${}^{q_e}_{p_e} s_e \in \mathcal{E}$ *must have the same cardinality constraints, which are the cardinality constraints associated to the end state description* p_e *and* q_e*:*

$$\begin{array}{l}\forall ({}^{q_i}_{p_i}\langle r_x, \langle s, v, c \rangle, r_y \rangle \in \mathcal{A} | S(c_s) \in \mathcal{E}) : p_i \geq p_e \wedge q_i \leq q_e \\ where \\ j = 1, \ldots, |\mathcal{E}|\end{array} \tag{1.32}$$

A protocol cannot have disconnected actions in its definition:

$$\forall r \in R \; \exists s_s \in \mathcal{S} \wedge R' \subset R : s_i = s_{i-1}(c_i) \; for \; 0 < i \leq k \tag{1.33}$$

where:

$$\begin{array}{l} r = \langle r_x, \langle s, v, c \rangle, r_y \rangle \\ R' = \{r_0, r_1, \ldots, r_k\} \\ r \notin R' \\ r_i = \langle r_{xi}, \langle s_i, v_i, c_i \rangle, r_{yi} \rangle \; for \; 1 \leq i \leq k \\ r_0 = \langle r_{x0}, \langle s_s, v_0, c_0 \rangle, r_{y0} \rangle \end{array}$$

This more relaxed approach facilitates different composition mechanisms for protocols and reflects the multi-directional nature of conversations. The set of actions of the protocol are also called *rules* as used in dialogue games [10].

Protocols that have the same pre- and post- conditions are not necessarily the same, but are expected to fit in a protocol composition well according to the semantics treated in the present work.

The nature of conversations makes the task of modeling and structuring them very hard. It is the intention of this work to provide mechanisms of organization and modularization of complex conversations without restricting them unnecessarily or in such a manner that ends up being unnatural for practical purposes. Therefore a technique for composing protocols using rigid structures that not always fit the nature of conversations will not be pursued here. Such structures found commonly in similar approaches are probably inherited from other programming structures, like *if... then... else...*, *while*

loops and specially strict *joining* associated to a previous *split* in the transition system. Even though this approach allows such structures, it is by far not restricted to them. Even so, some basic structures that appear in conversation models will be described next using the model for protocols:

1.4.1 Atomic protocol

An atomic protocol is the most basic protocol possible, in essence a speech act. Its preconditions is the state description and its post-conditions is the composed operation of the action. The cardinality constraints associated to the starting and ending state description is the same as the ones for the action.

Definition 20 *The atomic protocol for the* roled action ${}^{q}_{p}a = {}^{q}_{p}\langle r_s, \langle s, v, c\rangle, r_r\rangle$ *is:*

$$\pi = \langle \{{}^{q}_{p}s\}, l, \{{}^{q}_{p}s(c)\}, \{{}^{q}_{p}\langle r_s, \langle s, v, c\rangle, r_r\rangle\}\rangle \tag{1.34}$$

1.4.2 Protocol sequence

A protocol sequence is a protocol which seen from the outside will have only one possible run, which means it has only one starting and one ending state description with identical cardinality constraints.

Definition 21 *Protocol sequence is a protocol* π *such that:*

$$\pi = \langle \{{}^{q}_{p}s\}, l, \{{}^{q}_{p}e\}, R\rangle \tag{1.35}$$

All atomic protocols are hence protocol sequences.

1.4.3 Protocol splits

Protocols splits represent the most common situations found when modelling conversations. These are the situations in which a protocol has a set of rules, all of them sharing the same starting state description.

Splits are the situations in which different *runs* are created, therefore a very relevant aspect of splits in a protocol is the definition of cardinality constraints for each run. This same procedure is also used in several previous approaches, for instance in Agent UML [3].

Definition 22 *A protocol split* π *is composed of more than one rule, all of which share the same starting state description.*

$$\pi = \langle \{{}^{q_s}_{p_s}s\}, l, \{{}^{q_1}_{p_1}s(c_{s1}), {}^{q_2}_{p_2}s(c_{s2}), \ldots, {}^{q_n}_{p_n}s(c_{sn})\}, \{a_1, a_2, \ldots, a_n\}\rangle \tag{1.36}$$

where

$$n > 1$$

$$a_i = {}^{q_i}_{p_i}\langle r_{xi}, \langle s, l_i, c_{s_i}\rangle, r_{yi}\rangle$$

Protocol splits can be, in principle, of two kinds: a choice or a parallel split.

Choice:

These are situations in which choosing one action, where all the actions have the same sending role, will disable the other ones, making it impossible to perform other actions that start at the splitting state description. This is a situation where the role in turn is expected to decide which of the options it will take, choosing this way the path in the protocols to be taken. π is a choice split if

$$\forall a_i : s(c_{si}) \bigcap s = \emptyset \wedge r_{xi} = r_x \tag{1.37}$$

In a choice, different runs are created, but each conversation can follow only one of them.

Parallel:

These are situations in which actions belonging to the split do not disable themselves reciprocally or have different senders, making it possible to perform one or more of these actions. In the present model only one action is possible to be performed at the same time, but the enabling conditions for other actions in a parallel split are still valid after an actions is performed, making it still possible to perform the other ones. π is a parallel split if

$$\forall a_i, j \leq n \wedge j \neq i : s(c_{si}) \subset s \vee r_{xi} \neq r_{xj} \tag{1.38}$$

In a parallel, different runs are created as well, but a conversation can follow one or more paths.

Protocol Merges

In opposite to protocol splits, where new runs are created, there are also situations where different runs merge into the same path.

Definition 23 *A protocol merge π is composed of more than one rule all of which share the same ending state description.*

$$\pi = \langle \{ {}^{q_{s1}}_{p_{s1}} s_1, {}^{q_{s2}}_{p_{s2}} s_2, \ldots, {}^{q_{sn}}_{p_{sn}} s_n \}, l, \{s_f\}, \{a_1, a_2, \ldots, a_n\} \rangle \tag{1.39}$$

where

$$n > 1$$

$$a_i = {}^{q_{si}}_{p_{si}} \langle r_x, \langle s_i, l_i, c_{si} \rangle, r_y \rangle$$

$$s(c_{si}) = s_f$$

1.5 Protocol composition

Protocol composition is the creation of new conversation protocols by connecting other protocols together.

Definition 24 *Two protocols π_1 and π_2 can be composed to a new protocol π_3, if there is at least one ending state description s_1 in π_1 that is subset of a starting state s_2 in π_2 and at the same time, cardinality constraints in s_1 are equal or more restrictive than in s_2. Propositions and roles have to be* bound *together to establish the semantic connection between the two protocols π_1 and π_2, by specifying which roles and propositions in the first protocol will take the roles and replace the propositions in the second protocols respectively:*

$$\begin{array}{l}\pi_1 = \langle S_1,'' \pi_1'', E_1, A_1\rangle \\ A_1 = \{\langle x1_1, a1_1, y1_1\rangle, \langle x1_2, a1_2, y1_2\rangle, \ldots, \langle x1_n, a1_n, y1_n\rangle\} \\ {}^{cq_1}_{cp_1} s_1 \in E_1; s_1 = \{p1_1, p1_2, \ldots, p1_f\} \\ \pi_2 = \langle S_2,'' \pi_2'', E_2, A_2\rangle \\ A_2 = \{\langle x2_1, a2_1, y2_1\rangle, \langle x2_2, a2_2, y2_2\rangle, \ldots, \langle x2_m, a2_m, y2_m\rangle\} \\ {}^{cq_2}_{cp_2} s_2 \in S_1; s_2 = \{p2_1, p2_2, \ldots, p2_g\}\end{array}$$

where

$$\begin{array}{l} g < f \\ cp_1 \geq cp_2 \\ cq_1 \leq cq_2 \end{array}$$

a specific binding of roles is specified:

$$\begin{array}{l} x1_i = x2_j \\ y1_i = y2_j \\ 1 \leq i \leq n; 1 \leq j \leq m \end{array}$$

and a specific binding of propositions:

$$\left.\begin{array}{l} p1_k = p2_l \\ 1 \leq k \leq g \\ 1 \leq l \leq g \end{array}\right\} \Rightarrow s_1 \subseteq s_2$$

$$\pi_3 = \langle S_1 \bigcup S_2,'' \pi_3'', E_1 \bigcup E_2, A_1 \bigcup A_2\rangle \tag{1.40}$$

1.5.1 Protocol example

As an example, the case of a specific kind of auction will be observed. Figure 1.2 shows a graph representing the protocol. A manager M can send any time a message called *cfp1* to a set of bidders represented by the role B, which can be 0 to the maximum possible (N). *cfp1* creates the conversations making some propositions valid: that the auction is active (*auc_act*), a timer is created that after a period *ta* will enable the action *done*. The value v is created with

the starting value of the auction, say 0 and finally a commitment is created *C(B,M,bid,T(tb, reject))* that states that the bidder B will be committed to the manager M to bid within a period *tb*, after which, if no replies are received, it will be assumed that the Bidder has performed the action *reject*. B has to decide to send either a reject message, which practically takes him out of the auction, removing all propositions including *auc_act* or to bid, providing a bid value called *bid* which in case it is greater than *v* will have further effects: *v* will have the value of *bid*, a commitment to a previous winner of a auction round M.W will be removed (has no effect in the first iteration) and a commitment with the same terms is created for the bidder of *bid* in which it is committed to pay an amount *pay* in case it wins the auction: *win*. This leading bidder is then bound to the winning role. Notice that after each iteration, M *informs* M.W that it won the round. Finally the timeout for *done* is enabled finishing the auction: M sends *done* to all participants which removes the fact *auc_act* finishing the auction and to the winner M.W it sends a different kind of done which also brings about the final value of the auction *v*, the fact that the receiver won: *win* which automatically transforms the the conditional commitment created with the winner *bid* to an unconditional commitment of the bidder to the manager to pay within the timeout *tp*. Also the commitment of M to B to make *delivered* true within the timeout *td* in case *pay* is brought about is created. This would be the ending state of the auction, with another ending state for the rejected bidders and non winner bidders.

As an example of protocol composition, a very simple protocol π_2 is defined which has a starting state matching the end state of the auction. The first an only possible action is *pay* which automatically discharges the commitment from B to M and frees the commitment of M to B of its condition. After that, the only possible action is *deliver* which discharges the last commitment ending the protocol in a state where *delivered* is true. This example has the same propositions and roles which makes the binding explicit. The states are connected and a new protocol π_3 is created that resolves the commitments.

The protocols π_1 and π_2 look like this:

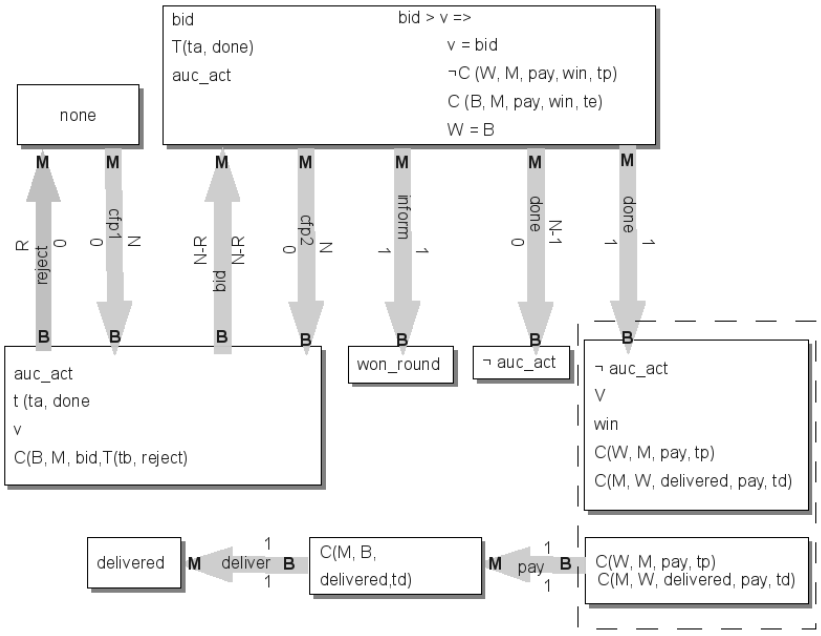

Fig. 1.2. Example of a protocol modelling a composition of an auction and a simple commitment resolution

$$
\begin{aligned}
\pi_1 = \langle\{\}, \pi_1, \{ \quad & s(\neg auc\_act), s(\neg auc\_act, v, win, \\
& C(M.B, pay, tp), C(M, B, delivered, pay, td))\}, \\
\{\langle M, \langle \quad & \{s(tb < ta)\}, cfp1, \{+auc\_act, +T(ta, done), +v, \\
& +C(B, M, bid, T(tb, reject))\}\rangle, B\rangle, \\
{}^{R}_{0}\langle B, \langle \quad & \{s(auc\_act, C(B, M, bid, T(tb, reject)))\}, reject, \\
& \{-C(B, M, bid, T(tb, reject), -auc\_act, \\
& -T(ta, done))\}\rangle, M\rangle, \\
{}^{N-R}_{N-R}\langle B, \langle \ & \{s(auc\_act, C(B, M, bid, T(tb, reject)), v\}, bid, \\
& \{+bid, bid > v \Rightarrow v = bid, -C(M.W, M, pay, win, tp)), \\
& +C(B, M, pay, win, tp), +(M.W = B)\}\rangle, M\rangle, \\
\langle M, \langle \quad & \{s(auc\_act, T(ta, done), tb < ta)\}, cfp2, \\
& \{+C(B, M, bid, T(tb, reject)), \} \\
& -won\_round, +v\}\rangle, B\rangle, \\
{}^{1}_{1}\langle M, \langle \quad & \{s(M.W = B), inform, \{+won_round\}\rangle, B\rangle \\
{}^{N-1}_{0}\langle M, \langle \ & \{s(auc\_act), \neg M.W = B\}, done, \{-auc\_act\}\rangle, B\rangle, \\
{}^{1}_{1}\langle M, \langle \quad & \{s(auc\_act), M.W = B\}, done, \{-auc\_act, \\
& +v, +win, +C(B, M, pay, tp), \\
& +C(M, B, delivered, pay, td)\}\rangle, B\rangle\}\rangle
\end{aligned}
$$

$$\begin{aligned}\pi_2 = \langle\{ \quad & s(C(B,M,pay,tp),C(M,B,delivered,pay,td))\},\pi_2,\\ & \{s(delivered)\},\\ \{\langle B,\langle\ & s(C(B,M,pay,tp),C(M,B,delivered,pay,td)),pay,\\ & \{+pay,+C(M,B,delivered,td)\}\rangle,M\rangle,\\ \langle M,\langle\ & \{s(pay,C(M,B,delivered,td),deliver,\\ & \{+delivered\}\rangle,B\rangle,\end{aligned}$$

1.6 Discussion

The usage of propositions in this approach is crucial, since it decouples the actions of their effects, allowing other actions or protocols that can have the same effects to be used instead, making the approach modular and protocols that have been predefined reusable. How successful a model is, will depend significantly on how the scenario being approached is modeled using propositions. In Section 1.3.3 some basic connections between the domain and this propositions was proposed, still how the effects of the actions are modelled is left free to the protocol designer. The options are between, on one side, the most trivial approach: making a proposition for each action that simply states that the specific action was performed, working similarly to not declarative approaches like [1] and, on the other side, very accurately selected proposition that are based on the effects several actions or protocols have in common. In the first case, the development will be very simple, but the protocols might end up rigid or difficult to recombine. The more flexible approach will take advantage of modularity, but at the same time, participating roles might be demanded to cope with very complex reasoning. A well balanced model would be the most suitable solution in most of the cases. It is clear that the minimum requirement to take advantage of this approach for modularity is to choose propositions that represent the effects of the actions, using them in all actions that semantically have the same effect.

The first part models the exhaustive state space, solving the *Frame Problem* [11] with a model of state descriptions and actions based on these descriptions and operations, to focus only on the relevant facts. Then actions are defined based on state descriptions and using composed operations that specify effects relative to their current state. This model helps to consolidate various contributions of the multiagent community using very heterogeneous models in a single comprehensive model.

1.6.1 Related work

A survey of current approaches to interaction protocol models is provided by [14]. The reader is referred to it to get more insight on proposals in the community.

The present work has as main objective to allow for formal modularity of protocols as was initially intended in FIPA [7]. The FIPA approach was

not successful due to various issues, but one of the main causes was that the FIPA speech acts were defined always from the perspective of the sending agent, which contrasts with the global perspective required for conversation protocols of this kind. This objective was, in principle, only tried by Singh et. al. [5]. As mentioned in Section 1.3.2, the same technique has been used here to model commitments, making some adaptations and modifications to integrate it concretely with the state-space model.

Timeouts are used by [2] as a "system" event. In the present approach timeouts are declared as propositions representing some facts about the environment. The concept of a global system [15] has been completely avoided, since it would not fit in an open system, as it is intended here. Timeouts have to be interpreted in the realization of roles, they are to be managed normally by the creating roles and simply represent facts about the conversation and things expected from roles.

This approach focuses only on the scope of a conversation, what is allowed, expected and demanded from participants in the conversation, it goes a bit beyond simple dialogue games [9] focusing not only in the actions that can be performed but also in their consequences. General norms and commitments of a scope outside the conversation are not treated in this model.

The present model is another alternative to give semantics to modelling techniques like in UML. For instance [4] proposes a Petri-Net for this purpose. Our approach has strong similarities, but has enabled the possibility to integrate critical concepts like commitments and the usage of propositions to connect better actions between each other and to the domain.

The way composition is approached, based on the state-action space model allows a more detailed specification of how composition can be done, compared to [12, 13] and some similar approaches, having the state-action space model serves better to achieve more detailed specifications, like propositions and cardinality constraints that give deeper insight about the conditions of the conversation.

1.7 Conclusion

A comprehensive and consolidating model for interaction protocols has been proposed. It provides a solution for modularity and composition of protocols for complex conversations. It involves many ideas proposed by the multi-agent community. The model discusses many issues about interaction protocols, the most important one is how to achieve modularity. Decoupling of actions and their effects is a fundamental advantage for this purpose. By separating this two concepts, it is possible to formalize a model that enables modularity, based on the simple fact that actions (an protocols) that share the same effect can replace, or at least be considered to replace each other. The model goes further and specifies, what has to be taken into account and how does a composition of protocols have to be.

Future work will be to extract out of a conversation protocol as presented here the set of rules and expectations for each participating role, also known as the projection of the protocol. This model has been proposed in an abstract level and shows how complex it can be to model protocols. It will be important to produce modelling tools that follow the concepts shown here in order to tame complexity and take advantage of reuse and composition.

The model proposed here can be used at different levels of abstraction, depending on how the propositions the model is based on are defined. This model can also be used to reason about the different concepts in the protocol, to help systems make decisions during conversations. They are modeled as first class objects, they can be used by designers but also by software agents to reason about them and first choose which protocol to use and second which path in the conversation to take.

References

1. Marco Alberti, Davide Daolio, Paolo Torroni, Marco Gavanelli, Evelina Lamma, and Paola Mello. Specification and verification of agent interaction protocols in a logic-based system. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pages 72–78. ACM, 2004.
2. Alexander Artikis, Marek Sergot, and Jeremy Pitt. Specifying Electronic Societies with the Causal Calculator. In *Agent-Oriented Software Engineering III*, volume 2585 of *LNCS*, pages 1–15. Springer-Verlag, 2003.
3. B. Bauer, J. P. Müller, and J. Odell. Agent UML: A Formalism for Specifying Multiagent Software Systems. In *ICSE 2000 Workshop on Agent-Oriented Software*, 2000.
4. Lawrence Cabac and Daniel Moldt. Formal semantics for auml agent interaction protocol diagrams. In *Agent-Oriented Software Engineering V*, volume 3382 of *LNCS*, pages 47–61. Springer-Verlag, 2005.
5. Nirmit Desai, Ashok U. Mallya, Amit K. Chopra, and Munindar P. Singh. Interaction Protocols as Design Abstractions for Business Processes. *Transactions on Software Engineering*, 31(12):1015 – 1027, 2005.
6. Rogier M. Van Eijk, Frank S. De Boer, Wiebe Van Der Hoek, and John-Jules Ch. Meyer. A verification framework for agent communication. *Autonomous Agents and Multi-Agent Systems*, 6(2):185–219, 2003.
7. FIPA . Foundation for Intelligent Physical Agents. On line, 2002. http://www.fipa.org.
8. Ashok U. Mallya and Munindar P. Singh. A Semantic Approach for Designing Commitment Protocols. In Rogier Van Eijk, editor, *Developments in Agent Communication*, volume 3396 of *LNAI*, pages 37–51. Springer, 2005.
9. Peter McBurney and Simon Parsons. Games that agents play: A formal framework for dialogues between autonomous agents. *J. of Logic, Lang. and Inf.*, 11(3):315–334, 2002.
10. Peter McBurney and Simon Parsons. Dialogue game protocols. *Communications in Multiagent Systems*, 2650:269–283, September 2003.
11. J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Readings in nonmonotonic reasoning*, pages 26–45, 1987.

12. Tim Miller and Peter McBurney. Using constraints and process algebra for specification of first-class agent interaction protocols. In *Engineering Societies in the Agents World VII*, volume 4457 of *LNAI*, pages 245–264, 2007.
13. Tim Miller and Peter McBurney. Annotation and matching of first-class agent interaction protocols. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 805–812, 2008.
14. Tim Miller and Jarred Mcginnis. Amongst first-class protocols. In *Engineering Societies in the Agents World VIII: 8th International Workshop, ESAW 2007, Athens, Greece, October 22-24, 2007, Revised Selected Papers*, pages 208–223. Springer-Verlag, 2008.
15. David Robertson. Multi-agent Coordination as Distributed Logic Programming. In *International Conference on Logic Programming*, volume 3132 of *LNCS*, pages 416–430. Springer-Verlag, 2004.
16. Munindar P. Singh. An Ontoloy for Commitments in Multiagent Systems. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
17. Munindar P. Singh, Amit K. Chopra, Nirmit V. Desai, and Ashok U. Mallya. Protocols for Processes: Programming in the Large for Open Systems. In *OOPSLA Companion*, volume 39 of *ACM SIGPLAN Notices*, pages 73–83. ACM, 2004.
18. Pinar Yolum and Munindar P. Singh. Commitment machines. In *ATAL '01: Revised Papers from the 8th International Workshop on Intelligent Agents VIII*, pages 235–247. Springer-Verlag, 2002.