AAMAS 2010
TORONTO

The 9th International Conference on
Autonomous Agents and Multiagent Systems
May 10-14, 2010
Toronto, Canada

Workshop 1

Agent Communication

Editors:
Wiebe van der Hoek
Gal A. Kaminka
Yves Lespérance
Michael Luck
Sandip Sen

Agent Communication 2010

In Conjunction with AAMAS 2010

Toronto, Canada, May 11, 2010

Alexander Artikis, Jamal Bentahar,
Amit K. Chopra, and Frank Dignum (Eds.)

Preface

Agent communication deals with the development of open multiagent systems. Open systems consist of autonomous and heterogeneous agents who interact in order to exchange or negotiate information, knowledge, and services. Moreover, the system configuration is dynamic: agents comes and go as they please.

Essentially, an open multiagent system is any that involves multiple organizations. Software engineering for applications involving multiple organizations is in general seeing a shift away from the formulation of processes to higher-level concepts rooted in agents. Such applications include services, collaborative design and argumentation, and e-governance. The agent communication research community has a lot to offer here: past research has yielded promising insights, abstractions, languages, and theories, especially via the specification of agent communication languages and protocols.

The principal challenge in agent communication research is to enable flexible and efficient communication among agents. This implies a focus on semantics, that is, the meaning of communication and pragmatics, that is, how to communicate. Semantics is foundational: a new conceptualization of meaning implies new abstractions and patterns, new theories, and new operationalizations in distributed settings. By contrast, traditional distributed systems research simply focuses on operational details at the expense of flexibility. Pragmatics is fundamental in designing sound and complete protocols and verifying their computational properties.

Although research in agent communication has led to useful advances, many issues remain to be addressed. Some are conceptual, for example related to the expressiveness of the abstractions in terms of modeling the application domains. Some are theoretical, for example the formalization of various notions such as strategies, correctness (compliance, interoperability, and conformance), completeness and the associated decision procedures. Some are practical, for example middleware, programming language, and tool support. Agent Communication 2010 workshop is built on previous advances towards addressing these challenging issues.

Specifically, the aim of Agent Communication 2010 workshop is to discuss research advances in: Agent Communication Languages (ACL); open agent systems and agent organizations; speech act theory; illocutionary logic; protocols and dialogues (specification and composition, verification of compliance, interoperability, conformance and other such properties); commitments, norms, and other social concepts; agent design and execution strategies (in terms of game-theoretic, BDI, learning, and other such notions) and their relation to agent communication; and programming languages and middleware to support high-level communication primitives and patterns.

We would like to thank Professor Daniel Vanderveken for the invited talk, the authors for their contributions, the members of the steering committee for their suggestions and support, and the members of the program committee for their excellent work during the reviewing phase.

March 2010

Alexander Artikis
Jamal Bentahar
Amit K. Chopra
Frank Dignum

Organization

Organizers

Alexander Artikis	National Centre for Scientific Research Demokritos, Greece
Jamal Bentahar	Concordia University, Canada
Amit K. Chopra	University of Trento, Italy
Frank Dignum	Utrecht University, The Netherlands

Program Committee

Guido Boella	University of Torino, Italy
Marco Colombetti	Politecnico di Milano, Italy
Nirmit Desai	North Carolina State University, USA
Nicoletta Fornara	Università della Svizzera Italiana, Switzerland
Koen Hindriks	Delft University of Technology, The Netherlands
Marc-Philippe Huget	University of Savoie, France
Michael N. Huhns	University of South Carolina, USA
Andrew J. I. Jones	King's College London, UK
Peter McBurney	University of Liverpool, UK
Tim Miller	, Melbourne University, Australia
Bernard Moulin	Laval University, Canada
Xavier Parent	University of Luxembourg, Luxembourg
Michael Rovatsos	University of Edinburgh, UK
Munindar P. Singh	North Carolina State University, USA
Paolo Torroni	University of Bologna, Italy
Rogier van Eijk	Utrecht University, The Netherlands
Wamberto Vasconcelos	University of Aberdeen, UK
Pinar Yolum	Bogazici University, Turkey

Steering Committee

Frank Dignum	Utrecht University, The Netherlands
Andrew J. I. Jones	King's College London, UK
Munindar P. Singh	North Carolina State University, USA

Additional Reviewers

Valerio Genovese
Wojtek Jamroga

Table of Contents

Invited Talk: On the Foundations of a Formal Discourse Pragmatics 1
Daniel Vanderveken

Constraints among Commitments: Regulative Specification of Interaction Protocols 2
Matteo Baldoni, Cristina Baroglio, and Elisa Marengo

Protocol Refinement: Formalization and Verification 19
Scott N. Gerard and Munindar P. Singh

Counter-proposal: A Multi-Agent Negotiation Protocol for Resolving Resource Contention in Open Control Systems 37
Jan Corfixen Sørensen and Bo Nørregaard Jørgensen

Verifying Conformance of Commitment Protocols via Symbolic Model Checking 53
Mohamed El-Menshawy, Jamal Bentahar, Wei Wan, and Rachida Dssouli

The Logic of Conversation: From Speech Acts to the Logic of Games 73
Michel A. Paquette

Author Index 91

Invited Talk:
On the Foundations of a Formal Discourse Pragmatics

Daniel Vanderveken[1]

University of Quebec, Trois-Rivières ,
daniel_vanderveken@UQTR.CA,
http://www.uqtr.ca/~vandervk/cvdveng.htm

Could we enrich speech-act theory to deal with discourse? Wittgenstein and Searle pointed out difficulties. Most conversations lack a conversational purpose, their background is indefinitely open, they can contain irrelevant and infelicitous utterances, they require collective intentionality, etc. In my view, the primary aim of discourse pragmatics is to analyze the structure and dynamics of language-games whose type is provided with an internal conversational goal. Such games that are indispensable to any kind of discourse have a descriptive, deliberative, declaratory or expressive point. So are exchanges of salutations, interrogations, negotiations and contracts. Logic can analyze felicity-conditions of such collective illocutions because they are conducted according to systems of constitutive rules. Speakers often speak non-literally or non-seriously. The units of conversation are attempted illocutions whether literal, serious or not. I will show how to construct speaker-meaning from sentence-meaning, conversational background and maxims. Like Montague, I believe that we need the resources of formalisms (proof-, model- and game-theories) and logic in pragmatics. I will explain how to further develop intensional and illocutionary logics, the logic of attitudes and of action in order to characterize our ability to converse. I will compare my approach to others (Austin, Belnap, Grice, Montague, Searle, Sperber and Wilson, Kamp, Wittgenstein) as regards hypotheses, methodology and issues. I will also deal with the nature of intelligent dialogues between man and machines in A.I.

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, 6, 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, 6], 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, 6].

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, 6] 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.

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*

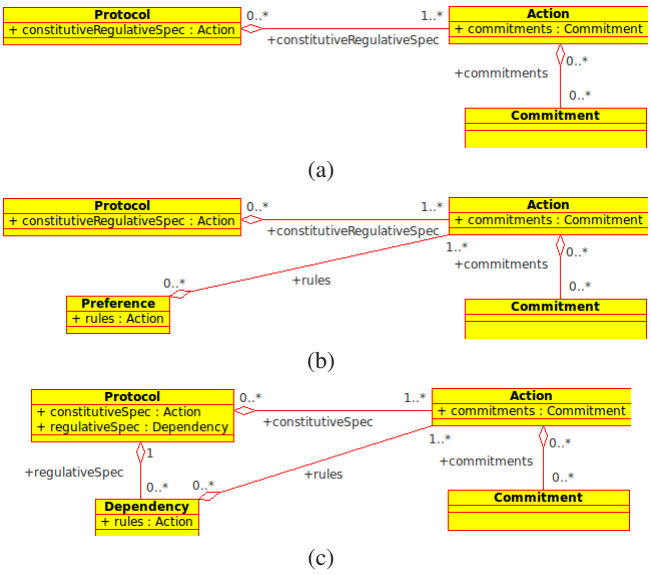


Fig. 1. (a) Chopra and Singh's implementation model: regulative specifications based on actions; (b) Mallya and Singh's model: adding preferences on actions; (c) Singh's dependencies among events.

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, 6, 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.

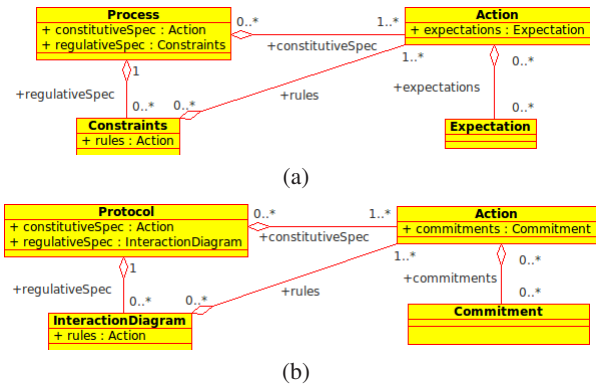


Fig. 2. (a) ConDec model: regulative specification given by means of constraints on actions, an extension supplies an expectation-based semantics for actions; (b) Fornara and Colombetti's model: regulative specification given by interaction diagrams defined on actions.

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 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].

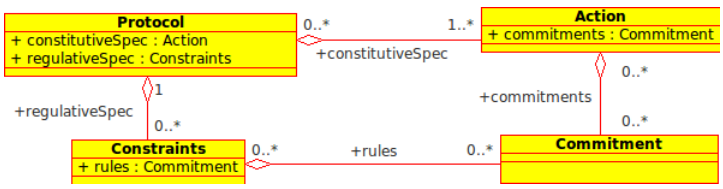


Fig. 3. Our proposal: decoupling between constitutive (actions) and regulative (constraints) specifications.

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 [6], 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:

$A \rightarrow (Action$ **means** $Operation)^+$
$Action \rightarrow protocolAction([paramList])$
$Operation \rightarrow Op(commitment) \mid fact \mid Operation \wedge Operation$
$Op \rightarrow$ CREATE | DELETE | DISCHARGE | RELEASE | DELEGATE | 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 [6, 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** 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:

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

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, $C(i, p, assign\_task) \rightarrowtriangle\!\!\bullet\ (refused\_task \vee C(p, i, solve\_task))$ 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	$a \rightarrowtriangle\!\!\bullet\ b$	$\neg b \mathsf{U} a$	$a \not\rightarrowtriangle\!\!\bullet\ b$	$\neg a \mathsf{U} b$
Cause	$a \bullet\!\!-\!\!\triangleright\!\!\bullet\ b$	$a \bullet\!\!-\!\!\triangleright b \wedge a \rightarrowtriangle\!\!\bullet\ b$	$a \bullet\!\!\not-\!\!\triangleright\!\!\bullet\ b$	$a \bullet\!\!\not-\!\!\triangleright b \wedge a \not\rightarrowtriangle\!\!\bullet\ b$
Premise	$a \triangleright\!\!- b$	$\Box(\bigcirc b \supset a)$	$a \triangleright\!\!\not- b$	$\Box(\bigcirc b \supset \neg a)$
Immediate response	$a \rightarrow\!\!\triangleright b$	$\Box(a \supset \bigcirc b)$	$a \not\rightarrow\!\!\triangleright b$	$\Box(a \supset \bigcirc \neg b)$

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 $a \twoheadrightarrow\!\!\bullet\ b$ is equivalent to $b \twoheadrightarrow\!\!\bullet\ a$.

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 $\Diamond a \wedge \neg\Diamond b$, 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 $\Diamond(\bigcirc b \wedge \neg a)$ 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** CREATE$(C(i,p,assigned_task))$
(b) $propose(p,i)$ **means** CREATE$(C(p,i,solved_task))$
(c) $refuse(p,i)$ **means** $refused\_task \wedge$ RELEASE$(C(i,p,assigned_task))$
(d) $accept(i,p)$ **means** $assigned\_task$
(e) $reject(i,p)$ **means** $rejected\_proposal \wedge$ DELETE$(C(i,p,assigned_task)) \wedge$ RELEASE$(C(p,i,solved_task))$
(f) $inform\_done(p,i)$ **means** $solved\_task$
(g) $failure(p,i)$ **means** $failed\_task \wedge$ 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 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: $C(i,p,assigned\_task) \bullet\!\!\rightarrow\!\!\bullet\ C(p,i,solved\_task)$ XOR $refused\_task$
c2: $C(p,i,solved\_task) \bullet\!\!\rightarrow\!\!\bullet\ rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \bullet\!\!\rightarrow\!\!\bullet\ solved\_task$ 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

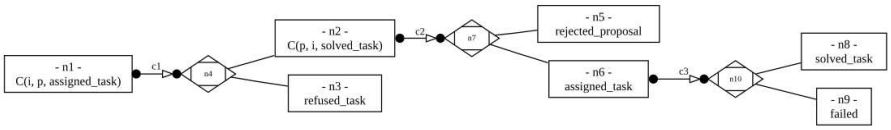


Fig. 4. Regulative specification of the Contract Net Protocol.

states its intention to solve the task. This is represented by the fact that the node $n1$ is connected to the nodes $n2$ ($C(p, i, solved\_task)$), 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: $C(p, i, solved\_task) \bullet\!\!\rightarrow rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \bullet\!\!\rightarrow solved\_task$ XOR $failed$

Instead, see Fig. 5(b), if a *response* ($\bullet\!\!\rightarrow$) 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: $C(i, p, assigned\_task) \bullet\!\!\rightarrow C(p, i, solved\_task)$ XOR $refused\_task$
c2: $C(p, i, solved\_task) \bullet\!\!\rightarrow rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \bullet\!\!\rightarrow solved\_task$ 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: $C(i, p, assigned\_task) \bullet\!\!\rightarrow C(p, i, solved\_task)$ XOR $refused\_task$
c2: $C(p, i, solved\_task) \rightarrow\!\!\!\!\rightarrow rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \bullet\!\!\rightarrow solved\_task$ 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* ($\rightarrow\!\!\bullet$) instead of being a *cause* ($\bullet\!\!\rightarrow\!\!\bullet$, in Fig. 4). We have further modified the CNP by changing the constraint $c2$ in Fig. 4, which is now an *immediate response* ($\rightarrow\!\!\!\!\rightarrow$), 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: $C(i, p, assigned\_task) \rightarrow\!\!\bullet\ C(p, i, solved\_task)$ XOR $refused\_task$
c2: $C(p, i, solved\_task) \rightarrow\!\!\!\!\rightarrow rejected\_proposal$ XOR $assigned\_task$
c3: $assigned\_task \bullet\!\!\rightarrow solved\_task$ 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

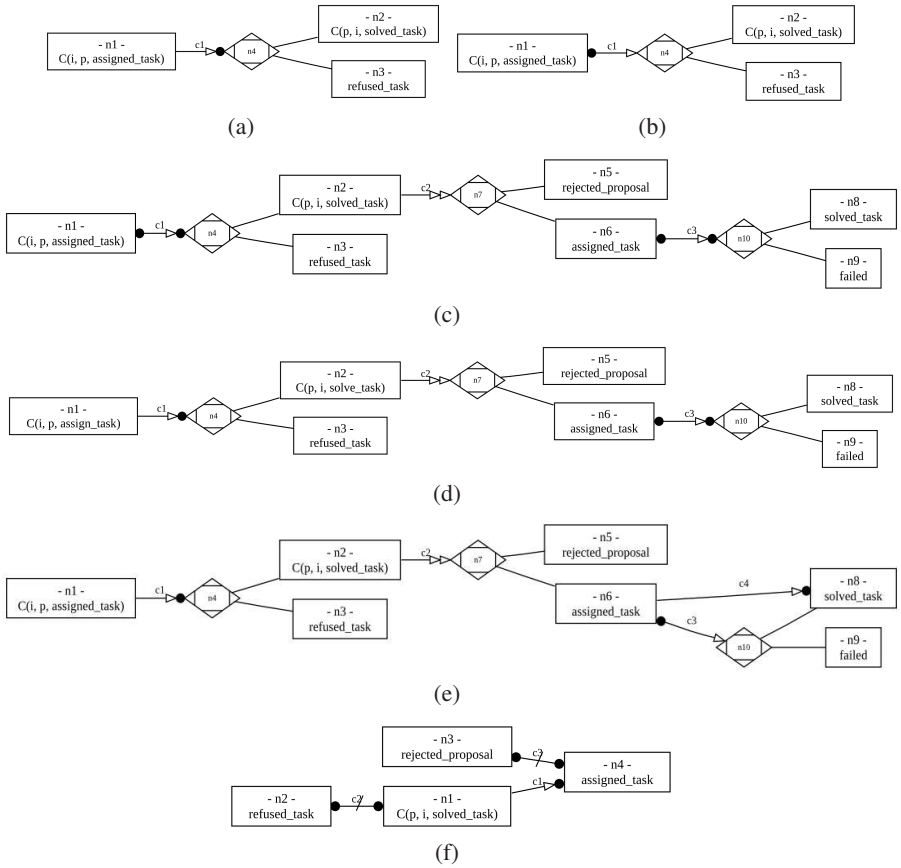


Fig. 5. (a) Lazy Participant; (b) Zealus Participant; (c) Contract Net with Immediate Answer; (d) Call for Bid; (e) Call for Bids with Anticipated Failure; (f) Soft Call for Bids.

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:

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

The changes concern the constraints after node $n6$. In the new version, instead of having simply a *cause* constraint, we have a *response* ($\bullet\!\!\rightarrowtriangle$). 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: $C(p, i, solved\_task) \multimap\!\bullet\ assigned\_task$
c2: $refused\_task \bullet\!\!\not\multimap C(p, i, solved\_task)$
c3: $rejected\_proposal \bullet\!\!\not\multimap assigned\_task$

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 $\bullet\!\!\not\multimap$) 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, 6, 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*, $\rightarrowtriangle$) 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 ($\multimap\!\!\!\triangleright$) 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 pro-

duce 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.

Protocol Refinement: Formalization and Verification

Scott N. Gerard and Munindar P. Singh

Department of Computer Science
North Carolina State University
Raleigh, NC 27695 USA
{sngerard,singh}@ncsu.edu

Abstract. A proper definition of protocols and protocol refinement is crucial to designing multiagent systems. Rigidly defined protocols can require significant rework for even minor changes. Loosely defined protocols can require significant reasoning capabilities within each agent. Protocol definitions based on commitments is a middle ground.
We formalize a model of protocols consisting of agent roles, propositions, and commitments. We define protocol refinement between a superprotocol and a subprotocol by mapping superprotocol elements to corresponding subprotocol elements. Mapping protocol commitments depends on a novel operation called serial composition. We demonstrate protocol refinement.

1 Introduction

We focus our attention on service engagements between businesses and customers (B2B and B2C) over the Internet. In current practice, such engagements are defined rigidly and purely in operational terms. Consequently, the software components of the business partners are tightly coupled with each other, and depend closely on the engagement specification. Thus the business partners interoperate, but just barely. Even small changes in one partner's components must be propagated to others, even when such changes are not consequential to the business being conducted. Alternatively, in current practice, humans carry out the necessary engagements manually with concomitant loss in productivity.

In such an environment, if there were no mechanisms to structure inter-agent communication, agent implementations would need to handle a wide variety of communication making agent implementations complex with sophisticated reasoning capabilities as each interaction would be unique and customized. It would be difficult to predict *a priori* whether two agents could interoperate.

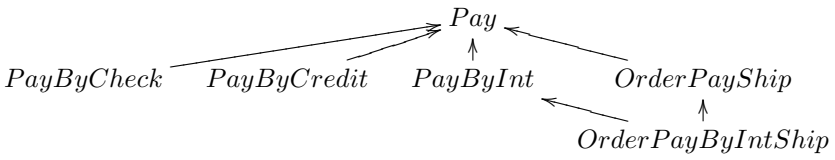
Protocols, as we understand them, provide a happy middle between rigid automation and flexible manual execution. Using protocols as a mechanism to structured communication, agent implementations can be less sophisticated. Protocol designers design and analyze protocols for desirable properties. Agents can publicly declare the protocols in which they can participate making it easier to find agents with whom to interoperate.

Protocols are a way to standardize communication patterns so agents can be used in many different multiagent interactions. Consider the simple protocol *Pay* consisting of a single action where a payer pays a payee. And consider protocol *OrderPayShip* where a

buyer and a seller agree to a price for a particular good, the buyer pays the seller and the seller ships the good to the buyer. The payer and payee roles in *Pay* should correspond to the buyer and seller roles in *OrderPayShip*. The payment in *Pay* should correspond to the payment in *OrderPayShip*. Therefore, we expect *OrderPayShip* refines *Pay*.

Suppose protocol *PayByInt* (pay by intermediary) is introduced where the payer first pays a middleman, who in turn pays the payee. Since both *Pay* and *PayByInt* send a payment from the payer to the payee, we expect *PayByInt* refines *Pay*. Similar arguments imply *PayByCheck*, *PayByCredit*, and others also refine *Pay*. If *PayByInt* becomes popular, we would like to construct a new protocol *OrderPayByIntShip*, which is just like *OrderPayShip*, except payments are made using *PayByInt* rather than *Pay*.

This diagram shows the expected refinement relationships between various protocols.



We are working to implement refinement checking via the MCMAS model checking [2] to handle complex protocols like those found in real service engagements.

Contributions

The main contributions of this paper are a definition of a refinement relation between two protocols, the notion of covering commitments, and the definition of serial composition of commitments. It describes why commitment-based protocols are more flexible than traditional computer protocols using the idea of multiple states of completion.

Organization

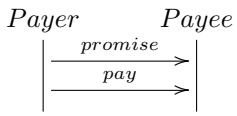
Section 2 introduces our running examples. Section 3 describes background material on commitments. Section 4 describes our intuitions and framework for protocol refinement, covering commitments, and serial composition of commitments. Section 5 briefly describes our intentions for implementing refinement checking with the MCMAS model checker. Section 6 demonstrates refinement on an example. Section 7 evaluates our approach. Section 8 describes other works and our future directions.

2 Examples

We introduce four running examples. *Pay* and *PayByInt* are basic payment protocols while *OrderPayShip* and *OrderPayByIntShip* are order protocols involving payments.

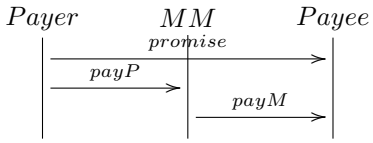
2.1 Pay

Pay is a basic payment protocol between a payer and a payee. If the payer chooses to do so, it commits to pay the payee by action promise. Then, at some later point, it sends a single payment directly to the payee. This sequence diagram describes the interaction.



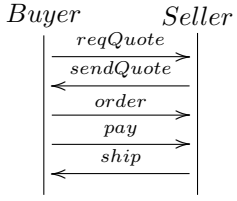
2.2 PayByInt

In protocol *PayByInt* (pay by intermediary), if the payer chooses to do so, it commits to pay the payee with promise. It then pays by sending a payment *indirectly* to the payee. The payer first pays a middleman, who in turn pays the payee. We assume the middleman commits to perform *payM* if payer performs *payP*. This sequence diagram shows a typical interaction, but sequence diagrams document only one message run. Other runs may also be valid. In this case, it is acceptable for the middleman to be generous and execute *payM* before *payP*.



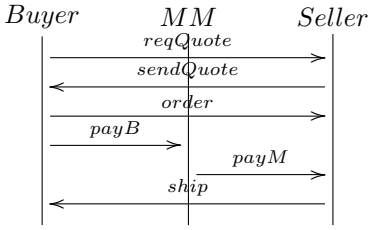
2.3 OrderPayShip

In *OrderPayShip* a buyer orders goods from a seller. The buyer requests a price quote for a good from the seller. The seller sends the price quote along with its commitment to ship the good if the buyer orders. The buyer can accept the offer by ordering and making its commitment to pay for the good if it orders. The seller can ship first, or the buyer can pay first.



2.4 OrderPayByIntShip

Protocol *OrderPayByIntShip* is similar to *OrderPayShip* except the payer uses *PayByInt* for payment. This introduces a new middleman role.



3 Background

3.1 Commitments

Commitments are a formal and concise method of describing how agent roles commit to perform future actions. We extend previous commitment definitions [5] in two ways. First, we allow both debtors and creditors to be sets of roles. This handles situations where a chain of debtors and intermediaries must all act to fulfill a commitment, and where a chain of creditors and intermediaries all need to know whether a commitment is satisfied. Second, we implement prior uses of **delegate** and **assign** with a single, new **transfer** operation.

Definition 1. *A commitment is an object*

$$\mathsf{C}_{\{debtors\},\{creditors\}}(\text{ant}, \text{csq}) \tag{1}$$

where debtors and creditors are sets of roles, ant is the antecedent, and csq is the consequent. When a commitment is active, the debtors are conditionally *committed to the creditors. Once ant becomes true, the debtors are* unconditionally *committed to make csq true at some point in the future.*

The valid operations on commitments are

- ***create**, performed only by debtors, creates a new commitment and makes it active.*
- *When the antecedent becomes true, the commitment is implicitly converted to an unconditional commitment.*
- *When the consequent becomes true, the commitment is implicitly satisfied and no longer active. Typically the consequent become true only after the antecedent becomes true, but this is not required.*
- ***transfer**, performed by either debtors or creditors, ends the current commitment and marks it as transferred to another commitment and no longer active.*
- ***release**, performed only by creditors, releases the debtors from their commitment. The commitment is released and no longer active.*
- ***cancel**, performed only by debtors, cancels the debtors' commitment. The commitment is violated and no longer active.*

A commitment is always in one of these states.

- *inact: the initial state;*
- *cond: after **create** with ant false, csq false, and no other operations;*
- *uncond: after **create** with ant true, csq false, and no other operations;*
- *sat: after **create** and csq true;*
- *xfer: after **create** and **transfer**;*
- *rel: after **create** and **release**; and*
- *can: after **create** and **cancel**.*

A commitment in state *sat*, *xfer*, *rel*, or *can* is said to be *resolved*.

For unconditional commitments, the debtors are committed to eventually make *csq* true. If the debtors fail, responsibility can be *several* (each debtor is responsible for just its portion), *joint* (each debtor is fully responsible for the entire commitment), or *joint and several* (the creditors hold one debtor fully responsible, who then pursues other debtors). We use *several* responsibility.

We note that contracts are built from multiple commitments; each party commits to perform the actions for which it is responsible. So while contracts are created by both debtors and creditors, commitments are created only by debtors.

Before a commitment's **create** (state *inact*), the commitment has no force. A created commitment (state *cond*) is conditionally committed. Unconditional commitments must eventually resolve to state *sat*, *xfer*, *rel*, or *can*. It is possible for the consequent to become true before the antecedent. While unusual, debtors have the option to act before being required to do so. Debtors are discouraged from **cancel**, but circumstances may require it, with consequences handled outside the current mechanisms.

In *Pay*, we represent the payer's commitment to paying the payee as

$$\mathsf{C}_{\text{Payer,Payee}}(\mathit{promise}, \mathit{pay})$$

In *PayByInt*, we represent the payer's and middleman's combined commitment as

$$\mathsf{C}_{\{\text{Payer,MM}\},\text{Payee}}(\mathit{promise}, pay_P \wedge pay_M)$$

Previous commitment descriptions allow debtors to **delegate**, or creditors to **assign**, a commitment to another role. Both terminate the existing commitment and create a new commitment with modified roles. We model these operations as a **transfer** operation which terminates the existing commitment plus a separate **create** of a new commitment.

$$\begin{aligned}\textbf{delegate}(\mathsf{C}_i, debt') &= \textbf{transfer}(\mathsf{C}_i) \wedge \textbf{create}(\mathsf{C}_{d'})\\ \textbf{assign}(\mathsf{C}_i, cred') &= \textbf{transfer}(\mathsf{C}_i) \wedge \textbf{create}(\mathsf{C}_{c'})\end{aligned}$$

where $\mathsf{C}_i = \mathsf{C}_{\text{debt,cred}}(\mathit{ant}, \mathit{csq})$, $\mathsf{C}_{d'} = \mathsf{C}_{\text{debt',cred}}(\mathit{ant}, \mathit{csq})$ and $\mathsf{C}_{c'} = \mathsf{C}_{\text{debt,cred'}}(\mathit{ant}, \mathit{csq})$. Since **delegate** and **assign** have essentially the same effect, **transfer** captures the essence of both and somewhat simplifies the definition of commitments.

3.2 Unconditional Commitments Must Resolve

We require debtors must eventually resolve all their unconditional commitments, even if that is **cancel**.

Model checkers have fairness constraints which eliminate unfair paths from consideration. Fairness constraints are typically used to eliminate unfair scheduler paths. We use fairness constraints to eliminate paths where agents never resolve their unconditional commitments. This is a constraint on agent implementations, not a constraint on the protocol itself.

4 Framework

4.1 Protocol Basics

A protocol specification language needs to describe all participating agent roles, the actions they can perform, and any constraints (guards) on their actions. Agents send *message actions* to each other. Message actions are an agent-level concept. Below we decompose message actions into an unordered set of *basic actions*.

We model protocols using CTL. We consider runs of basic actions which generate state runs as is traditional for model checking. An action run is modeled as

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots$$

When we compare points in time and actions, "$s_i < a_j < s_k$" means $i \leq j < k$. When we compare two actions, "$a_i < a_j$" means $i < j$.

While support for looping protocols is desirable, we simplify the initial problem and do not consider them here. We hope to extend our work to cover looping protocols later.

4.2 Messages and Guards

Both message actions and basic actions can have guard conditions. An action is enabled for execution only when its guard is true.

Some protocols must constrain message orders. For example, (1) when one message provides a value required by another message, or (2) due to regulatory requirements (you must show a valid ID before boarding an airplane). An action's guard is written

$$guard < action$$

which means *action* can occur in a state only if *guard* is true in that state. The action is not required to execute when the guard is true. Guards $g_1 < action$ and $g_2 < action$ can be combined into $g_1 \wedge g_2 < action$.

Guards for message actions have been used elsewhere including MCMAS. We also introduce guards for basic actions since the protocol designer may need to constrain the ordering of basic actions in subprotocols. The designer can specify multiple guards for an action. The complete guard condition for a basic action includes any guards for the basic action as well as any guards for its containing message action. We expect tooling to combine all designer-specified guards into a single guard expression.

Example: A protocol designer might require

$$reqQuote \wedge sendQuote < order$$

4.3 Multiple Stages of Completion

Commitments evolve through four stages; proposition evolve through only two. The occurrence of a $prop = value$ basic action divides time into two stages: before and after the basic action.

A commitment evolves through four stages: (1) before creation (inactive), (2) conditionally committed (cond), (3) unconditionally committed (uncond), and (4) resolved.

$$\xrightarrow{inact} \textbf{create} \xrightarrow{cond} ant \xrightarrow{uncond} csq \xrightarrow{resolved}$$

Commitments increase protocol flexibility, because guards can specify "partially performed" actions. A protocol can make progress sooner if an action's guard specifies one of the first three stages.

Example: Using proposition *ship*, *OrderPayShip* can guard *pay* based only on the two stages of *ship*. The decision is "all" (*ship*ped) or "nothing" (not *ship*ped).

$$ship < pay \tag{2}$$

Using commitments, the protocol can guard *pay* based on any of the four commitment stages. A guard can enable *pay* as soon as the debtor has committed to make *ship* true.

$$\textbf{create}(\mathsf{C}_{\text{Seller,Buyer}}(pay, ship)) < pay \tag{3}$$

A protocol framework that includes commitments is inherently more flexible than traditional computer protocol frameworks. Where traditional protocol frameworks operate on an all-or-nothing basis with just two stages of completion, agent-based frameworks can operate based on four stages of completion (commitments have four stages of completion). Basing a decision on a completed action provides essentially no risk to a creditor. But commitments allow more flexible enactments because creditors can also base their decisions on the promises of the debtors which are partial or intermediate stages of completion. While creditors assume more risk in doing so, they assume less risk than acting without debtor promises.

4.4 Every Sub-run is a Super-run

According to the Liskov substitution principle [1], if $\phi(p)$ is a property provable about objects p of type P, then $\phi(q)$ should be true for objects q of type Q when Q is a subtype of P. We apply this principle to protocols. If $\phi(x)$ holds for a superprotocol, then it must also hold for its subprotocols. For example, let $\phi(x)$ be the property that action $order$ precedes action *pay*. $\phi(p)$ will be true of some runs, but not of other runs.

Subprotocols must satisfy every superprotocol property $\phi(x)$, but they can satisfy additional properties (x). Every subprotocol run must satisfy both $\phi(x)$ and (x), so there are fewer subprotocol runs than superprotocol runs because (x) eliminates some runs $\phi(x)$ does not.

Definition 2 (Protocol Refinement). *Every subprotocol basic action run must be a superprotocol basic action run.*

Our definition of protocol refinement does not imply that agents that can participate in a superprotocol can necessarily participate in a subprotocol unchanged. Agents work with message actions, but our definition for run comparison is based on basic actions. In our model, agents may need to be modified to participate in subprotocols. For example, an agent capable of participating in a basic payment protocol may need to change to handle payments via check or credit card.

4.5 Time Expansion

Basic actions which are concurrent at a high-level of abstraction (superprotocol) may not be concurrent at a lower-level of abstraction (subprotocol). When a superprotocol's message action is refined in the subprotocol, its basic actions can spread out over time (no longer concurrent).

Decomposing a message action into multiple basic actions requires splitting action intervals and creating additional time points in the run (expanding time).

4.6 Decomposition

A message action may have multiple effects. To better understand and characterize a *message action*, we decompose each message action into an unordered set of one or more *basic actions*. Each basic action has well-defined semantics and is useful for analyzing and understanding the meanings of message actions.

We consider two different kinds of basic actions. Setting the value of a proposition to true or false is a propositional basic action. Each of the commitment operations **create**, **transfer**, **release**, and **cancel** are commitment basic actions.

Example: The seller's message action *sendQuote* in *OrderPayShip* decomposes into two basic actions. It sets the propositional *sendQuote* fluent to true to record the fact that it responded to the buyer. And it creates a commitment that the seller will ship if the goods are ordered.

See the discussion for diffusion below for the exact guard conditions.

4.7 Mapping

Since superprotocols represent a higher-level abstractions than subprotocols, the differences in levels must be addressed. There is often no one-to-one correspondence between superprotocol and subprotocol elements. Protocol elements must be mapped between the two protocols to compare them. Since subprotocols typically contain more detail than superprotocols, we map every single superprotocol element (role, proposition or commitment) to an expression of one or more corresponding subprotocol elements. Subprotocols may contains sub-elements that do not correspond with any superelement.

Example: an *openAccount* basic action in a high-level, banking superprotocol, might decompose to multiple basic actions in a low-level subprotocol: *checkIdentity*, *checkCredit*, *createAccount*, and *notifyCustomer*.

Example: the *pay* basic action in *Pay* maps to the *payP* and *payM* expression in *PayByInt*.

While it would be desirable for the mapping between superprotocols and subprotocols to be automatically determined, there can be multiple, distinct mappings between some protocol pairs. In the mapping between *Pay* and *PayByInt*, one mapping groups the Middleman with the Payer, making the Middleman work on behalf of the Payer. Another mapping groups the Middleman with the Payee, making the Middleman work on behalf of the Payee. Both interpretations are valid.

Every super-basic-action is mapped to an expression of sub-basic-actions. There is one mapping function for each super-basic-action. The mapping function for basic propositions is a boolean expression of sub-basic-propositions. The mapping function for super-basic-commitments is a serial composition of sub-basic-commitments.

$$super \mapsto map_{super}(sub_1, sub_2, \dots)$$

4.8 Projection

Let SubOnly be the set of basic actions that occur only in the subprotocol (basic actions not in the superprotocol and not in a mapping expression). Ignore all SubOnly basic actions in the subprotocol during run comparison.

This means we compare all basic actions in the superprotocol with just the matching basic actions in the subprotocol. The subprotocol is free to include additional basic actions that are unknown to the superprotocol.

4.9 Diffusion

Consider the case where a super-basic-action p maps to two sub-basic-actions $p \mapsto q_1 \wedge q_2$. For p to occur at the same time point in both sub-run and super-run, p must occur at the same time as the *last* of q_1 and q_2. For conjunction, the other sub-actions can occur at any *earlier* and possibly non-adjacent points in the run. In the case where $p \mapsto q_1 \vee q_2$, then p must occur at the same time point as the *first* of q_1 or q_2. For disjunction, the other sub-actions can occur at any *later* and possibly non-adjacent points in the run.

Example: in *OrderPayShip*, the buyer's order action is composed of the two basic actions of setting an *order* proposition and creating a commitment. In refinements of *OrderPayShip* these two basic actions can occur at different points in time.

When a sub-action guard is true, we require the corresponding super-action guard to also be true. Otherwise, the subprotocol could perform the sub-action while the superprotocol can not perform its corresponding super-action. The model checker tests this using formula

$$\mathbf{AG}(sub.guard \rightarrow super.guard) \tag{4}$$

We represent the diffusion condition as a guard condition. Since runs are serialized basic actions, we do not need to consider multiple basic actions occurring simultaneously.

If $super \mapsto sub_1 \wedge \dots \wedge sub_i$, the super-basic-action's guard must be true when the last of the sub-basic-action's guard is true.

$$sub_i.effguard = sub_i.baguard \wedge [sub_i.maguard \vee \neg(\bigwedge_{j \neq i} sub_i.occurred)]$$

The sub-basic-action's effective guard is built from two pieces. Any guard specifically applied to the sub-basic-action ($sub_i.baguard$) must always be true for the basic action to fire. Also, the message action guard must be true, or this must not be the last sub-basic-action. This effective guard is checked with equation 4.

If $super \mapsto sub_1 \vee \cdots \wedge sub_i$, the super-basic-action's guard must be true when the first of the sub-basic-action's guards is true.

$$sub_i.effguard = sub_i.baguard \wedge [sub_i.maguard \wedge (\bigwedge_i \neg sub_i.occurred)]$$

The super-basic-action's effective guard is again built from two pieces. Any guard specifically applied to the sub-basic-action ($sub_i.baguard$) must always be true for the basic action to fire. Also, the message action guard must be true when no sub-basic-actions have fired. This effective guard can be checked with equation 4.

Example: $pay \mapsto pay_P \wedge pay_M$ generates the effective guards

$$pay_P.effguard = true \wedge [(\textbf{create}(\mathsf{C}_{payM}) \wedge promised) \vee \neg payM]$$
$$pay_M.effguard = true \wedge [(\textbf{create}(\mathsf{C}_{payM}) \wedge promised) \vee \neg payP]$$

Since neither pay_P nor pay_M have specific basic action guards ($baguard = true$), the message action guard is **create** and $promised$, and they each require the other basic action must not have fired.

4.10 Run Comparison

Figure 1 illustrates the steps required for refinement. Compare one super message action run and one sub message action run, as follows.

1. Begin with the superprotocol's and subprotocol's message action runs.
2. Decompose each run of message actions to a run of basic actions.
3. Map every super-basic-action to its expression as sub basic actions.
4. Serialize the basic actions sets in any order consistent with the basic action guards.
5. Ignore (project out) SubOnly basic actions in the sub run.
6. Diffuse basic actions.
7. Find any ordering of basic actions in both sub and super runs that satisfy all the ordering constraints.
8. If the two state runs are the same, the runs match.

4.11 Commitment Strength

We need a way to compare two commitments, in particular, to compare a commitment in the subprotocol with a commitment in the superprotocol.

Definition 3 (Commitment Strength). *A commitment C_S is stronger than a commitment C_W, written $\mathsf{C}_S \geq \mathsf{C}_W$, iff*

$$\mathsf{C}_W.debt \subseteq \mathsf{C}_S.debt \tag{5}$$
$$\mathsf{C}_W.cred \subseteq \mathsf{C}_S.cred \tag{6}$$
$$\mathsf{C}_W.ant \vdash \mathsf{C}_S.ant \tag{7}$$
$$\mathsf{C}_S.csq \vdash \mathsf{C}_W.csq \tag{8}$$

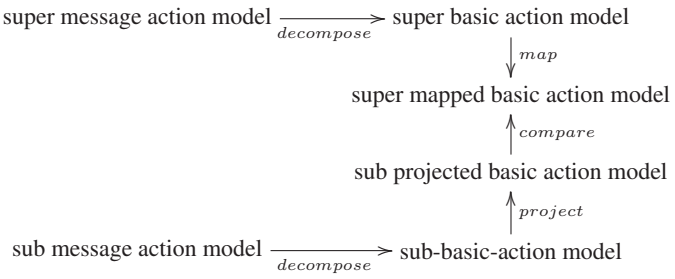
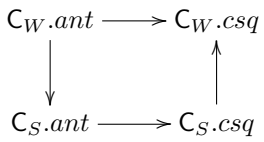


Fig. 1. Relationships between models

where $\subseteq$ is subset and $\vdash$ is derives.

Equations (5) and (6) allow additional roles in the subprotocol to be involved as both debtors and creditors. Equations (7) and (8) are based on the following diagram.



If C_S is stronger than C_W, then both side implications are true by equations (7) and (8). If C_S is satisfied, then the bottom implication is true. Then C_W, the top implication, will be true.

Theorem 1. *Commitment strength is reflexive and transitive.*

Proof. Reflexive and transitivity are immediate from the definition.

Example: $\mathsf{C}(order \vee freeCoupon, ship) \geq \mathsf{C}(order, ship)$ since the stronger commitment commits at least when $order$ is true. It also commits when $freeCoupon$ is true.

Example: $\mathsf{C}(order, ship \wedge expressDelivery) \geq \mathsf{C}(order, ship)$ since the stronger commitment commits to the additional consequent $expressDelivery$.

4.12 Covering

When a commitment C_{sub} in a subprotocol is stronger than a commitment C_{super} in a superprotocol we say the sub-commitment "covers" the super-commitment. We require every super-commitment must be covered by some sub-commitment. This guarantees that the super-commitment is satisfied whenever the sub-commitment is satisfied.

A single subprotocol commitment clearly covers an identical superprotocol commitment. A sub-commitment with more debtors, more creditors, a weaker antecedent, or a stronger consequent covers a super-commitment. However, this is not always enough. We allow serial composition of commitments as another way to cover commitments.

4.13 Intermediaries

Whereas two roles may communicate directly with each other using a single message action in a protocol at a high-level of abstraction, there is a natural tendency for message communication to pass through multiple intermediary roles as that protocol is refined to lower-levels of abstraction. Protocol refinement must properly handle intermediaries. One super-proposition could map to an expression of multiple sub-propositions, each controlled by different roles (intermediaries). One super-commitment could be fulfilled through multiple intermediaries. Super-elements must be able to span intermediaries.

Example: the *pay* action in *Pay* becomes the two distinct *payP* and *payM* actions in the subprotocol *PayByInt*. These two actions must be in different message actions in *PayByInt*, because they are performed by different roles.

Example: commitments can chain through multiple intermediaries. When *PayByInt* refines *Pay*, the single commitment from the payer to the payee in *Pay* does not appear explicitly in *PayByInt*. Rather, the subprotocol *PayByInt* has two separate commitments that form a chain passing through the middleman. That chain commits the payer to pay the payee.

4.14 Serial Composition

We also need a mechanism for commitments to span intermediaries. Previous commitment formulations [6] included the idea of commitment chaining, but we formalize this in a new way as "serial composition" of commitments. Serial composition computes a result commitment from a chain of commitments.

Definition 4 (Serial Composition). *Two commitments* C_A *and* C_B *are combined into a resultant commitment* $\mathsf{C}_\oplus = \mathsf{C}_A \oplus \mathsf{C}_B$ *if the operation is well-defined*

$$\mathsf{C}_A.ant \wedge \mathsf{C}_A.csq \vdash \mathsf{C}_B.ant \tag{9}$$

Then $\mathsf{C}_\oplus$ *is defined as*

$$\mathsf{C}_\oplus.debt := \mathsf{C}_A.debt \cup \mathsf{C}_B.debt \tag{10}$$
$$\mathsf{C}_\oplus.cred := \mathsf{C}_A.cred \cup \mathsf{C}_B.cred \tag{11}$$
$$\textbf{\textit{create}}(\mathsf{C}_\oplus) := \textbf{\textit{create}}(\mathsf{C}_A) \wedge \textbf{\textit{create}}(\mathsf{C}_B) \tag{12}$$
$$\mathsf{C}_\oplus.ant := \mathsf{C}_A.ant \tag{13}$$
$$\mathsf{C}_\oplus.csq := \mathsf{C}_A.csq \wedge \mathsf{C}_B.ant \wedge \mathsf{C}_B.csq \tag{14}$$
$$\textbf{\textit{transfer}}(\mathsf{C}_\oplus) := \textbf{\textit{transfer}}(\mathsf{C}_A) \vee \textbf{\textit{transfer}}(\mathsf{C}_B) \tag{15}$$
$$\textbf{\textit{release}}(\mathsf{C}_\oplus) := \textbf{\textit{release}}(\mathsf{C}_A) \vee \textbf{\textit{release}}(\mathsf{C}_B) \tag{16}$$
$$\textbf{\textit{cancel}}(\mathsf{C}_\oplus) := \textbf{\textit{cancel}}(\mathsf{C}_A) \vee \textbf{\textit{cancel}}(\mathsf{C}_B) \tag{17}$$

$\mathsf{C}_\oplus$ is a new commitment object whose attributes are defined in terms of the attributes of C_A and C_B. $\mathsf{C}_\oplus$ does not provide any information beyond that given in C_A and C_B, but it expresses it in the form of a new commitment.

In $\mathsf{C}_\oplus$, represents that the debtor group is committed to the creditor group to bring about consequent $\mathsf{C}_A.csq \wedge \mathsf{C}_B.ant \wedge \mathsf{C}_B.csq$ when just antecedent $\mathsf{C}_A.ant$ is true.

Debtors are *severally* responsible for $\mathsf{C}_{\oplus}$, so that debtors do not become responsible for more than their input commitments.

Our well-defined condition (equation 9) generalizes the chain rule in [6].

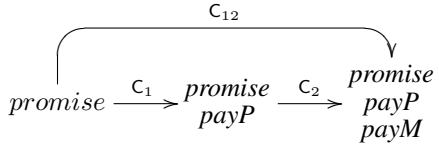
Longer chains of commitments can be composed if each operation is well-defined. We always evaluate $\oplus$ left-to-right.

$$\mathsf{C}_{12\ldots n} = ((\mathsf{C}_1 \oplus \mathsf{C}_2) \oplus \ldots) \oplus \mathsf{C}_n$$

As an example, consider the two commitments in *PayByInt*.

$$\begin{aligned}
\mathsf{C}_1 &= \mathsf{C}_{\text{Payer,Payee}}(\mathit{promised}, \mathit{paidP}) \\
\mathsf{C}_2 &= \mathsf{C}_{\text{MM,Payer}}(\mathit{paidP}, \mathit{paidM}) \\
\mathsf{C}_{12} &= \mathsf{C}_{\{\text{Payer,MM}\},\{\text{Payer,Payee}\}}(\mathit{promised}, \mathit{paidP} \wedge \mathit{paidM})
\end{aligned}$$

which can illustrated as a chain of commitment edges connecting nodes of propositions.



Theorem 2. *Serial composition is not commutative and not associative.*

Usually, serial composition creates stronger commitments. $\mathsf{C}_A \oplus \mathsf{C}_B$ is stronger than C_A alone because, even though both have the same antecedent ($\mathsf{C}_A.ant$), in general, $\mathsf{C}_A \oplus \mathsf{C}_B$ has a stronger consequent ($\mathsf{C}_A.csq \wedge \mathsf{C}_B.ant \wedge \mathsf{C}_B.csq$) with more conjuncts. However, the next theorem shows this is not always the case.

Operator $\oplus$ obeys the following idempotent-like property.

Theorem 3. *Extending a serial composition with a commitment already part of the chain, does not create a stronger commitment.*

$$\mathsf{C}_1 \oplus \cdots \oplus \mathsf{C}_k \oplus \cdots \oplus \mathsf{C}_n \oplus \mathsf{C}_k \tag{18}$$
$$= \mathsf{C}_1 \oplus \cdots \oplus \mathsf{C}_k \oplus \cdots \oplus \mathsf{C}_n \tag{19}$$

Proof. $(\mathsf{C}_1 \oplus \cdots \oplus \mathsf{C}_n) \oplus \mathsf{C}_k$ is well-defined because $\mathsf{C}_k.ant$ is already part of the left-hand side of equation 9. By simple inspection, conditions (10-17) are the same for both sides. Expression 18 is identical to, not stronger than, expression 19.

A commitment can usefully be added to a commitment chain only once; doing so does not create a stronger serial composition. Repeating any part of a loop does not create a stronger serial composition. Given n commitments, the number of distinct serial compositions is bounded above by 2^n.

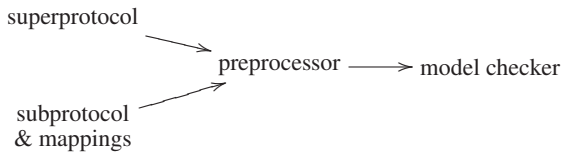


Fig. 2. Processing Steps

5 Processing

Figure 2 shows an overview of our proposed processing. The protocol specifications for the superprotocol and subprotocol are read from files. Note the subprotocol contains one or more mappings between the protocols. These files are read by a preprocessor and are used to generate an MCMAS ISPL model. The ISPL model is input to the MCMAS model checker which checks all the formulae. If all the formula are true, the subprotocol refines the superprotocol.

The preprocessor generates the following checking formulae

$$\mathbf{AG}(sub.guard \rightarrow super.guard) \tag{20}$$

$$\mathbf{AG}(\mathsf{C}_{super}.state = uncond \rightarrow \mathbf{AF}(\mathsf{C}_{super}.state \neq uncond)) \tag{21}$$

$$\mathbf{AG}(\mathsf{C}_A.ant \wedge \mathsf{C}_A.csq \rightarrow \mathsf{C}_B.ant) \tag{22}$$

$$\mathbf{AG}(\mathsf{C}_{super}.ant \rightarrow \mathsf{C}_{sub}.ant) \tag{23}$$

$$\mathbf{AG}(\mathsf{C}_{sub}.csq \rightarrow \mathsf{C}_{super.csq}) \tag{24}$$

Equation 20 ensures the super-run can perform super-basic-actions whenever the sub-run can perform corresponding sub-basic-actions. While MCMAS fairness constraints eliminate runs where unconditionally committed sub-commitments fails to resolve, equation 21 ensures unconditionally committed super-commitments must resolve. Equation 22 parallels equation 9 ensuring serial compositions are well-defined. Equations 23-24 and parallel equations 13-14 ensuring sub-commitments cover super-commitments.

6 PayByInt Refines Pay

6.1 Pay

Space precludes a detailed description of our proposed protocol specification language, so we simply state the protocol specifications with a few notes. Line (1) names and defines the commitment. Lines (2)-(6) describe the Payer role. There are two message actions: *promise* and *pay*, and both are sent by Payer to Payee. The unordered set of basic actions are listed between $\{$ and $\}$. Line (4) defines the guard for *pay* as *promised* and the creation of the commitment. Payee sends no messages in this protocol.

Algorithm 1 *Pay* Protocol

1: $\mathsf{C}_{pay} = \mathsf{C}_{Payer,Payee}(promised, paid)$
2: **role** Payer {
3: $promise = Payer \rightarrow Payee\{promised, \textbf{create}(\mathsf{C}_{pay})\}$
4: $promised \wedge \textbf{create}(\mathsf{C}_{pay}) < pay$
5: $pay = Payer \rightarrow Payee\{paid\}$
6: }
7: **role** Payee {
8: }

Algorithm 2 *PayByInt* Protocol

1: $\mathsf{C}_{payP} = \mathsf{C}_{\text{Payer,Payee}}(promised, paidP)\}$
2: $\mathsf{C}_{payM} = \mathsf{C}_{\text{MM,Payer}}(paidP, paidM)$
3: **role** Payer {
4: $promise = Payer \rightarrow Payee\{promised, \textbf{create}(\mathsf{C}_{payP})\}$
5: $promised \wedge \textbf{create}(\mathsf{C}_{payM}) < payP$
6: $payP = Payer \rightarrow MM\{paidP\}$
7: }
8: **role** MM {
9: $init = \{\textbf{create}(\mathsf{C}_{payM})\}$
10: $payM = MM \rightarrow Payee\{paidM\}$
11: }
12: **role** Payee {
13: }
14: **map** M1: Pay $\mapsto$ PayByInt {
15: $Payer \mapsto \{Payer, MM\}$
16: $Payee \mapsto \{Payee\}$
17: $promised \mapsto promised$
18: $paid \mapsto paidP \wedge paidM$
19: $\mathsf{C}_{pay} \mapsto \mathsf{C}_{payP} \oplus \mathsf{C}_{payM}$
20: }
21: **map** M2: Pay $\mapsto$ PayByInt {
22: $Payer \mapsto \{Payer\}$
23: $Payee \mapsto \{MM, Payee\}$
24: $promised \mapsto promised$
25: $paid \mapsto paidP \wedge paidM$
26: $\mathsf{C}_{pay} \mapsto \mathsf{C}_{payP} \oplus \mathsf{C}_{payM}$
27: }

6.2 PayByInt

Initially (9), Middleman commits to Payer to pass along any payment it receives. This protocol does not address how this commitment came to be created. Payer can not pay Middleman until this commitment has been created (5). Since there is no guard on *payM*, Middleman is free to pay early; but the decision is part of Middleman's agent implementation.

6.3 Refinement Test

We now sketch *PayByInt* refines *Pay*. We test refinement by comparing basic action runs. This following diagram shows superprotocol *Pay* on the top half and subprotocol *PayByInt* on the bottom half. The message action runs are shown on the very top and very bottom for easy reference, but the heart of the diagram is the two basic action runs in the middle. Each position in a basic action run is a set of basic actions.

One of *PayByInt*'s message action runs its basic action run is

Pay msg :		$promise$			pay
Pay $basic$:		$\left\{\begin{array}{c} promised \\ \textbf{create}(\mathsf{C}_{pay}) \end{array}\right\}$			$\{paid\}$
PayByInt $basic$:	$\{\textbf{create}(\mathsf{C}_{payM})\}$	$\left\{\begin{array}{c} promised \\ \textbf{create}(\mathsf{C}_{pay}) \end{array}\right\}$		$\{paidP\}$	$\{paidM\}$
PayByInt msg :	$init$	$promise$		$payP$	$payM$

Erase the message action boundaries and serialize the sets of basic actions to individual basic actions. The basic actions within a set can be serialized in any order that does not violate the basic action guards.

$time$:	0	1	2	3	4
Pay :		$promised$	$\textbf{create}(\mathsf{C}_{pay})$		$paid$
PayByInt :	$\textbf{create}(\mathsf{C}_{payM})$	$promised$	$\textbf{create}(\mathsf{C}_{payP})$	$paidP$	$paidM$

PayByInt refines *Pay* under two mappings. Here, we demonstrate only the mapping M1 in lines 14-20 where Payer and Middleman form a coalition. M1 shows the inter-protocol mappings with *Pay*'s elements are on the left and *PayByInt*'s elements are on the right. Note serial composition of the two commitments in *PayByInt* is required to cover the commitment in *Pay* (line 19).

Lines 15-16 map a single super-role to one or more sub-roles. Line 17 maps a single super-proposition to a single sub-proposition and both occur at time 1 in the run. Line 18 maps the single super-proposition *paid* to a conjunction of sub-propositions. The super-proposition's occurrence must align with the latest sub-proposition's occurrence (time 4).

Finally, we must compute the serial composition in Line 19. From equation 9, we verify $promised \wedge paidP \vdash paidP$ so the composition is well-defined. Equations 10-17

define the composition

$$\mathsf{C}_{\oplus}.debt := \{Payer, MM\} \tag{25}$$
$$\mathsf{C}_{\oplus}.cred := \{Payer, Payee\} \tag{26}$$
$$\textbf{create}(\mathsf{C}_{\oplus}) := \textbf{create}(\mathsf{C}_{payP}) \wedge \textbf{create}(\mathsf{C}_{payM}) \tag{27}$$
$$\mathsf{C}_{\oplus}.ant := promised \tag{28}$$
$$\mathsf{C}_{\oplus}.csq := paidP \wedge paidM \tag{29}$$
$$\textbf{transfer}(\mathsf{C}_{\oplus}) := \textbf{transfer}(\mathsf{C}_{payP}) \vee \textbf{transfer}(\mathsf{C}_{payM}) \tag{30}$$
$$\textbf{release}(\mathsf{C}_{\oplus}) := \textbf{release}(\mathsf{C}_{payP}) \vee \textbf{release}(\mathsf{C}_{payM}) \tag{31}$$
$$\textbf{cancel}(\mathsf{C}_{\oplus}) := \textbf{cancel}(\mathsf{C}_{payP}) \vee \textbf{cancel}(\mathsf{C}_{payM}) \tag{32}$$

Then we verify the sub-commitment $\mathsf{C}_{\oplus}$ covers the super-commitment ($\mathsf{C}_{\oplus} \geq \mathsf{C}_{pay}$) using equations 5-8.

Therefore, the run is a valid sub-run and is a valid super-run, which is consistent with Definition 2. We have only demonstrated a single run here. A full refinement demonstration requires demonstrating all sub-runs. Model checking is required to check all the runs for large protocols.

7 Evaluation

We are writing a proprocessor to generate input for the MCMAS model checker from a protocol descriptions. Model checking protocols should use a fraction of the states supported by current model checking technology, so we should not experience performance or scale problems.

We don't currently support protocols with loops. While loop-free protocols are sufficient for many situations, we hope to remove this limitation in the future.

7.1 Reusable Protocol Library

Protocol design requires substantial effort. Users would like to reuse previously designed protocols rather than having to design their own protocols from scratch. We envision a library of reusable protocols.

Any reasonable definition of refinement should be reflexive and transitive. Transitivity helps to structure such a library. When searching the library, whole subtrees can be eliminated from further consideration by one refinement failure.

8 Discussion

A protocol framework that includes commitments is inherently more flexible than traditional computer protocol frameworks. Where traditional protocol frameworks operate on an all-or-nothing basis with just two stages of completion, agent-based frameworks can operate based on a commitment's four stages of completion. Commitments allow

more flexible enactments because creditors can base their decisions on the promises the debtors which are partial stages of completion.

De Silva [4] propose interaction protocols for open systems based on Petri nets. They enable actions based on past and future preconditions. While their past preconditions are similar to our guards based on propositions, our guards based on conditional or unconditional commitments are more rigorously formalized than their future preconditions. They have no notion of protocol refinement and treat each protocol independently of every other.

Singh [6] states rules similar to those those proposed here for commitment strength. Our equation 9 is slightly stronger than their chain rule, they do not directly state a rule for stronger consequents, and they do not directly state a rule similar to serial composition.

Mallya & Singh [3] propose a definition of protocol refinement (there called subsumption) that compares the order of state pairs. But, we have found if superprotocol states map to overlapping ranges of equivalent subprotocol states, their definition can return false positives. Our procedure does not allow this possibility.

References

1. B. H. Liskov and J. M. Wing. A behavioral notion of subtyping. *ACM Transactions on Programming Languages and Systems*, 16:1811–1841, 1994.
2. A. Lomuscio, H. Qu, and F. Raimondi. Mcmas: A model checker for the verification of multi-agent systems. In A. Bouajjani and O. Maler, editors, *CAV*, volume 5643 of *Lecture Notes in Computer Science*, pages 682–688. Springer, 2009.
3. A. U. Mallya and M. P. Singh. An algebra for commitment protocols. *Journal of Autonomous Agents and Multi-Agent Systems*, 14(2):143–163, Apr. 2007.
4. L. Priyalal, M. Winikoff, and W. Liu. Extending agents by transmitting protocols in open systems. In *In Proceedings of the Challenges in Open Agent Systems Workshop*, 2003.
5. M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7:97–113, 1999.
6. M. P. Singh. Semantical considerations on dialectical and practical commitments. In *Proceedings of the 23rd Conference on Artificial Intelligence (AAAI)*, pages 176–181, Menlo Park, July 2008. AAAI Press.

Counter-proposal: A Multi-Agent Negotiation Protocol for Resolving Resource Contention in Open Control Systems

Jan Corfixen Sørensen and Bo Nørregaard Jørgensen

Maersk Mc-Kinney Moller Institute, University of Southern Denmark,
Campusvej 55, 5230 Odense M, Denmark.
{jcs,bnj}@mmmi.sdu.dk
http://www.mmmi.sdu.dk

Abstract. The resource contention problem in control systems based on software agents occurs when agents with different goals compete with each other, to control a shared resource. In this paper we propose the counter-proposal protocol, a negotiation protocol that resolves the resource contention problem in control systems based on software agents. The protocol is evaluated by an example of a real control problem. The evaluation of the example demonstrates that the protocol is able to find acceptable solutions between competing software agents, if they exist. The focus of this paper is to present the design and implementation of the protocol.

1 Introduction

Contrary to traditional engineered control systems, which typically have a centralized approach to structuring the control logic, control systems based on software agents take a decentralized approach and distribute the control logic among the individual agents [1, 2]. A very important property of this approach is the increased flexibility which software agents add to the reconfigurability of control systems [3].

Hence, control systems based on software agents are constructed by selecting a set of agents, each responsible for different dimensions in the control domain. Later on, if something changes in the control domain, which it inevitably will, the control system can simply be adjusted to the changed operational requirements by changing the configuration of agents. This dynamic nature of systems based on software agents is an important property for creating open control systems[1]. However, the goals of *independently developed* agents in open control systems may conflict when trying to control the same resources in the controlled system. As this situation can cause unwanted behavior to emerge, depending on the agents present in the control system, it must be avoided from happening.

One means to avoid resource contention is to let the agents coordinate their access to shared resources using automated negotiation. In this paper, we propose a protocol

[1] By open we mean that new control agents can be added dynamically while the control system is operational.

for automated negotiation which allows software agents to negotiate the value of shared control parameters. Our approach progresses the state of the art in the area of automated negotiation as it allows agents to participate in a negotiation without knowing anything about the other agents utility function. The key contribution of our protocol is that it allows us to extend a control system's functionality simply by adding new agents without requiring any changes to the logic of the agents already present in the system.

The paper is organized as follows: Section 2 describes state of the art related to automated negotiation protocols. Then, the elements of our negotiation protocol. Section 4 provides an experimental evaluation of our protocol using a real life climate control example. At last, Section 5 discusses how well our approach solved the resource contention issue. Finally, we conclude the paper in Section 6.

2 State of The Art

The research area of automated negotiation constitutes several negotiation mechanisms inspired by a broad range of research fields (Artificial Intelligence, Economics, Social Psychology, and Game Theory) [4–7].

In 1982 Rubinstein looked at the *bargaining problem*[2] in game theory and describes the negotiation process as a game between two agents that can have opposing goals. Each player can in turn give an offer and the opponent can either accept or reject it. The rejecting player has to give a counter offer. The same procedure starts over again alternating between the players. A key assumption in Rubinsteins model is that the agents have perfect information about each other. The main result gives conditions under which the game has a unique sub-game perfect equilibrium [8].

In 1994 and 2001 Kraus contributed with her strategic negotiation model [9, 10] that is based on Rubinsteins alternating offers protocol. The strategic negotiation model allows, in addition to the alternating offers protocol, agents to opt out of the negotiation process. In contrast to the alternating offers protocol, time preferences can influence the outcome of the negotiations.

In 1994 Rosenschein and Zlotkin published their book *Rules of Encounter* where game theory is used in a new way to address the question of how to design the rules of interaction for automated agents. They describe the bilateral *monotonic concession protocol* where both agents start with their best proposals and continue to concede in each negotiation round, until they reach a compromise deal, or both agents can no longer concede. Rosenschein and Zlotkin describe the optimal strategy for the monotonic concession protocol also known as the *Zeuthen strategy* [3]. The key assumption of the monotonic concession protocol is that agents know each others' cost functions for evaluating deals to figure out which agent should concede [11, 3].

The protocols by Rubinstein, Kraus, Rosenschein and Zlotkin are all based on game theory. Several assumptions are associated with game theoretic protocols: The number

[2] The bargaining problem refers to situations when two individuals can have several contractual agreements. Both individuals are interested in reaching an agreement but their interest is not entirely identical. The problem / question is: "What will be the agreement when both parties are assumed to act rationally?"

[3] The Zeuthen strategy ensures that the agent that has most to lose, is the one that concedes

of agents and their identities are assumed to be fixed and known to each other. All agents are assumed to be rational. The space of possible solutions is often assumed to be known by all the agents. The models assume perfect computational rationality; that is, no computation is required to find mutually acceptable solutions within a feasible range of outcomes. Time is assumed not to be an issue when seeking for a solution. Agents are assumed to know each others' utility functions[12].

The computational limitations for finding solutions using the game theoretic approaches led to the establishment of heuristic based protocols [13–15]. The heuristic-based protocols acknowledge that there is a relevant cost associated with computation. The consequence is that the protocols aim at finding good solutions instead of optimal solutions using heuristic rules. Heuristic negotiation models can be based on more realistic assumptions and can therefore be used in broader contexts. Although heuristic protocols solve a subset of the difficulties encountered in game theoretic protocols, they also have disadvantages. The models require an extensive evaluation and often lead to solutions that are sub-optimal.

Common to protocols based on game theory and heuristics, is that they are limited in the feedback that can be achieved with a counter proposal. This limitation makes it difficult to change the issue under negotiation. The argument based protocols was created to address this limitation. The idea behind argument based protocols is to let the negotiating parties exchange arguments about why a proposal is acceptable or unacceptable. Thus, argument based protocols can provide a more directed search of the solution space. Notable examples of argument based protocols includes [16–18]. The problem with argumentation based protocols is that they add considerable overhead to the negotiation process to handle the evaluation of the arguments [5].

State of the art protocols have in common that they do not focus on *independent development* of multiple control agents for an *open control platform*. Agents in such an environment require limited information about each other to support extensibility of the system. To our knowledge, there exists no protocol that support independent development of multiple control agents for an open control platform. In an open control platform, we cannot assume that the number of agents will be fixed and that each agent has extensive knowledge about each other. Agents cannot know about each others utility functions and the entire space of the possible solutions.

What is needed is a multi-agent negotiation protocol that allows combinations of an unknown number of independent developed control agents over time. In our protocol we make no assumptions about the agents utility functions as individual agents in an open control system cannot presume anything about the other agents present in the system. Only the negotiation protocol is known by all agents. However to achieve openness, agents are required to consider the content of the original proposal when composing a counter-proposal. We take a heuristic approach when evaluating the solution space against different control concerns in a global context. Our counter-proposal protocol makes it possible to inspect how different control concerns influence the outcome of the negotiation process and identify the causes of conflicts.

Supporting both *limited information* sharing among agents' and *heuristic evaluation* of the agent's solution space, we contribute with a protocol that supports the independent development of agents in open control systems.

3 Protocol

This section describes the design of our counter-proposal negotiation protocol. Our negotiation protocol offers a structured method of dispute resolution, in which a neutral third agent, the negotiator, attempts to assist the participating agents in finding a resolution to their problem through a negotiation process.

First, the concepts in our proposal protocol are described, then we describe the negotiation process in pseudo-code and the messages that are exchanged between the different agents.

3.1 Concepts

The different concepts in the counter-proposal protocol can be categorized as follows:

Initiator A negotiation process is always initiated by one of the participants and the agent that initiates a negotiation is called the *initiator* agent.

Negotiator The neutral third party, who assists in negotiations and conflict resolution in a negotiation process between the participating agents, is called the *negotiator*. It is the negotiator's responsibility to mediate the initial proposal(s) and possible counter-proposals to the agents involved in the process.

Participant The agents engaged in a negotiation process are called *participants*.

Proposal The negotiation object that represents a range of issues over which agreement must be reached is called the *proposal*. A proposal can contain any control parameters.

3.2 Negotiation Process

The negotiator's responsibility is to manage the negotiation process and to manage all possible proposals. The negotiator does so by keeping track of the history of proposals proposed by the negotiating participants. In our example, the negotiator keeps track of the proposal space using a *tree data structure*.

In the tree data structure each node represents a proposal and an edge represents an interaction between the negotiator and the agent that returned a counter-proposal, see Fig. 1. The negotiator constructs the proposal tree dynamically as the negotiation process progresses. That is, the negotiator tracks the negotiation process by visiting the proposal nodes using level-order traversal.

The pseudo code of the negotiation protocol is described in Algorithm 3. Proposals are respectively removed and added from the first-order traversal queue in Algorithm 3 line 3 and 11.

Two options exist for starting a negotiation process. Considering the first start option, only one agent is allowed to make the initial proposal. The other agents must then either start with a compromise or challenge the initiating agent by making their best offer. The second start option is to put all the initial proposals into the proposal list. The second option ensures that the initial proposal from each agent will be evaluated in the negotiation process. Note, that for both start options only one initiator agent exist. The initiator is selected randomly from the list of participating agents. The start condition

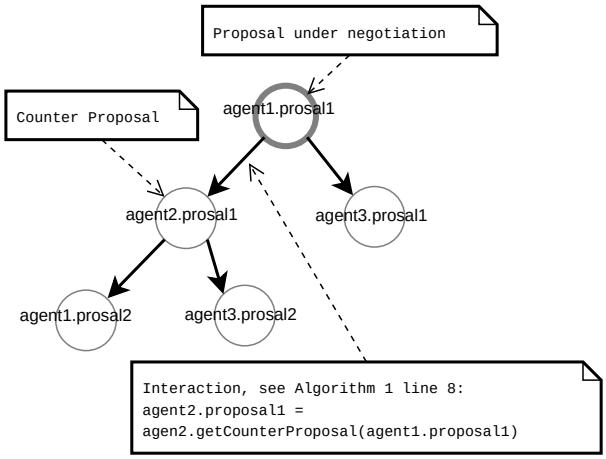


Fig. 1. Each node represents a proposal. The current node represents the proposal undergoing negotiation. An edge represents the interaction between the negotiator and the participant.

of the process is modeled in the tree by adding a root node that represents the initiator's initial proposal, see Algorithm 3 line 1.

While having setup the start condition, the negotiation process can start by letting the negotiator propose the initiator's initial proposal to the negotiation participants. The negotiator adds a new node to the tree each time an interaction between the negotiator and the agent results in a counter-proposal, see Algorithm 3 line 8. When traversing the tree, the current node in Algorithm 3 line 3 represents the proposal under negotiation.

It is important to realize that the counter-proposal protocol covers an iterative process. Finding a solution may take several iterations of proposal negotiations; One iteration for each counter-proposal. The negotiation process continues as long as counter-proposals from the participating agents exist. Section 3.5 explains how infinite negotiation is avoided.

3.3 Messages

In this subsection we describe each of the messages sent between the negotiator and the participants.

Initial Proposal To start the negotiation process, the initiator sends her initial proposal to the negotiator. Upon receipt of the initial proposal, the negotiator request each of the negotiation participants for acceptance of the initial proposal.

Accept When a participating agent has received a proposal from the negotiator, she has two options. The agent can either accept or refuse the proposal, see Algorithm 3 line 6. The logic for determining whether a proposal can be accepted or not, is contained within each participating agent.

Counter Proposal If a participating agent cannot accept a proposal, but can provide a counter-proposal (Algorithm 3 line 8), the negotiation process will be extended

with a new iteration where the counter-proposal will be negotiated, see Algorithm 3 line 11. An agent must take the unacceptable proposal into consideration when generating her counter-proposal.

3.4 Termination

The negotiation process terminates when one of the following conditions has been fulfilled:

Solution Agents have found a solution; that is, all agents have mutually accepted at least one proposal, and there no pending counter-proposals in the negotiation process exist, see Algorithm 3 line 18.

No Solution The negotiation process ends with no solution when all proposals result in unsolvable conflicting counter-proposals. We have an unsolvable negotiation conflict when any of the participating agents continue to provide the same counter-proposals in subsequent iterations. That is, the agents cannot agree on any proposal or counter-proposals. If those unsolvable negotiation conflicts were not handled, the negotiation process would continue forever. Fortunately, unsolvable negotiations conflicts can easily be detected, see Subsect. 3.5 for more details.

Algorithm 3 startNegotiation(initialProposal)

```
Require: Initial proposals exists
Ensure: All proposal negotiated
    proposals.add(initialProposal)
 2: while Not proposals.isEmpty do
        proposal = proposals.removeFirst
 4:     for all participant in negotiation do
            if Not proposer of proposal then
 6:             accept = participant.accept(proposal)
                if Not accept then
 8:                 cp = participant.getCounterProposal(proposal)
                    if cp exists then
10:                     if Not conflict then
                            proposals.add(cp)
12:                     end if
                    end if
14:             end if
            end if
16:     end for
        if accept then
18:         solutions.add(proposal)
        end if
20: end while
```

3.5 Unsolvable Conflict Detection and Resolution

The negotiator detects an unsolvable conflict by monitoring the counter-proposals stored in its proposal history tree. In tree terminology, an unsolvable conflict has occurred if an agent gives the same counter-proposal in a subsequent iteration. This implies the agent cannot provide a counter-proposal that allows a solution to be found, see example Fig. 2. There may be more intermediate counter-proposals before the same counter-proposal reappears. The reappearance of a counter-proposal indicates that the agent's counter-proposal logic has entered a loop due to its boundary conditions. If an unsolvable conflict is detected, the counter-proposal is excluded from the negotiation process and stored in the conflict list.

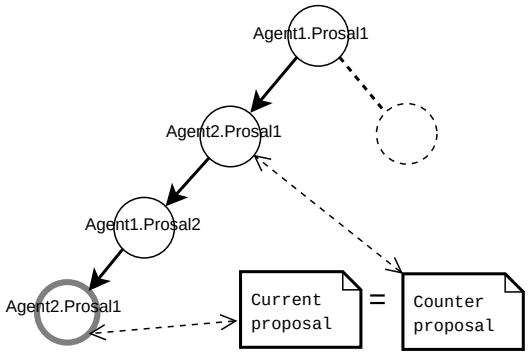


Fig. 2. The example illustrates an unsolvable conflict situation between *Agent 1* and *Agent 2*. *Agent 2* proposes the same proposal *Agent2.Proposal1* more than one time.

Unsolvable conflicts are difficult to resolve automatically, but they can be resolved manually by the user of the system. The user can make a rational decision about a conflict resolution, if she has enough information about the cause of the conflict and the context in which it happened. Our approach allows the user to resolve conflicts by changing the boundary conditions of the agents involved in the conflict.

4 Experimental Validation

In this section, we present an example of a real control problem in greenhouse climate control. The example is based on the knowledge we have obtained by implementing an agent-based dynamic climate control system based on the work of Aaslyng et al. [19].

We have experienced that *artificial light control* is one aspect of climate control that illustrates the resource contention problem very well, and at the same time is relatively easy to explain. We have therefore chosen to base our example on dynamic artificial light control in greenhouses.

To achieve a dynamic light control it is important to support different light control agents that use different strategies for controlling artificial light. Each strategy can

generate several different light-plan proposals. The example is based on the following three light control strategies used respectively by three different control agents. All the strategies control the artificial light at specific hours of the day to reach a specific goal. E.g., a goal could be specified as the photosynthesis sum gained over a day.

Dark Hour The idea of the *dark-hour* strategy is to turn the artificial light off for at least one hour, placed one hour after astronomical sunset and one hour before astronomical sunrise. The incentive behind the dark-hour strategy is that plants need at least one hour of darkness for regeneration.

Electricity Price Electricity prices fluctuate over time because of variations in supply and demand, see Fig. 3. The incentive behind the *electricity-price* strategy is to turn the artificial light on when we get the most photosynthesis to the lowest electricity price. Normally, the cheapest electricity prices will be in the middle of the night where most people are at sleep.

Work Light Artificial light in greenhouses can be used for working light at fixed hours specified for each month in a year. E.g., artificial light can be configured to be turned on at hour 7 and 8 each day in February.

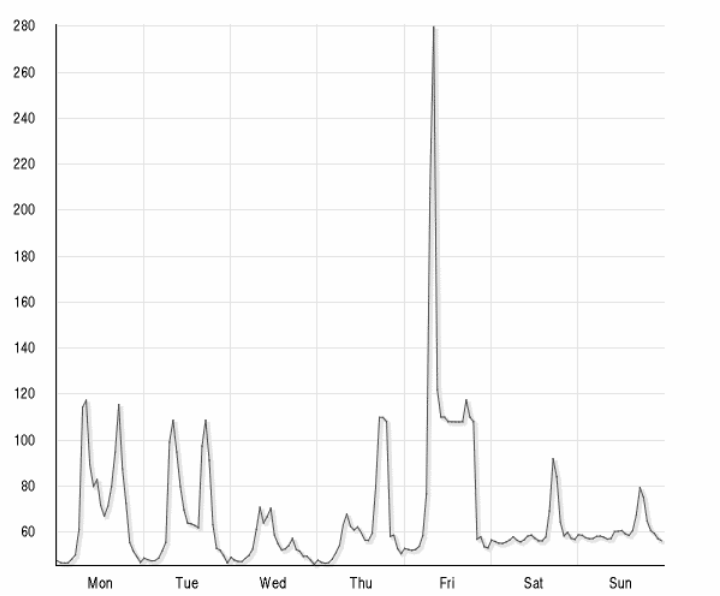


Fig. 3. Fluctuating electricity prices (Euro/MWh) Week 3 in February 2010. See www.nordpool.com

The negotiation object in our example is a 24-hour light-plan represented by an array containing status codes ON(1)/OFF(0)/DontCare(o) for each hour of the day, see Fig. 4. The *hour of day* in the examples refer to specific hours of the 24-hour clock. The

first hour of day is 0, and the last hour of day is 23. E.g., at 10:00 PM the *hour of day* is 22.

Resource conflicts occur when the light control agents simultaneously compete to control a shared light resource. E.g., if one agent wants to turn light off and another agent wants to turn light on, at a specific hour during the night, then a resource conflict has occurred.

In the next subsections, we present three light control examples where our protocol has been applied. First, we present a scenario without conflicts between the negotiating participants. Next, we introduce a scenario with conflicts between the participants resulting in multiple solutions. Finally, we describe a scenario with conflicts between the agents, that cannot be solved without user intervention.

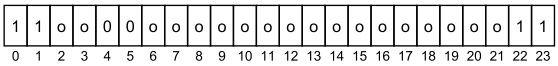


Fig. 4. A 24-hour light-plan that has light turned on at hour 0, 1, 22 and 23. Light is turned off at hour 4 and 5. The status code (o) for all other hours indicates that no decision has been made about controlling light in those hours.

To alleviate the reading of the examples we designate each agent with shorthand names. Agent *Elec* is the agent that uses the electricity price strategy, *Dark* is the agent that uses the dark-hour strategy and *Work* is the agent using the work-hour strategy. Each proposal will be named with the abbreviation *P* following the proposal ID. I.e. *Elec.P0* means proposal with *ID 0* from agent *Elec*. An interaction between two agents is represented by a function call with a proposal argument. I.e. *Dark(Elec.P0)* $\Rightarrow$ *Dark.P1* means that *Dark* receives *Elec's* proposal *Elec.P0* and as a result *Dark* gives a counter-proposal *Dark.P1*.

4.1 Negotiation with one solution

I our first scenario, agent *Elec* has a photosynthesis sum goal of such size, that it can be achieved by turning the light on in the middle of the night. *Elec* will achieve its goal by turning light on at the hours 0 to 5 because of the lowest electricity prices, see Fig. 5. Agent *Dark* finds alternative dark hours seeking clockwise one hour after sunset (6:00 PM), so that two dark hours exist at hour 19 and 20. Agent *Work* turns the light on at hour 7 and 8. Here, no conflicts exist between the agents.

Figure 6 illustrates the proposals mediated by the negotiator between the participants in each iteration of the negotiation process.

Agent *Elec* starts the negotiation by sending its initial proposal *Elec.P0* to the negotiator, suggesting light on at the hours 0 to 5. The negotiator mediates the initial proposal *Elec.P0* to the two other participating agents.

Iteration 1. *Elec's* initial proposal is proposed to *Dark* and *Work*. *Elec's* initial proposal *Elec.P0* does not say anything about the light at the hours *Dark* and *Work*

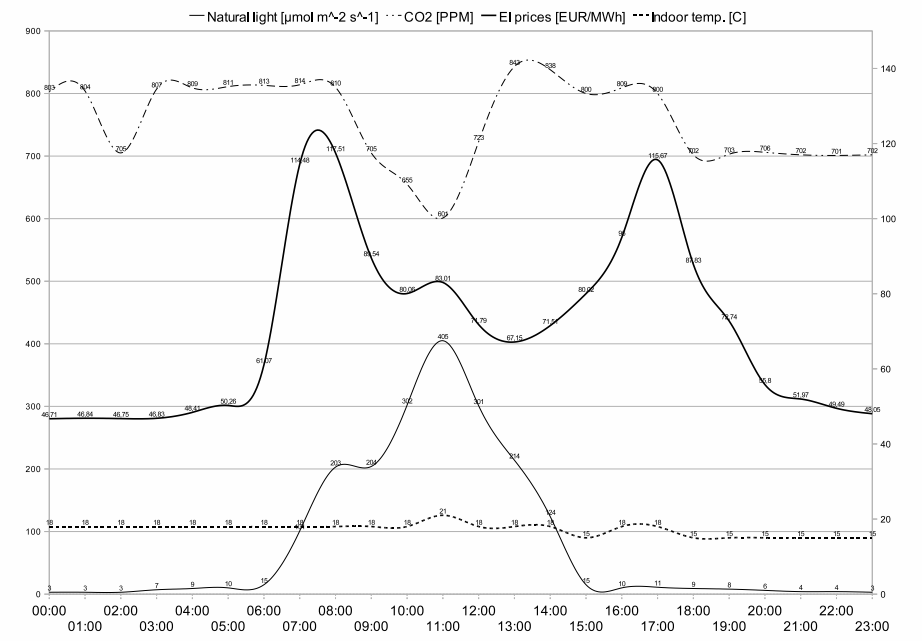


Fig. 5. Electricity prices and climate data for a 24-hour day.

are interested in. Both *Dark* and *Work* can therefore not accept proposal *Elec.P0* but can respectively give counter-proposals *Dark.P1* and *Work.P1* that incorporates *Elec's* initial proposal *Elec.P0*. As a result of the first iteration, counter-proposal *Dark.P1* and *Work.P1* are added to the negotiator's proposal list.

Iteration 2. Due to the level order traversal of the proposal list, the negotiator starts the second iteration by proposing *Dark.P1* to *Elec* and *Work*. *Elec* can accept *Dark.P1* because *Dark.P1* is a counter-proposal that takes *Elec.P0* into account. Proposal *Dark.P1* does not contain any information about the hours *Work* is interested in and *Work* gives a counter-proposal *Work.P2* based on *Dark.P1*.

Iteration 3. Proposal *Work.P1* is proposed to *Elec* and *Dark*. *Elec* accepts *Work.P1* because *Work.P1* is based on *Elec.P0*. *Dark* cannot not accept, but *Dark* can propose *Dark.P2* that accounts for *Work.P1*. By the end of iteration three, the proposal list will contain proposal *Work.P2* and *Dark.P2*.

Iteration 4. Note that proposal *Work.P2* and *Dark.P2* are identical and have both been modified by all the negotiation participants. All the participants have contributed their acceptable proposals without conflicting and as a result *Work.P2* and *Dark.P2* are acceptable to all participants. Iteration four and five ends by respectively adding *Work.P2* and *Dark.P2* to the acceptance list.

Iteration 5. The negotiation process ends because the solution termination criteria is fulfilled, see Subsect. 3.4. No more proposals exist in the proposal list and all agents have mutually accepted at least one proposal.

The example in Fig. 6 illustrates the negotiation of proposals that is not in conflict with each other. The result of the negotiation shows that a conflict free negotiation will result in solutions, all of which are identical. The identical solutions are a consequence of the proposals being modified by all participants, which in turn incorporate their wishes into the counter-proposals. If there is no conflict the solution will be a proposal that incorporates all the participants best wishes. I.e. none of participants have to concede giving their counter-proposals.

Note that we started the negotiation process by letting *Elec* propose its initial proposal first. If we had started the negotiation process by letting all participants give their initial proposals is would result in more iterations but the result would be identical; A merge of the participant's initial proposals.

4.2 Negotiation with Multiple Solutions

The example in this section describes a negotiation scenario where a conflict between *Dark* and *Elec* occurs as a consequence of conflicting strategies. *Elec* has the same strategy from the previous example but *Dark* now has a strategy that requires the dark hours to be the darkest hour of the day, and that alternative dark hours are found counterclockwise from sunrise (6:00 AM) to sunset (6:00 PM).

In order to minimize the number of negotiation iterations, we have chosen to focus only on the two conflicting agents. If we had chosen to include the third agent the negotiation would have resulted in more iterations than would be practical for an example. Fig 7 exemplifies that *Elec's* first proposal *Elec.P0* is to turn light on at hour 0-5 due to the lowest electricity prices from Fig. 5. *Dark's* proposal *Dark.P0* is to turn the light off at hour 1 and 2 because those hours are the darkest according the natural light-level data from Fig. 5.

The negotiation starts by letting *Elec* and *Dark* propose their initial proposals *Elec.P0* and *Dark.P0*.

Iteration 1. In the example the proposal to be negotiated first is *Elec.P0* because it was the proposal that was added first to the proposal list during the start of the process. *Dark* receives *Elec.P0* and realizes that it cannot accept the proposal because there is a conflict; *Elec* wants to turn the light on at the dark hours 1 and 2. *Dark* is a flexible agent and can allow to concede and include *Elec.P0* in its counter-proposal. *Dark* concedes by seeking counterclockwise for alternative dark hours where light can be turned off without conflicting with *Elec.P0*. The result of *Darks* concession is *Dark.P1* where light is proposed to be turned off at hour 22 and 23 and turned on as *Elec* proposed.

Iteration 2. *Dark's* initial proposal is proposed to *Elec* during the second iteration. *Elec* cannot accept because of the conflict, but she can concede and allow *Dark* to turn off light at hour 1 and 2 and instead turn the light on at hour 22 and 23 to achieve her photosynthesis sum goal.

Iteration 3. and 4. At last *Dark* accepts *Elec.P1* and *Elec* accepts *Dark.P1* as consequence of the concessions in iteration one and two. The negotiation process now ends because solutions have now been found and no more proposals exists in the proposal list, see termination criteria in Subsect. 3.4.

```
START
Proposal list: [Elec.P0 111111ooooooooooooooooooo];

ITERATION 1
Dark(Elec.P0 111111ooooooooooooooooo)  =>
     Dark.P1 111111ooooooooooooo00ooo;
Work(Elec.P0 111111ooooooooooooooooo)  =>
     Work.P1 111111o11ooooooooooooooo;

Proposal list: [Dark.P1, Work.P1];
Accept list:   [];
Conflict list: [];

ITERATION 2
Elec(Dark.P1 111111ooooooooooooo00ooo)  =>  Elec.Accept;
Work(Dark.P1 111111ooooooooooooo00ooo)  =>
     Work.P2 111111o11oooooooooo00ooo;

Proposal list: [Work.P1, Work.P2];
Accept list:   [];
Conflict list: [];

ITERATION 3
Elec(Work.P1 111111o11ooooooooooooooo)  =>   Elec.Accept;
Dark(Work.P1 111111o11ooooooooooooooo)  =>
     Dark.P2 111111o11oooooooooo00ooo;

Proposal list: [Work.P2, Dark.P2];
Accept list:   [];
Conflict list: [];

ITERATION 4
Elec(Work.P2 111111o11oooooooooo00ooo)  =>  Elec.Accept;
Dark(Work.P2 111111o11oooooooooo00ooo)  =>  Dark.Accept;

Proposal list: [Dark.P2];
Accept list:   [Work.P2];
Conflict list: [];

ITERATION 5
Elec(Dark.P2 111111o11oooooooooo00ooo)  =>  Elec.Accept;
Work(Dark.P2 111111o11oooooooooo00ooo)  =>  Work.Accept;

Proposal list: [];
Accept list:   [Work.P2, Dark.P2];
Conflict list: [];

END
```

Fig. 6. The result of a conflict free negotiation with one solution.

The example exemplifies how negotiation between conflicting flexible agents will result i multiple solutions due to the fact that both agents were able to concede and take each others proposals into account.

```
START
Proposal list: [Elec.P0 111111ooooooooooooooooo;
                Dark.P0 o00oooooooooooooooooo]

ITERATION 1
Dark(Elec.P0 111111ooooooooooooooooo)  =>
     Dark.P1 111111oooooooooooooooo00;

Proposal list: [Dark.P0, Dark.P1];
Accept list:   [];
Conflict list: [];

ITERATION 2
Elec(Dark.P0 o00oooooooooooooooooo)  =>
     Elec.P1 100111ooooooooooooooo11;

Proposal list: [Dark.P1, Elec.P1];
Accept list:   [];
Conflict list: [];

ITERATION 3
Elec(Dark.P1 111111oooooooooooooo00)  =>  Elec.Accept;

Proposal list: [Elec.P1];
Accept list:   [Dark.P1];
Conflict list: [];

ITERATION 4
Dark(Elec.P1 100111ooooooooooooooo11)  =>  Dark.Accept;

Proposal list: [];
Accept list:   [Dark.P1, Elec.P1];
Conflict list: [];

END
```

Fig. 7. The result of a simple negotiation with multiple solutions.

4.3 Negotiation with No Solutions

During wintertime a scenario may emerge where light must be lit all night to reach *Elec's* photosynthesis sum goal. If *Dark* uses the same strategy as in previous example

and seeks counterclockwise for alternative dark hours, the result will be an unsolvable conflict. The unsolvable conflict will occur because *Elec* will not be able to concede as the photosynthesis sum goal requires light to be lit all night. *Dark* therefore needs to concede in each iteration. After N iterations *Dark* have to propose the same proposal twice in a subsequent iteration. The unsolvable conflict will then be detected as described in Subsect. 3.5. The unsolvable conflict cannot be solved automatically. Instead the cause of the conflict can be presented to the user, suggesting how the conflict can be solved by adjusting the agents goals. E.g., the conflict between *Dark* and *Elec* could be solved by lowering *Elec's* photosynthesis sum goal thereby allowing *Dark* to find its dark hours.

5 Discussion

When the negotiation process has terminated, the negotiator has constructed a tree that contains the history of all negotiated proposals. In case more than one counter-proposal was accepted by all agents, we have to find the best solution in the acceptance list. One approach is to seek for a Pareto optimal solution [20]; that is, a solution where a change from one proposal to another makes at least one agent better off without making any other agent worse off. To find a Pareto optimal solution in the set of accepted counter-proposals, we need to add an utility function to each agent and use the agents' utility functions to calculate a price that each agent has to pay for accepting a counter-proposal. The value of each agent's price depends on the agent's local preference. E.g., *Elec* would calculate its value based on photosynthesis and electricity prices while *Dark* would calculate its value based on the difference in natural light levels from the darkest hour. Note, that we do not allow each participating agent to know each others utility function. Only the negotiator can access the utility functions of each agent. The negotiator now considers an agent to be better off, if she has to pay the smallest price for accepting a counter-proposal compared to any of the others she has accepted. That is the compromise with the lowest cost. The solution is now selected as the counter-proposal where the most agents lose the least. The agent, who made the counter-proposal that was accepted by the other agent, is always better off.

The performance of the counter-proposal protocol depends on the implementation of the agents counter-proposal strategies. The faster an agent can concede in case of a conflict, the faster a solution can be found.

In this paper we have demonstrated three examples which validate the applicability of our protocol, to resolve resource conflicts in dynamic light control in greenhouses. In future work we plan to provide a statistical validation of the protocol to document the stability of the protocol.

6 Conclusions

The counter-proposal protocol is a mechanism that offers structuring of high-level interactions between agents for solving the resource contention problem, that can occur when more agents want to change the same set of values simultaneously. E.g., control parameters, at a given point in time.

In this paper we have focused on explaining our negotiation protocol and the idea behind it. Our case study validates that the approach can solve the resource contention problem between multiple agents when these agents behave rationally.

References

1. Parunak, H., Irish, B., Kindrick, J., Lozo, P.: Fractal actors for distributed manufacturing control. In: Proceedings of the Second IEEE Conference on Artificial Intelligence Applications. (1985) 653–660
2. Shaw, M.J.: Distributed scheduling in computer integrated manufacturing: the use of local area network. International Journal of Production Research (25) (1987) 1285–1303
3. Bussmann, S., Jennings, N.R., Wooldridge, M.: Multiagent Systems for Manufacturing Control. Springer-Verlag (2004)
4. Friedman, D.: The double auction market institution. In: The Double Auction Market: Institutions, Theories and Evidence, Perseus Publishing (1993) 3–25
5. Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Wooldridge, M., Sierra, C.: Automated negotiation: Prospects methods and challenges. Group Decision and Negotiation **10**(2) (2001) 199–215
6. Sandholm, T.W.: Distributed rational decision making (1999)
7. Sycara, K.: Argumentation: Planning other agents' plans. In: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence. (1989)
8. Rubinstein, A.: Perfect equilibrium in a bargaining model. Econometrica **50**(1) (1982) 97–110
9. Kraus, S.: Negotiation and cooperation in multi-agent environments. Artificial Intelligence **94** (1997) 79–97
10. Kraus, S.: Strategic negotiation in multiagent environments. MIT Press, Cambridge, MA, USA (2001)
11. Rosenschein, J.S., Zlotkin, G.: Rules of encounter: designing conventions for automated negotiation among computers. MIT Press, Cambridge, MA, USA (1994)
12. Zeng, D., Sycara, K.: How can an agent learn to negotiate. In: Intelligent Agents III. Agent Theories, Architectures, and Languages, number 1193 in LNAI, Springer Verlag (1997) 233–244
13. Barbuceanu, M., Lo, W.K.: A multi-attribute utility theoretic negotiation architecture for electronic commerce. In: AGENTS '00: Proceedings of the fourth international conference on Autonomous agents, New York, NY, USA, ACM (2000) 239–246
14. Carles, P.F., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. International Journal of Robotics and Autonomous Systems **24** (1998) 3–4
15. Sathi, A., Fox, M.S.: Constraint-directed negotiation of resource reallocations. In: Distributed Artificial Intelligence, Morgan Kaufmann (1989) 163–193
16. Department, L.A., Amgoud, L., Mary, Q., College, W.: Modelling dialogues using argumentation (2000)
17. Parsons, S., Jennings, N.R.: Negotiation through argumentation-a preliminary report (1996)
18. Sierra, C., Faratin, P., Jennings, N.R.: A service-oriented negotiation model between autonomous agents. In: Proceedings of the 8th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, London, UK, Springer-Verlag (1997) 17–35
19. Aaslyng, J., Lund, J., Ehler, N., Rosenqvist, E.: Intelligrow: a greenhouse component-based climate control system. Environmental Modelling & Software **18**(7) (September 2003) 657–666

20. Lai, G., Sycara, K., Li, C.: A pareto optimal model for automated multi-attribute negotiations. In: AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, ACM (2007) 1–3

Verifying Conformance of Commitment Protocols via Symbolic Model Checking

Mohamed El-Menshawy[1], Jamal Bentahar[2], Wei Wan[1], and Rachida Dssouli[2]

[1] Concordia University, Department of Electrical and Computer Engineering Canada
{m_elme,w_wan}@encs.concordia.ca
[2] Concordia University, Concordia Institute for Information Systems Eng., Canada
{bentahar,dssouli}@ciise.concordia.ca

Abstract. Commitment protocols have been widely used to capture flexible and rich interactions among agents in multi-agent systems. Although there are several approaches specifying commitment protocols, none of them synthesize formal specification and automatic verification of these protocols within the same framework. This paper presents a new approach to automatically verify the conformance of commitment protocols having a social semantics with specifications at design time. The contributions of this paper are twofold: first, we present a new language to formally specify the commitment protocols, which is derived from a new logic extending CTL^* with modality of social commitments and actions on these commitments; and second, we develop a symbolic model checking algorithm for the proposed logic, which is used to express the protocol properties we aim to check such as safety and liveness. We also present experimental results of verifying the NetBill protocol as a motivating and specified example in the proposed language using the MCMAS model checker along with NuSMV and CWB-NC as benchmarks.

1 Introduction

Several approaches have been put forward to specify interaction protocols that regulate and coordinate interactions among autonomous and heterogenous agents in multi-agent systems. Recently, some approaches have formalized these protocols in terms of creation and manipulation of commitments [7, 10, 13, 20, 21]. This kind of interaction protocols are called commitment protocols. Other approaches have been proposed to specify interaction protocols using computational logic-based languages [1, 3, 2] or a modified version of finite state machines that enables recombination and reusability of interaction protocols [16].

This paper concerns with defining a declarative specification of commitment protocols using a new language that extends CTL^* introduced in [9] with modality of commitments and actions on these commitments. We adopt the commitment protocols as they are increasingly used in different applications such as business processes [10, 20], artificial institutions [13] and web-based applications [19] during the past years. These protocols have social semantics in terms of the commitments that capture interactions among agents. In fact, commitments support flexible executions and provide declarative

representations of protocols by enabling the interacting agents to reason about their actions [21]. This flexibility is related to the fact of accommodating exceptions that arise at run time by offering more alternatives (or computations) to handle these exceptions [20]. For example, a commitment deadline may be renegotiated among participating agents, or the merchant may prefer to deliver goods before receiving the agreed amount of money. The protocols with fewer alternatives are less flexible, which restrict the autonomy of the participants. Commitments also capture the intrinsic business meanings of the exchanged messages [10, 20] and provide a principled basis for checking compliance of agents with given protocols via capturing the interactions states [7]. As a result, there is a tradeoff between protocol flexibility and the complexity of verifying the compliance. In addition, specifying and designing the commitment protocols that ensure flexible interactions are necessary, but not sufficient to automatically verify the conformance of protocols with some desirable properties that meet the important requirements of multi-agent business processes. This is because the automatic verification of protocol specifications at design time (i.e., before the actual implementation) leads to reduce the cost of development process and increase confidence on the safety, efficiency and robustness.

The aim of this paper is to address the above challenges by formally specifying commitment protocols and verifying them against some given properties using symbolic model checking. In fact, this work is a continuation of our previous publication [12], which is mainly focused on developing a new logical model unifying the full semantics of commitment operations and semantics of social commitments within the same framework. Figure 1 gives an overview of our approach. We begin with develop-

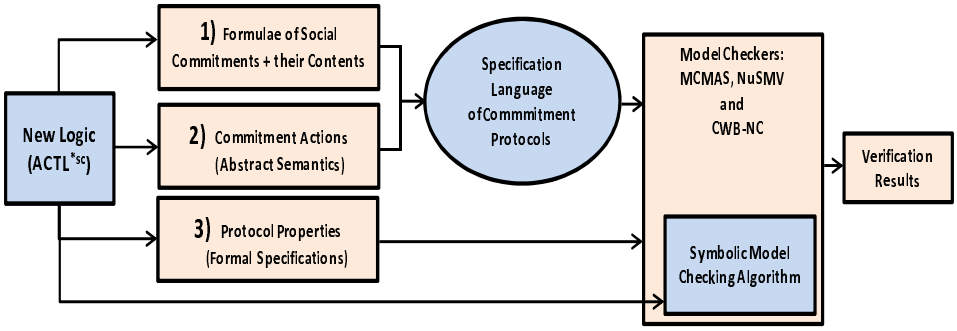


Fig. 1. A schematic view of our approach

ing a new language $\mathcal{L}$ by extending CTL^* with modality of social commitments and actions on these commitments. The resulting logic, called $ACTL^{*sc}$, is used to: (1) express well-formed formulae of commitments and their contents; (2) formally specify an abstract semantics of commitment actions that capture dynamic behavior of interacting agents; and (3) express protocol desirable properties to be checked such as fairness and liveness. By abstract semantics, we mean a semantics that does not define the meaning of all concrete action instances (e.g., withdraw and fulfill actions) but only the meaning of an abstract action denoted in this paper by θ. However, the concrete semantics of

commitments actions is given in our previous work [12]. On the other hand, the protocol properties are used to eliminate unwanted and bad agents' behaviors. Using social commitments and their actions, we define a new specification language of commitment protocols, which we use to regulate and coordinate the interaction among autonomous and heterogenous agents. We then proceed to develop a symbolic model checking algorithm for the proposed $ACTL^{*sc}$ logic based on OBDDs that support the compact representation of our approach. We present experimental results of verifying automatically the soundness of the NetBill protocol, as a running example, taken from e-business domain and specified using our specification language against some given properties. Finally, the implementation of this protocol is done using the MCMAS model checker [17] along with NuSMV [8] and CWB-NC [22] as benchmarks.

The remainder of this paper is organized as follows. Section 2 presents the $ACTL^{*sc}$: syntax and semantics. In Section 3, we use commitments and their actions to define a new specification language of commitment protocols. In Section 4, we encode the proposed logical model based on OBDDs and develop a symbolic model checking algorithm for this model. The implementation of the NetBill protocol and its verification using the MCMAS, NuSMV and CWB-NC model checkers with different experimental results is discussed in Section 5. The paper ends with some discussions of relevant literature in Section 6.

2 $ACTL^{*sc}$ Logic

In this section, we present $ACTL^{*sc}$ logic that we use to specify commitment protocols (see Sect.3) and express the properties to be verified (see Sect.5.2). We enhance CTL^* with social commitments and action formulae applied to these commitments. These modalities are needed for agent interactions and cannot be expressed using CTL^*. Formally, social commitments are related to the state of the world and denoted by $SC(Ag_1, Ag_2, \phi)$ where Ag_1 is the debtor, Ag_2 the creditor and ϕ a well-formed formula representing the commitment content. In some situations, especially in business scenarios, an agent wants to only commit about some facts when a certain condition is satisfied. We use conditional commitments to capture these situations [12]. Formally, conditional commitments are represented by $\tau \rightarrow SC(Ag_1, Ag_2, \phi)$ where "$\rightarrow$" is a logical implication, Ag_1, Ag_2 and ϕ have the above meanings and τ is a well-formed formula representing the commitment condition. We use $SC^c(Ag_1, Ag_2, \tau, \phi)$ as an abbreviation of $\tau \rightarrow SC(Ag_1, Ag_2, \phi)$. In this case, we have $SC(Ag_1, Ag_2, \phi) \triangleq SC^c(Ag_1, Ag_2, true, \phi)$.

The commitments can be manipulated or modified in a principled manner with the interaction progresses using commitment actions. These actions, reproduced from [18], are two and three-party actions. The former ones need only two agents to be performed such as: $Create$, $Withdraw$, $Fulfill$, $Violate$ and $Release$ actions. The latter ones need an intermediate agent to be completed such as: $Delegate$ and $Assign$ actions. Below the syntax and semantics of our language $\mathcal{L}$.

2.1 Syntax of $ACTL^{*sc}$

In the following, we use $\Phi_p = \{p, p_1, p_2, \ldots\}$ for a set of atomic propositions, $\Phi = \{\phi, \tau, \quad, \ldots\}$ for a set of propositional formulae, $\texttt{AGT} = \{Ag, Ag_1, Ag_2, \ldots\}$ for a set

of agent names and $\texttt{ACT} = \{\theta, \theta_1, \theta_2, \ldots\}$ for a set of commitment actions. Agt and Θ are nonterminals corresponding to AGT and ACT respectively. Table 1 gives the formal syntax of the language $\mathcal{L}$ expressed in a BNF-like grammar where "::=" and "|" are meta-symbols of this grammar.

Table 1. The Syntax of $ACTL^{*sc}$-Logic

$\mathcal{S} ::= p \mid \neg\mathcal{S} \mid \mathcal{S} \vee \mathcal{S} \mid \mathcal{S} \wedge \mathcal{S} \mid E\mathcal{P} \mid A\mathcal{P} \mid \mathcal{C}$
$\mathcal{P} ::= \mathcal{S} \mid \mathcal{P} \vee \mathcal{P} \mid \mathcal{P} \wedge \mathcal{P} \mid X\mathcal{P} \mid \mathcal{P}U\mathcal{P} \mid \Theta(Agt, Agt, \mathcal{C}) \mid Create(Agt, Agt, \mathcal{C})$
$\mathcal{C} ::= SC(Agt, Agt, \mathcal{P})$

Formulae in $ACTL^{*sc}$ are classified into state formulae $\mathcal{S}$ and path formulae $\mathcal{P}$. The state formulae are formulae that hold on given states, while path formulae express temporal properties of paths and action formulae. The intuitive meanings of the most constructs of $ACTL^{*sc}$ are straightforward from CTL^* operators. The formula $A\phi$ (resp. $E\phi$) means that ϕ holds along all (some) paths starting at the current state. The formula $SC(Ag_1, Ag_2, \phi)$ means that agent Ag_1 commits towards agent Ag_2 that the path formula ϕ is true. Committing to path formulae is more expressive than committing to state formulae as state formulae are path formulae. The formula $X\phi$ means ϕ holds from the next state, $\phi_1 U \phi_2$ means ϕ_1 holds until ϕ_2 becomes true. The action formula $\theta(Ag_1, Ag_2, \mathcal{C})$ means that an action θ is performed by Ag_1 directed to Ag_2 on the commitment $\mathcal{C}$. For example, if θ is the *Assign* action, Ag_2 will be the agent to which the commitment is assigned. Furthermore, there are some useful abbreviations based on temporal operators: $F\phi \triangleq true\ U\phi$ (sometimes in the future) and $G\phi \triangleq \neg F \neg \phi$ (globally).

2.2 Semantics of $ACTL^{*sc}$

The semantics of this logic is interpreted with respect to the formal model M associated to the commitment protocol using a Kripke-structure as follows: $M = \langle \mathbb{S}, \texttt{ACT}, \texttt{AGT}, R_t, \mathbb{V}, \mathbb{R}_{sc}, \mathbb{L}, s_0 \rangle$ where: $\mathbb{S}$ is a set of states; ACT and AGT are defined above; $R_t \subseteq \mathbb{S} \times \texttt{AGT} \times \texttt{ACT} \times \mathbb{S}$ is a transition relation among states; $\mathbb{V} : \Phi_p \rightarrow 2^{\mathbb{S}}$ is an evaluation function; $\mathbb{R}_{sc} : \mathbb{S} \times \texttt{AGT} \times \texttt{AGT} \rightarrow 2^{\sigma}$, where σ is the set of all paths, is an accessibility modal relation that associates with a state s the set of possible paths along which the social commitments made by the debtor towards the creditor at s hold; $\mathbb{L} : \mathbb{S} \rightarrow 2^{\texttt{AGT} \times \texttt{AGT}}$ associates a given state s with a set of pairs and each pair represents the two interacting agents in s; and $s_0 \in \mathbb{S}$ is the initial state.

Instead of $(s_i, Ag_k, \theta_l, s_{i+1})$, transitions will be written as $s_i \xrightarrow{Ag_k:\theta_i} s_{i+1}$. The paths that path formulae are interpreted over have the form $P_i = s_i \xrightarrow{Ag_k:\theta_l} s_{i+1} \xrightarrow{Ag_{k+1}:\theta_{l+1}} s_{i+2} \ldots$ where $i \geq 0$. The set of all paths starting at s_i is denoted by σ^{s_i} and $\langle s_i, P_i \rangle$ refers to the path P_i starting at s_i. Also, when a state s_j is a part of a path P_j, we write $s_j \in P_j$. Excluding commitment modality and action formulae, the semantics of $ACTL^{*sc}$ state formulae is as usual (semantics of CTL^*) and a path formula satisfies

a state formula if the initial state in the path does so. $M, \langle s_i \rangle \models \phi$ means "the model M satisfies the state formula ϕ at s_i" and $M, \langle s_i, P_i \rangle \models \phi$ means "the model M satisfies the path formula ϕ along the path P_i starting at s_i". A state formula $SC(Ag_1, Ag_2, \phi)$ is satisfied in the model M at s_i iff the the content ϕ is true in every accessible path P_i, to which Ag_1 is committed towards Ag_2, starting from this state using $\mathbb{R}_{sc}$. Formally:

$M, \langle s_i \rangle \models SC(Ag_1, Ag_2, \phi)$ iff $\forall P_i \in \sigma^{s_i} : P_i \in \mathbb{R}_{sc}(s_i, Ag_1, Ag_2)$
$\Rightarrow M, \langle s_i, P_i \rangle \models \phi$ where "$\Rightarrow$" stands for implies.

To make the semantics computationally grounded, which is important for model checking, the accessibility relation $\mathbb{R}_{sc}$, that extends the original Kripke-structure, should be given a concrete (computational) interpretation to be able to describe our model as a computer program. This paper adopts a simple solution saying that if a commitment made by Ag_1 towards Ag_2 is satisfied at state s_i, then there is a path starting at this state (i.e., a possible computation) along which the commitment holds. The intuitive interpretation is as follows: when an agent Ag_1 commits towards another agent Ag_2 about ϕ, this means that there is at least a possible computation starting at this state satisfying ϕ. Formally, we use the following semantic rule:

$M, \langle s_i \rangle \models SC(Ag_1, Ag_2, \phi) \Rightarrow (Ag_1, Ag_2) \in \mathbb{L}(s_i)$ and $M, \langle s_i \rangle \models E\phi$

A path P_i starting at s_i satisfies $Create(Ag_1, Ag_2, SC(Ag_1, Ag_2, \phi))$ in the model M iff Ag_1: $Create$ is the label of the first transition on this path and $SC(Ag_1, Ag_2, \phi)$ holds in the next state s_{i+1}. Formally:

$M, \langle s_i, P_i \rangle \models Create(Ag_1, Ag_2, SC(Ag_1, Ag_2, \phi))$ iff $s_i \xrightarrow{Ag_1:Create} s_{i+1} \in R_t$
and $M, \langle s_{i+1} \rangle \models SC(Ag_1, Ag_2, \phi)$

Because of space limits, we only capture the abstract semantics of action formulae. However, the following concrete instances of these actions ($Fulfill$, $Violate$, $Withdraw$, $Release$, $Assign$, $Delegate$) are used to specify commitment protocols (see Sect.3). As mentioned in the introduction, the concrete semantics of these actions is entirely defined in our previous work [12]. The abstract semantics means that a path P_i starting at s_i satisfies $\theta(Ag_1, Ag_2, SC(Ag_1, Ag_2, \phi))$ in the model M iff $Ag_1 : \theta$ is the label of the first transition on this path and the commitment has been created in the past. Formally:

$M, \langle s_i, P_i \rangle \models \theta(Ag_1, Ag_2, SC(Ag_1, Ag_2, \phi))$ iff $s_i \xrightarrow{Ag_1:\theta} s_{i+1} \in R_t$ and
$\exists j \leq i, M, \langle s_j \rangle \models SC(Ag_1, Ag_2, \phi)$

3 Commitment Protocols

In this section, we present a formal specification language of commitment protocols derived from our logical model M (see Sect.2.2). For the sake of clarity, we use the NetBill protocol to demonstrate this specification.

3.1 Protocol Specification

In this paper, we define the specification of commitment protocols as a set of commitments capturing the business interactions among the interacting agents (or roles) at

design time. In addition to what messages can be exchanged and when, a protocol also specifies the meanings of the messages in terms of their effects on the commitments and each message can be mapped to an action on a commitment. Autonomous agents communicate by exchanging messages and we assume that this exchanging is reliable, which means messages do not get lost and the communication channel ordered preserving.

The protocol specification begins with the commitment COM, which is followed by a message MSG. This message MSG may be directly mapped into commitment actions (captured by Θ) or into inform action. Specifically, MSG could either be withdrawn, fulfilled, violated, released, assigned, delegated or informed message. The delegated (resp. the assigned) message is followed by create message that enables the delegatee (resp. the assignee) to create a new commitment. The inform message is an action performed by the debtor Ag_1 to inform the creditor Ag_2 that a domain proposition holds. It is not a commitment action, but indirectly affects commitments by caus-

Table 2. The formal specification of commitment protocols

Protocol	::= COM ; MSG
COM	::= $SC^c(Ag_1, Ag_2, Prop, Prop) \mid SC(Ag_1, Ag_2, Prop)$
Prop	::= A well-formed formula in our $\mathcal{L}$
MSG	::= $Withdraw(Ag_1, COM) \mid Fulfill(Ag_1, COM)$
	$\mid Violate(Ag_1, COM) \mid Release(Ag_2, COM)$
	$\mid [Assign(Ag_2, Ag_3, COM) \mid Delegate(Ag_1, Ag_3, COM)]$
	; $Create(Ag_1, COM)$; MSG
	$\mid Inform(Ag_1, Ag_2,$ Dom-Pro$)$ | Dom-Pro
Dom-Pro	::= Identify domain propositions

ing transformation from SC^c to SC commitments. The domain proposition Dom-Pro identifies the set of propositions (e.g., $Price\ request$) related to the application domain of the protocol. The inform message allows each agent to resolve its commitment, for example the merchant agent can use it to send a $refund$ to the customer agent. Each domain application can be represented by a suitable ontology. The formal specification of the proposed protocol, that is compatible with the standard protocols in business processes, is defined using a BNF-like grammar with meta-symbols: "::=" and "|" for the choice and ";" for action sequence (see Table 2).

The protocol terminates when the interacting agents do not have commitments to each other. Furthermore, the above specification language of commitment protocols can either be used at run time to reason about the actions [20] or compiled into a finite state machine at design time. At run time, the agents can logically compute their transitions using some reasoning rules. These rules enable agents to choose appropriate actions from the current situation and they are useful for the verification process [7] in relatively small systems. However, these rules are not enough to verify the correctness of the protocols against some given properties when the system is large and complex. For the purposes of this paper, the protocol specification is compiled into a finite state machine at design time where the business meaning of a state is given by the commitments that

hold in this state and the business meaning of actions are given by the actions applied on commitments (see Fig.2). We use symbolic model checking in order to verify the correctness of the protocols specified in our specification language (see Sect.4).

3.2 A Running Example

Let us consider the NetBill protocol taken from e-business domain as a running example to clarify the specification of the commitment protocols. This protocol begins at s_0 with a customer (Cus) requesting a quote for some desired goods like software programs or journal articles. This request is followed by the merchant (Mer) reply with sending the quote as an offer, which means creating a commitment. The Cus agent could either reject this offer, which means releasing this offer and the protocol will end at the failure state s_9 (see Fig.2), or accept this offer, which means creating a commitment at s_3. The Cus's commitment means that he is willing to pay the agreed amount if the Mer agent delivers the requested goods. At this state, the Cus agent still has two possibilities: to withdraw his commitment or to delegate it to a financial company (say Bank: B) to pay the Mer agent on his behalf. The most important thing here, the B agent can delegate this commitment to another bank B_1, which delegates the commitment back to the B agent. The banks B and B_1 delegate the commitment back and forth infinitely and this is presented by a loop at s_{11}. In a sound protocol, this behavior should be avoided (in Sect.5.2, we will show how to verify this issue).

The Mer agent, before delivering the goods to the Cus agent, can withdraw his offer at s_{10} and immaturely moving to the failure state s_9 after refunding payment to the Cus agent. However, when the Cus agent pays for the requested goods and the Mer agent delivers them (within a specified time), then the Mer agent fulfills his commitment at s_5 and then moved to sending the receipt to the Cus agent. Conversely, the Cus agent can pay for the requested goods without being delivered by the Mer agent within a specified time. In this case, the Mer agent violates his commitment at s_8 and

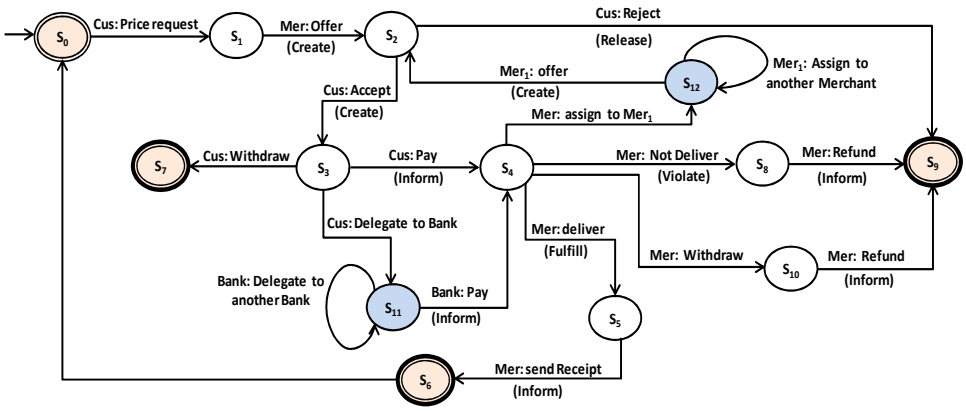


Fig. 2. Representation of NetBill protocol

immaturely moving to the failure state s_9 after refunding the payment to the Cus agent. Finally, the Mer agent, for some reasons, can assign his commitment to another mer-

Table 3. Business meaning of messages

Message	Meaning
Offer$(Mer, Cus, Pay, Deliver)$	Create$(Mer, SC^c(Mer, Cus, Pay, Deliver))$
Accept$(Cus, Mer, Good, Pay)$	Create$(Cus, SC^c(Cus, Mer, Good, Pay))$
Reject$(Cus, Mer, Good, Pay)$	Release$(Cus, SC^c(Mer, Cus, Good, Pay))$
Pay(Cus, Mer, Pay)	Inform(Cus, Mer, Pay)
Deliver$(Mer, Cus, Good)$	Fulfill$(Mer, SC(Mer, Cus, Good))$
Not Deliver$(Mer, Cus, Good)$	Violate$(Mer, SC(Mer, Cus, Good))$

chant (say Mer_1) at s_{12}. Specifically, the Mer agent releases the current commitment with the Cus agent and a new commitment between Cus and Mer_1 is created as a new offer to deliver the requested goods to the Cus agent. As for delegate scenario, the assign action can be repeated infinitely many times among interacting agents and this scenario, presented by a loop at s_{12}, is unwanted behavior in our protocol. Table 3 gives part of the NetBill protocol representation using our specification language along with the business meanings of the exchanged messages that are not expressed directly using commitment actions.

4 Symbolic Model Checking

Here, we describe how to encode the model M and the commitment protocol with Boolean variables and Boolean formulae. This encoding makes our representation more compact and enable us to use symbolic computations. Moreover, the verification algorithms that can operate on this representation are built progressively for symbolic model checking technique.

4.1 Boolean Encoding

In our approach, we use the standard procedure introduced in [9] to encode the concrete model $M = \langle \mathbb{S}, \mathtt{ACT}, \mathtt{AGT}, R_t, \mathbb{V}, \mathbb{L}, s_0 \rangle$ with OBDDs. The number of Boolean variables required to encode states $\mathbb{S}$ is $\mathcal{N} = \lceil \log_2 |\mathbb{S}| \rceil$ where $|\mathbb{S}|$ is the number of states in the model. Let $\overline{v} = \{v_1, \ldots, v_{\mathcal{N}}\}$ be a vector of $\mathcal{N}$ Boolean variables encoding each element $s \in \mathbb{S}$. Each tuple $\overline{v} = \{v_1, \ldots, v_{\mathcal{N}}\}$ is then identified with a Boolean formula, represented by a conjunction of variables or their negation. Thus, the set of states is encoded by taking the disjunction of the Boolean formulae encoding the states. We introduce $\mathcal{N}$ more variables to encode the "destination" state in a transition by means of vector $\overline{v'} = (v'_1, \ldots, v'_{\mathcal{N}})$, $\mathcal{O} = \lceil \log_2 |\mathtt{ACT}| \rceil$ and $\mathcal{A} = \lceil \log_2 |\mathtt{AGT}| \rceil$ variables to encode actions and agents resp. This representation allows us to encode the transition relations in R_t. Let us consider a generic pair $R_{t_1} = (s, Ag, \theta, s')$ be a transition relation in R_t, then its Boolean representation is given by $\overline{v} \wedge \overline{Ag} \wedge \overline{\theta} \wedge \overline{v'}$ in which $\overline{v}$ and $\overline{v'}$ are the Boolean representation of states s and s' respectively, $\overline{Ag}$ is the Boolean encoding for the agent and $\overline{\theta}$ is the Boolean encoding for the action. The Boolean formula

$f_{R_{t_1}}$ corresponding to R_{t_1} is obtained by taking the disjunction of all the possible such pairs. Thus, the Boolean formula f_{R_t} corresponding to the whole transition relation in our model is encoded by taking the conjunction of all the transition relations in R_t, where n is the number of transition relations.

$$f_{R_t}(v_1, \ldots, v_{\mathcal{N}}, Ag_1, \ldots, Ag_{\mathcal{A}}, \theta_1, \ldots, \theta_{\mathcal{O}}, v'_1, \ldots, v'_{\mathcal{N}}) = \bigwedge_{i=1}^{n} f_{R_{t_i}}$$

The evaluation function $\mathbb{V}$ is translated into a Boolean function $f_{\mathbb{V}} : \Phi_p \rightarrow B(v_1, \ldots, v_{\mathcal{N}})$ taking atomic proposition and producing the set $B(v_1, \ldots, v_{\mathcal{N}})$ of Boolean functions when a given atomic proposition is true. For example, given atomic proposition $p \in \Phi_p$, then $f_{\mathbb{V}}(p)$ is a Boolean function encoding the set of states where p is true. In the same way, the function $\mathbb{L}$ is translated into a Boolean function $f_{\mathbb{L}} : \mathbb{S} \rightarrow B(v_1, \ldots, v_{\mathcal{N}})$ taking a state and associating the set $B(v_1, \ldots, v_{\mathcal{N}})$ of Boolean functions representing the two interacting agents having a commitment made at this state. The Boolean encoding process is completed by encoding the initial state $s_0 \in \mathbb{S}$ as a set of Boolean variables like each member in $\mathbb{S}$.

Figure 3 depicts our verification workflow, which is performed in three phases. It starts with the specification of a commitment protocol as an input file written in the Interpreted Systems Programming Language (ISPL) for MCMAS, in the SMV language for NuSMV and in the Calculus of Communicating System (CCS) for CWB-NC. In the middle phase, the protocol properties to be checked are expressed in $ALTL^{sc}$ and $ACTL^{sc}$, which are Linear Temporal Logic (LTL), Computational Tree Logic (CTL) [9] augmented with commitments and their actions. In the last phase, the interpreted protocol specification and properties are the arguments of the model checking algorithm that computes the truth value of each property with respect to this specification.

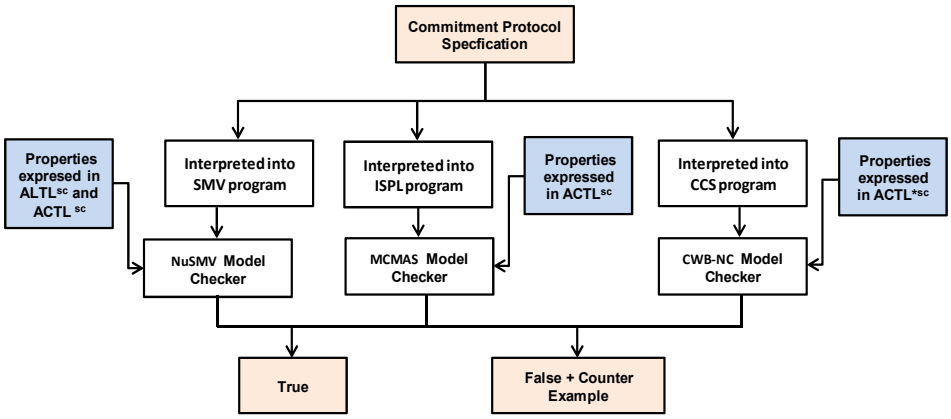


Fig. 3. Verification workflow of the protocol

4.2 Symbolic Model Checking Algorithm

In a nutshell, given the model M representing our protocol and a logical formula ϕ describing a property, the model checking is defined as the problem of computing whether

the model M satisfies ϕ (i.e. $M \models \phi$) or not (i.e. $M \not\models \phi$). Like proposed in [9] for CTL^* logic, in our approach the problem of model checking $ACTL^{*sc}$ formulae can be reduced to the problem of checking $ALTL^{sc}$ and $ACTL^{sc}$ formulae. Figure 4 depicts the expressive powers of the main components of our logic in which $ALTL^{sc}$

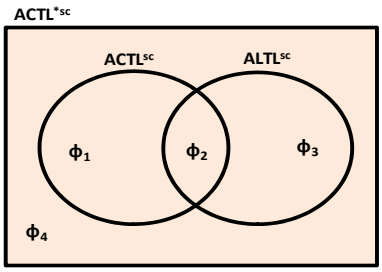


Fig. 4. The expressive powers of $ACTL^{*sc}$, $ACTL^{sc}$ and $ALTL^{sc}$

formulae (e.g., ϕ_3) are $ACTL^{*sc}$ path formulae in which the state sub-formulae are restricted to atomic propositions. Whilst, $ACTL^{sc}$ formulae (e.g., ϕ_1) are $ACTL^{*sc}$ formulae where every occurrence of a path operator is immediately preceded by a path quantifier. The formulae belonging to the intersection (e.g., ϕ_2) can be expressed in $ACTL^{sc}$ and $ALTL^{sc}$. However, the formulae outside the union (e.g., ϕ_4) can only be expressed in $ACTL^{*sc}$, which are usually defined as conjunctions or disjunctions of $ACTL^{sc}$ and $ALTL^{sc}$ formulae.

In our approach, for a given model M and for a given $ACTL^{*sc}$ formula ϕ, the algorithm $SMC(\phi, M)$ (see Table 4) computes the Boolean formula encoding the set of states where ϕ holds, we write this set as $[\![\phi]\!]$. Similarly to the standard OBDD-based model checking for CTL and LTL [9, 15], the Boolean formulae resulting from this algorithm can be manipulated using OBDDs. The OBDD for the set of reachable states in the model M is compared to OBDD corresponding to each formula. If the two are equivalent (i.e., the formula holds in the model), then the algorithm reports a positive output (or true), otherwise a negative output (or false) plus counter example (see Fig.3) is produced.

Table 4. $ACTL^*$ symbolic model checking algorithm

1. $SMC(\phi, M)$ {// for $ACTL^{*sc}$ formulae
2. ϕ is an atomic formula: return $\mathbb{V}(\phi)$;
3. ϕ is $\neg\phi_1$: return $\mathbb{S} \backslash SMC(\phi_1, M)$;
4. ϕ is $\phi_1 \vee \phi_2$: return $SMC(\phi_1, M) \cup SMC(\phi_2, M)$;
5. ϕ is $SC(Ag_1, Ag_2, \phi_1)$: return $SMC_{sc}(Ag_1, Ag_2, \phi_1, M)$;
6. ϕ is $SC^c(Ag_1, Ag_2, \tau_1, \phi_1)$: return $SMC_{scc}(Ag_1, Ag_2, \tau_1, \phi_1, M)$;
7. ϕ is $\theta(Ag_1, Ag_2, \mathcal{C})$: return $SMC_{act}(\theta, Ag_1, Ag_2, \mathcal{C}, M)$;
8. ϕ is $E\phi_1$: return $SMC_E(\phi_1, M)$;
9. }

When the formula ϕ is of the form $SC(Ag_1, Ag_2, \phi_1)$, then the algorithm calls the

Table 5. The procedure for checking $\phi = SC(Ag_1, Ag_1, \phi_1)$

10. $SMC_{sc}(Ag_1, Ag_2, \phi_1, M)$ {//for social commitment modality
11. $X = SMC_E(E\phi_1, M)$;
12. $Y = \{s \in \mathbb{S} \mid (Ag_1, Ag_2) \in \mathbb{L}(s)\}$;
13. return $X \cap Y$;
14. }

procedure $SMC_{sc}(Ag_1, Ag_2, \phi_1, M)$, which begins with computing the set of states where the existential path formula ϕ_1 holds (i.e., $[\![E\phi_1]\!]$) using the standard procedure $SMC_E(E\phi_1, M)$ (see Table 5). Then builds the set of states in which the agent Ag_1 commits towards the agent Ag_2 to bring about ϕ with respect to the function $\mathbb{L}$. The set of states satisfying $SC(Ag_1, Ag_2, \phi_1)$ is finally computed by taking the conjunction of

Table 6. The procedure for checking $\phi = SC^c(Ag_1, Ag_1, \tau_1, \phi_1)$

15. $SMC_{scc}(Ag_1, Ag_2, \tau_1, \phi_1, M)$ {//for a conditional commitment modality
16. $X = SMC(\tau_1, M)$;
17. return $\neg X \cap SMC_{sc}(Ag_1, Ag_2, \phi_1, M)$;
18. }

the two sets. The procedures $SMC_{scc}(Ag_1, Ag_2, \tau_1, \phi_1, M)$ and $SMC_{act}(\theta, Ag_1, Ag_2, \mathcal{C}, M)$ for the formulae of the form $SC^c(Ag_1, Ag_2, \tau_1, \phi_1)$ and $\theta(Ag_1, Ag_2, \mathcal{C})$ are presented in Tables 6 and 7 respectively.

Table 7. The procedure for checking $\phi = \theta(Ag_1, Ag_2, \mathcal{C})$

19. $SMC_{act}(\theta, Ag_1, Ag_2, \mathcal{C}, M)$ {//for action formulae
20. $X = \{s \mid \exists s' \in \mathbb{S}$ and $f_{R_t}(s, Ag, \theta, s')\}$;
21. $Y = SMC_{sc}(Ag_1, Ag_2, \phi_1, M)$;
22. $Z = \{s \mid \exists s' \in X \; and \; \exists P \in \sigma^s : s' \in P\}$;
23. return $Y \cap Z$;
24. }

The procedure for computing the set of states satisfying $ACTL^{*sc}$ formula ϕ of the form $E\phi_1$ is shown in Table 8. This procedure $SMC_E(\phi_1, M)$ first checks if the formula ϕ_1 is $ACTL^{sc}$ formula, then it calls the model checking algorithm $SMC_{actl}(\phi_2, M)$ for $ACTL^{sc}$ to compute the set of states satisfies this formula. Otherwise, it calls the model checking algorithm $SMC_{altl}(\phi', M)$ for $ALTL^{sc}$ after replacing each maximal state sub-formula with a new atomic proposition and the evaluation function is adjusted by adding the set of states that satisfy the new proposition to the existing states (for more details see [9]). That is, if $E_1, \ldots, E_k$ are the maximal state sub-

formulae of ϕ' (i.e., for all $E\ _i$ in ϕ such that $E\ _i$ is not contained in any other maximal state sub-formula of ϕ), $p_1, \ldots, p_k$ are atomic propositions, then the formula ϕ' is obtained by replacing each sub-formula $E\ _i$ with atomic proposition p_i. The resulting formula ϕ' is a pure $ALTL^{sc}$ path formula (to clarify this notion see Example1). Like the standard algorithm of CTL (resp. LTL) formulae [15], the $SMC_{actl}(\phi_2, M)$ (resp. $SMC_{altl}(\phi', M)$) algorithm computes the set of states in which the formula ϕ_2 (resp. ϕ') holds. Moreover, $SMC_{actl,EX}$, $SMC_{actl,EG}$, $SMC_{actl,EU}$, $SMC_{altl,X}$ and $SMC_{altl,U}$ are the standard procedures defined in [15, 9] to compute EX, EG and EU of $ACTL^{sc}$ operators and X and U of $ALTL^{sc}$ operators resp.

Table 8. The procedure for checking $\phi = E\phi_1$

25. $SMC_E(\phi_1, M)$\{ // for existential formula
26. If ϕ_1 is an $ACTL^{sc}$ formula: return $SMC_{actl}(\phi_2, M)$;
27. Otherwise $\phi' = \phi_1[p_1/E\ _1, \ldots, p_1/\ _k]$;
28. for all $E\ _i \in \phi_1$;
29. For i:=1,. . . , k do
30. if $s \in \mathbb{V}(E\ _i)\ then\ \mathbb{V}(E\ _i) := \mathbb{V}(E\ _i) \cup \mathbb{V}(p_i)$;
31. $\Phi_p := \Phi_p \cup p_i$;
32. end for all;
33. return $SMC_{altl}(\phi', M)$;
34. }
35. $SMC_{actl}(\phi_2, M)$ {// for $ACTL^{sc}$ formula
36. ϕ_2 is an atomic formula: return $\mathbb{V}(\phi_2)$;
37. ϕ_2 is $\neg\phi'$: return $\mathbb{S}\backslash SMC_{actl}(\phi', M)$;
38. ϕ_2 is $\phi' \vee \phi''$: return $SMC_{actl}(\phi', M') \cup SMC_{actl}(\phi'', M)$;
39. ϕ_2 is $EX\phi'$: return $SMC_{actl,EX}(\phi', M)$;
40. ϕ_2 is $EG\phi'$: return $SMC_{actl,EG}(\phi', M)$;
41. ϕ_2 is $E[\phi' \cup \phi'']$: return $SMC_{actl,EU}(\phi', \phi'', M)$;
42. }
43. $SMC_{altl}(\phi', M)$ { // for $ALTL^{sc}$ formula
44. ϕ' is an atomic formula: return $\mathbb{V}(\phi')$;
45. ϕ' is $\neg\phi_1$: return $\mathbb{S}\backslash SMC_{altl}(\phi_1, M)$;
46. ϕ' is $\phi_1 \vee \phi_2$: return $SMC_{altl}(\phi_1, M) \cup SMC_{altl}(\phi_2, M)$;
47. ϕ' is $X\phi_1$: return $SMC_{altl,X}(\phi_1, M)$;
48. ϕ' is $\phi_1 \cup \phi_2$: return $SMC_{altl,U}(\phi_1, \phi_2, M)$;
49. }

The time complexity of the proposed algorithm depends on the time complexity of the model checking algorithms for $ACTL^{sc}$ and $ALTL^{sc}$. The complexity of $ACTL^{sc}$ model checking problem (like CTL) is P-complete with respect to the size of an explicit model and PSPACE-complete in case of $ALTL^{sc}$ (like LTL). As a result, the time complexity of $ACTL^{*sc}$ model checking problem in the worst case is PSPACE-complete with respect to symbolic representations.

Example 1. Let ϕ be an $ACTL^{*sc}$ formula, which means that whenever the Cus agent does not pay for the goods, then either the goods will be never delivered or the Cus

agent will eventually withdrawn. Formally:
$\phi = AG(\neg Pay(Cus) \rightarrow A(G\neg Deliver(Mer) \vee FWithdraw(Cus)))$. In order to simplify the model checking, we consider only the existential path quantifier. Thus, ϕ is rewritten as: $\neg EF(\neg Pay(Cuse) \wedge E(F\ Deliver(Mer) \wedge G\ Withdraw))$

- At level 0: all atomic propositions: Pay(Cus), Deliver(Mer) and Withdraw(Cus) are checked.
- At level 1: the formula $\neg Pay(Cus)$ is checked to compute the set of states that satisfy it. The other formula of level 1, $E(F\ Deliver(Mer) \wedge G\ Withdraw(Cus))$, is a pure $ALTL^{sc}$ formula, which is checked by calling SMC_{altl} procedure.
- At level 2: the formula $(\neg Pay(Cus) \wedge E(F\ Deliver(Mer) \wedge G\ Withdraw(Cus)))$ is also checked by calling SMC_{altl} procedure as it is a pure $ALTL^{sc}$ formula.
- At level 3: the formula $E(F\ Deliver(Mer) \wedge G\ Withdraw(Cus))$ is first replaced by the atomic proposition p. The $ALTL^{sc}$ model checking algorithm is then applied to the pure $ALTL^{sc}$ formula $EF(\neg Pay(Cus) \wedge p)$. Finally, all states in the NetBill protocol satisfy the formula: $\neg EF(\neg Pay(Cus) \wedge p)$, which means that this property always holds for this protocol.

5 Implementation

Currently, there are many model checkers developed for different purposes. In this paper, we use MCMAS, a symbolic model checker [17] based on OBDDs to verify the proposed protocol against some properties. More specifically, MCMAS has been implemented in C^{++} and developed for verifying multi-agent systems. It is mainly used to check a variety of properties specified as CTL or $ACTL^{sc}$ formulae in our language. In MCMAS, the multi-agent systems are described by the ISPL language where the system is distinguished into two kinds of agents: environment agent and a set of standard agents. Environment agent is used to describe the boundary conditions and observations shared by standard agents and is modeled as a standard agent. Standard agent can be seen as an non-deterministic automaton with the following components: a set of local states (some of which are initial states), a set of actions, protocol functions and evolution functions that describe how the local states of the agents evolve based on their current local states and other agents' actions.

As benchmarks, we use NuSMV, a symbolic model checker [8] and CWB-NC, a non-symbolic model checker (or automata-based model checker). More specifically, NuSMV has been successfully adopted to model checking multi-agent systems. It is a reimplementation and extension of SMV, the first model checker based on OBDDs. NuSMV is able to process files written in an extension of the SMV language. In this language, the different components and functionalities of the system are described by finite state machines and translated into isolated and separated modules. These modules can be composed synchronously and asynchronously. This paper specifically uses NuSMV to check the properties expressed in $ALTL^{sc}$, which cannot be verified using MCMAS. Meanwhile, it also uses NuSMV to compare the verification results of checking $ACTL^{sc}$ properties obtained by MCMAS (see Sect.5.3).

On the other hand, CWB-NC uses Milner's Calculus of Communicating Systems (CCS) as the design language to model concurrent systems. In fact, CCS is a process

algebra language, which is a prototype specification language for reactive systems. CCS can be used not only to describe implementations of processes, but also the specifications of their expected behaviors. We elect CWB-NC because it uses $GCTL^*$ that generalizes CTL^* with actions and closes to our $ACTL^{*sc}$ formulae without considering social commitments. Thus, we adapt CWB-NC to capture commitment formulae and in this case CWB-NC takes as input the CCS code and $ACTL^{*sc}$ property and automatically checks the validity of this property by building its Alternating Büchi Tableau Automata (ABTA).

5.1 Translating Commitment Protocols

The main step in our verification workflow (see Fig.3) is to translate protocol specification into ISPL, SMV and CCS. In MCMAS, this process begins with translating a set of interacting agents (the Mer and Cus agents in our running example in Sect.3.2) into standard agents in Agent section and commitments into local variables in Vars section. Such variables are of enumeration type including all possible commitment states (see Fig.2), which verify whether the protocol is in a conformant state or not. The actions on commitments are directly expressed in Actions statement where such actions work as constraints to trigger or stop transitions among states. The translation is completed by declaring a set of initial states in InitStates section from which the protocol verification starts to compute the truth value of formulae that are declared in Formulae section.

On the other hand, in SMV, the set of interacting agents are translated into isolated modules, which are instantiated in the main module that also includes the definition of the initial conditions using the INIT statement and the formulae that need to be checked using the SPEC statement. The commitment states are defined in SMV variables in VAR statement. Such states with actions are used as reasoning rules to evolve the state changes. The transition relation between commitment states and their actions is described using TRANS statement where all necessary transitions are defined (see Fig.2). The TRANS statement proceeds with defining the local initial condition using the INIT statement and includes the definition of the evolution function that mainly captures the transition relations using the next statement and Case statement that represents agent's choices. Finally, in CCS, each agent in our protocol is represented by a set of processes and each process is specified by proc statement. The states of a commitment are captured by the set of actions performed on this commitment. Each transition relation is represented by the action labeling this transition followed by another process. For example, let M0,M1,M2 be three processes:

```
proc M0 = 'Request(Cus).M1
proc M1 = Accept(Mer).M2 + Release(Mer).M0
```

means after receiving a request from the Cus agent, the Mer agent accepts or releases. The formulae that we want to check are written in a special file with extension .gctl using $ACTL^{*sc}$.

5.2 Verifying Commitment Protocols

To automatically verify the soundness of the proposed protocol specifications, we need to introduce some desirable properties. In fact, some proposals have been put forward

to classify these properties that satisfy different requirements of the commitment protocols [21, 11]. Specifically, P. Yolum has verified the correctness of the commitment protocols at design time with respect to three kinds of a generic properties: effectiveness, consistency and robustness [21]. N. Desai et al. have classified these properties into two classes: general properties and protocol-specific properties to verify the commitment protocols and their composition [11]. In the following, we specify some temporal properties: fairness, safety, liveness, reachability and deadlock-freedom using $ALTL^{sc}$, $ACTL^{sc}$ and $ACTL^{*sc}$. These properties are more general than the properties introduced in [11] and satisfy the same functionalities of the properties presented in [21]. The differences and similarities of our properties with the properties developed in [11, 21] are explained in Section 6.

1. Fairness constraint: it is needed to rule out unwanted behaviors of agents (e.g. a printer being locked forever by a single agent) [9]. In our protocol, if we define the formula: $AG(AF\ \neg Delegate(Bank))$ as a fairness constraint, then a computation path is fair iff infinitely often the $Bank$ agent does not delegate commitments. This constraint will enable us to avoid situations such as the banks agents delegate the commitment back and forth infinitely (see Sect.3.2). Thus, by considering fairness constraints, the protocol's fairness paths include only the paths that the interacting agents can follow to satisfy their desired states fairly.
2. Safety: means that "something bad never happens". This property is generally expressed by $AG\neg p$ where p characterizes a "bad" situation, which should be avoided. For example, in our protocol a bad situation is: in all paths the Cus agent always pays the agreed payment, but the Mer agent will not eventually deliver the requested goods in all paths starting from the state where the Cus agent has payed: $AG(\neg(Pay(Cus)\ \wedge AF(AG\neg Deliver(Mer)))$.
3. Liveness: means that "something good will eventually happen". For example, in all paths where the Cus agent eventually pays the agreed payment, then there is a path in its future the Mer agent will either (1) deliver the goods and in all paths in the future he will send the receipt; (2) withdraw the commitment; or (3) violate it and in all paths in the future he will refund the payment to the Cus agent: $AF(Pay(Cus) \rightarrow EF((Deliver(Mer) \wedge AFReceipt(Mer)) \vee (Withdraw(Mer) \wedge AFRefund(Mer)) \bar{\vee} (Not\ Deliver\ (Mer) \wedge AFRefund\ (Mer))))$. Another example of liveness is: in all paths where the Mer agent is eventually in "withdraw" state or "not deliver" state, then he will eventually, in all paths starting at these states, refund the payment to the Cus agent: $AF(Withdraw(Mer) \vee Not\ Deliver(Mer) \rightarrow AFRefund(Mer))$.
4. Reachability: a particular situation can be reached from the initial state via some computation sequences. For example, in all paths in the future, there is a possibility for the Mer agent to deliver the request goods to the Cus agent: $AF(EFDeliver(Mer))$. Also, this property could be used to show the absence of deadlock in our protocol. Formally: $\neg AF(EFDeliver(Mer))$, which means that the deadlock is the negation of the reachability property, which is supposed to be false.

The above formulae are only some examples, which were given for the fragment of $ACTL^{sc}$. By considering the definition of LTL given in [9], it is clear that the fairness

and safety properties have equivalent $ALTL^{sc}$ formulae. Furthermore, the first example of liveness property cannot be expressed in $ALTL^{sc}$ because of the existence quantifier and the second example of this property also cannot be expressed in $ALTL^{sc}$ because the candidate $ALTL^{sc}$ formula: $AF(\neg(Withdraw(Mer) \vee Not\ Deliver(Mer)) \wedge FRefund(Mer))$ is weaker than the $ACTL^{sc}$ formula. Note that this does not mean that a meaningful formula in $ACTL^{sc}$ does not have an equivalent formula in $ALTL^{sc}$, for example $AG(AF\neg Delegate(Bank))$ has equivalent $ALTL^{sc}$ formula: $AGF\neg Delegate(Bank)$. In fact, when considering conjunctions or disjunctions of the above formulae, we can construct a formula that is $ACTL^{*sc}$ formula, but neither $ACTL^{sc}$ nor $ALTL^{sc}$.

5.3 Experimental Results

In this section, we present three experimental results using the MCMAS, NuSMV and CWB-NC model checkers. From business process point of view, these experiments try to capture some real-life business scenarios that our specification language of commitment protocols can effectively formalize. In the first experiment we only consider the simple business scenario between two agents (*Cus* and *Mer*) that use the NetBill protocol to coordinate their interactions starting when the Cus agent requests some goods until the Mer agent delivers them. In the second one, we extend the first experiment by adding an agent (say *Mer1*) to which a commitment can be assigned and in the third one, we give the Cus agent the possibility to delegate his commitment to another agent (say *Bank*) to complete the commitment on his behalf.

These experiments show that the current widely known symbolic model checkers are supporting the verification of $ACTL^{sc}$. We use MCMAS as it includes an agent-based specification and is easy to use. However, for $ALTL^{sc}$ formulae that cannot be checked with MCMAS, we use NuSMV. On the other hand, we use CWB-NC in order to verify $ACTL^{*sc}$ formulae as it is the only model checker that can accomplish this kind of verification. Moreover, the use of CWB-NC is motivated by the fact that we need to compare the verification results obtained by the symbolic approach with the automata-based technique to demonstrate the main benefits of our approach.

Results Analysis:
We start our analysis with defining the main criteria that we used to evaluate the performance of model checking theoretically and practically. These criteria are generally related to the model size and total time (i.e., the time of building the model and verification time). Theoretically, we define the size of the model as $|M| = |\mathbb{S}| + |R_t|$, where $|\mathbb{S}|$ is the state space and $|R_t|$ is the relation space. For example, in the third experiment we have: $|\mathbb{S}| = |\mathbb{S}_{Cus}| \times |\mathbb{S}_{Mer}| \times |\mathbb{S}_{Mer_1}| \times |\mathbb{S}_{Bank}| \times |\mathbb{S}_{CP}|$, where $|\mathbb{S}_{Ag_i}|$ is the number of states for $Ag_i \in \{Cus, Mer, Mer_1, Bank\}$ and $|\mathbb{S}_{CP}|$ is the number of states of the commitment protocol. An agent state is described in terms of the possible messages, which he uses to interact with the other and each message is described by a set of states. For example, the two-party actions need 2 states and three-party actions need 3 states. Thus, for the Cus, Mer, Mer_1 and $Bank$ agents in the third experiment, we have 16, 20, 10, 6 states respectively. The protocol is described by the legal actions (see Fig.2), so it needs 13 states. In total, the number of states needed for this

experiment is $|\mathbb{S}| = 249600 \approx 2.5.\ 10^5$. To calculate $|R_t|$, we have to consider the operators of $ACTL^{*sc}$, where the total number is 10 operators. We can then approximate $|R_t|$ by $10.|\mathbb{S}|^2$. So we have $|M| \approx 10.|\mathbb{S}|^2 \approx 6.23.\ 10^{11}$. The model size in our three experiments are shown in Table 9.

Table 9. The model size in the three experiments

	Exp.1	Exp.2	Exp.3
The theoretical model size	$\approx 3.5.\ 10^7$	$\approx 9.73.\ 10^{10}$	$\approx 6.23.\ 10^{11}$

Hereafter, we only analyze the practical results of verifying the above properties of the third experiment because it considers a rich variety of interacting agents. Table 10 displays the full results of these three experiments and the execution times (in seconds) on a laptop running Windows Vista Business on Intel Core 2 Duo CPU T5450 1.66 GHz with 2.00GB memory. The total time for verifying our protocol using MCMAS increases when augmenting the number of agents from 2 to 4 agents. This is normally because the number of OBDD variables needed to encode agents states increases with the number of interacting agents.

As we mentioned, the NuSMV and CWB-NC model checkers are used as benchmarks to evaluate the results obtained by MCMAS. The three experimental results using the same machine as for MCMAS are given in Table 10. In terms of symbolic model checkers, these results reveal that the number of OBDD variables (which reflect the model size) is greater in NuSMV than in MCMAS, but the total time in NuSMV is better than in MCMAS as some optimization techniques are implemented in NuSMV such as on-fly-model checking and cashing. In terms of non-symbolic model checkers, as we expect, when the size model is small, then the total time in CWB-NC is better than in MCMAS and NuSMV (see Exp.1 and Exp.2 in Table 10). However, when the state space increases (as in Exp.3), then the total time in MCMAS and NuSMV is going to be better than in CWB-NC. Note that, we put "–" in Table 10 because CWB-NC does not use OBDD variables.

To conclude this section, the MCMAS model checker underpins different logics such as CTL-logic, supports agents' specifications and performs moderately better than both NuSMV (in terms of OBDD variables) and CWB-NC (in terms of the total time).

Table 10. The statistical results of the MCMAS, NuSMV and CWB-NC

	MCMAS			NuSMV			CWB-NC		
	Exp.1	Exp.2	Exp.3	Exp.1	Exp.2	Exp.3	Exp.1	Exp.2	Exp.3
OBDD Variables	24	39	49	33	53	67	–	–	–
Number of Agents	2	3	4	2	3	4	2	3	4
Total Time(sec)	≈ 0.52	≈ 2	≈ 6	≈ 0.23	≈ 1.11	≈ 1.98	≈ 0.094	≈ 0.564	≈ 8.814

6 Related Work

We review the recent literature with respect to our work. In terms of defining commitment protocols, Chopra et al. [7] have defined the commitment protocol as a transition system and investigated an agent's compliance with the protocol and interoperability with other agents by considering only the two-part actions. Desai et al. [10] have used the Web Ontology Language (WOL) to specify the commitment protocols and describe some concepts of composing multiple protocol specifications to simplify the development of business processes. Fornara et al. [13] have proposed an application independent method to define interaction protocols having social semantics in artificial institutions. They treat commitments like objects as in object-oriented programming and do not consider delegation, assignment, or release of commitment. Yolum and Singh [20] have used commitment operations to show how to build and execute commitment protocols and how to reason about them using event calculus. Their approach only indirectly models the fulfillment of a commitment. Our approach belongs to the same line of research, but it complements the above frameworks by using a more compatible logic with agent open choices, which is an extension of CTL^*-logic. We have also developed symbolic model checking algorithm, which can verify the proposed modality and action formulae without suffering from the state explosion problem that could be occurred in large systems as in [4].

In terms of verifying the conformance of commitment protocols, Venkatraman et al. [19] have presented an approach for testing whether the behavior of an agent in open systems complies with a commitment protocol specified in CTL-logic. The proposed approach complements this work by introducing the model checking technique and the verification of the structural properties geared toward the interactions between agents. Desai et al., [11] has introduced model checking using Promela and Spin to verify commitment-based business protocols and their compositions based on LTL logic. The authors also define general properties in terms of the capabilities of Spin model checker to verify the deadlocks and livelocks, where deadlock can result from the contradiction among composition axioms without considering fairness constraints and reachability properties. They introduced some "protocol-specific properties", which can be defined using the safety property. Moreover, the specification language here is not only $ALTL^{sc}$ specification, but also $ACTL^{sc}$ specification. We also use MCMAS and NuSMV tools, which underpin symbolic representation based on OBDDs that is computationally more efficient than automata-based model checkers such as Spin. Baldoni et al. [3] have addressed the problem of verifying that a given protocol implementation using a logical language conforms to its AUML specification. Alberti et al. [1] have considered the problem of verifying on the fly the compliance of the agents' behaviors to protocols specified using a logic-based framework. These approaches are different from the technique presented in this paper in the sense that they are not based on model checking and do not address the problem of verifying if a protocol satisfies some given properties. Aldewereld et al. [2] have used a theorem proving method to verify the norm compliance of interaction protocols. This norm (e.g., permission) and some temporal properties (e.g. safety and liveness) that need to be checked are expressed in LTL, which is different from our approach that uses symbolic model checking and social commitments.

Recently, Leôn-Soto [16] has presented a model based on the "state-action" space to develop interaction protocols from a global perspective with the flexibility to recombine and reuse them in different scenarios. However, these approaches do not consider the formal specification of the interactions protocols as well as the automatic verification of these protocols. Giordano and her colleagues [14] addressed the problem of specifying and verifying systems of communicating agents in a dynamic linear time temporal logic (DLTL). However, the dynamic aspect of our logic is represented by action formulae and not by strengthening the until operator by indexing it with the regular programs of dynamic logic. In [4], the interacting agent-based systems communicate by combing and reasoning about dialogue games, the verification method is based on the translation of formula into a variant of alternating tree automata called alternating Büchi tableau automata. Unlike this approach, our verification algorithm is encoded using OBDDs. Thereby, it avoids building or exploring the state space corresponding to the model explicitly. As a result, it is more suitable for complex and large systems.

In terms of commitment protocol properties, Yolum in [21] has presented the main generic properties that are required to develop commitment protocols at design time. These properties are categorized into three classes: effectiveness, consistency and robustness. The proposed properties meet these requirements in the sense that the reachability and deadlock-freedom can be used to satisfy the same objective of the effectiveness property. The consistency property is achieved in our protocol by satisfying the safety property. Moreover, the robustness property is satisfied by considering liveness property and fairness paths accompanied with recover states that capture the protocol failures such as if the Mer agent withdraws or violates his commitment, then he must refund the payment to the Cus agent. Hence, our approach can be applied to verify the protocol's properties defined in [21]. As a future work, we plan to expand the formalization of commitment protocol with metacommitments and apply our symbolic model checking to verify the business interactions between agent-based web services.

Acknowledgements

We would like to thank the reviewers for their valuable comments and suggestions. Jamal Bentahar would like to thank Natural Sciences and Engineering Research Council of Canada (NSERC) and Fond Québecois de la recherche sur la société et la culture (FQRSC) for their financial support.

References

1. Alberti, M., Daolio, D., Torroni, P., Gavanelli, M., Lamma, E., Mello, P.: Specification and verification of agent interaction protocols in a logic-based system. In: Proc. of ACM Symposium on Applied Computing, pp. 72–78. ACM Press (2004)
2. Aldewereld, H., Vázquez-Salceda, J., Dignum, F., Meyer, J.-J. Ch.: Verifying norm compliancy of protocols. In: Bossier, O., Padget, J., Dignum, V., Lindeman, G., Matson, E., Ossowski, S., Sichman, J., Vázquez-Salceda, J. (eds.), Coordination, Organisation, Institutions and Norms in Agent Systems, vol.3913 of LNAI, pp. 231–245. Springer (2006)
3. Baldoni, M., Baroglio, C., Martelli, A., Patti, V., Schifanella, C.: Verifying protocol conformance for logic-based communicating agents. In: Lenite, J.A., Torroni, P. (eds.), Computational Logic in Multi-Agent Systems (CLIMA V), vol.3487 of LNAI, pp. 196–212. Springer (2005)

4. Bentahar, J., Meyer, J.-J. Ch., Wan, W.: Model checking communicative agent-based systems. Knowledge Based Systems, 22(3): 142–159 (2009)
5. Bordini, R.H., Fisher, M., Pardavila, C., Wooldridge, M.: Model checking agent-speak. In: Proc. of the 2nd International Joint Conference on AAMAS, pp. 409–416. ACM Press (2003)
6. Chopra, A.K., Singh, M.P.: Multi-agent commitment alignment. In: Proc. of the 8th International Conference on AAMAS, vol.2, pp. 937–944. ACM Press (2009)
7. Chopra, A.K., Singh, M.P.: Producing compliant interactions: Conformance, coverage and interoperability. In: DALT, vol.4324 of LNCS, pp. 1–15. Springer (2006)
8. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An open source tool for symbolic model checking. In: Proc. of the 14th International Conference on CAV, vol.2404 of LNCS, pp. 359–364. Springer (2002)
9. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. The MIT Press, Cambridge (1999)
10. Desai, N., Mallya, A.U., Chopra, A.K., Singh, M.P.: Interaction protocols as design abstractions for business processes. IEEE Transactions on Software Engineering, 31(12): 1015–1027 (2005)
11. Desai, N., Cheng, Z., Chopra, A.K., Singh, M.P.: Toward verification of commitment protocols and their compositions. In: Proc. of the 6th International Joint Conference on AAMAS, pp. 144–146. ACM Press (2007)
12. El-Menshawy, M., Bentahar, J., Dssouli, R.: Verifiable semantic model for agent interactions using social commitments. In: Dastani, M., El Fallah, A.S., Leite, J.A., Torroni, P. (eds.), Languages, Methodologies and Development tools for Multi-agent Systems, LNAI. Springer (2009) (In Press)
13. Fornara, N., Colombetti, M.: Specifying artificial institutions in the event calculus (Chap.14). In: Dignum, V. (editor), Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models, IGI Global, pp. 335–366 (2009)
14. Giordano, L., Martelli, A., Schwind, C.: Verifying communicating agents by model checking in a temporal action logic. In: Logics in Artificial Intelligence (JELIA), vol.3229 of LNAI, pp. 57–69. Springer (2004)
15. Huth, M., Ryan, M.: Logic in computer science: Modelling and reasoning about systems (2nd edition). Cambridge University Press (2004)
16. Leôn-Soto, E.: Modeling interaction protocols as modular and reusable 1st class objects. In: Proc. of Agent-Based Technologies and Applications for Enterprise Interoperability, vol.25 of LNBIP, pp. 174–196 (2009)
17. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A model checker for the verification of multi-agent systems. In: CAV, pp. 682–688 (2009)
18. Singh, M.P.: An ontology for commitments in multi-agent systems: Toward a unification of normative concepts. In: AI and Law, vol.7, pp. 97–113 (1999)
19. Venkatraman, M., Singh, M.P.: Verifying compliance with commitment protocols: Enabling open web-based multi-agent systems. In: Autonomous Agents and Multi-Agent Systems, pp. 217–236 (1999)
20. Yolum, P., Singh, M.P.: Flexible protocol specification and execution: Applying event calculus planning using commitments. In: Proc. of the 1st International Joint Conference on AAMAS, pp. 527–534. ACM Press (2002)
21. Yolum, P.: Design time analysis of multi-agent protocols. Data Knowladge Engneering, 63(1): 137–154. Elsevier (2007)
22. Zhang, D., Cleaveland, R., Stark, E.W.: The integrated CWB-NC/PIOA tool for functional verification and performance analysis of concurrent systems. In: TACAS, vol.2619 of LNCS, pp. 431–436. Springer (2003)

The Logic of Conversation: From Speech Acts to the Logic of Games *

Michel A. Paquette

Collège de Maisonneuve
michel.paquette@cmaisonneuve.qc.ca

Abstract. The classical theory of speech acts has been extended to account for rational goal accomplishment in intelligent dialogues by Vanderveken. The principles of speech act theory yields a natural typology of goal-oriented discourse upon which a general theory of success and satisfaction can be built. Adding some assumptions and additional requirements to this theory, we propose a representation of the speech act account of purposeful communication in game theoretical semantics. We describe the basic components of the resulting logic of conversation. Arguments are proposed to support two specific claims. The first claim is that there is a non-strategic dimension in intelligent dialogue that is best described in terms of the success conditions of a joint coordinated activity. This alleviates some of the difficulties of trying to fit every aspect of a dialogue in the game-theoretical framework. The second claim is that a dialogue is successful in the sense of the illocutionary account of dialogues if and only if (1) there is a solution to the corresponding dialogue game in the form of a Nash equilibrium (2) the participating agents are efficiency maximizers and (3) the participating agents are rational. This equivalence is the pathway from speech acts to the logic of games. In a broader perspective, we want to illustrate how the logical analysis of game theory (the logic of games) offers a fresh perspective on the basic principles of dialogical interaction.

1 Background and Motivation

The present contribution is part of ongoing attempt to bring together a theory of intelligent dialogues that is an extension of speech act theory, namely the one advanced in Vanderveken [25] and some concepts from game theory as represented through the use of epistemic models.[1] This paper is a revised and improved version of an earlier unpublished article [13]. In this contribution, we want to proceed directly to the task that we set for ourselves without trying to bring new support or defense to the background theories of Searle and Vanderveken. We refer the skeptical reader to the works of Searle and Vanderveken [**?**] and Vanderveken [25] for a defense of this approach. In what follows we refer to these theories as (FIL) and (ILDT) respectively.[2] Extending the speech

* Research supported by Gouvernement du Québec, Fonds FQRSC et MRI, programme de Coopération France-Québec.

[1] For an overview of recent work in the study of rational dynamics and epistemic logic in games, see van Benthem [22]. For full details, see van Benthem [23]

[2] We build the acronyms FIL and ILDT from the titles *Foundations of Illocutionnary Logic* and "Illocutiionary Logic and Discourse Typology".

act theory of dialogues towards game theory, or "gamifying" IDLT, raises a number of significant questions. Can the basic components of speech act theory be expressed in a standard game-theoretical framework ? What additional concepts and principles are required to express the notion of success of a dialogue in a discourse type in terms of the concepts of equilibrium and solution for a n-person game? Is the game-theoretical framework rich enough to express the relevant features of rational interactions that are specific to dialogues? Is the logic of games formulated in terms of models of epistemic logic helpful? Does it put matters in a new perspective and raise new questions? We would like to know if this approach can help us combine elements from earlier work such as Lewis [8], Grice [6], Stalnaker [19], Mann [**?**] and Parikh [14]? We offer answers to some of these questions and some hints with respect to others.

We believe that there are some general underlying logical structures to human dialogical interaction. Note that we express this belief in *structures* using the plural form since we want to leave open the possibility that these structures do not collapse in one unified model. The existence of structure is especially clear when that social interaction is purposeful and goal-oriented. People who try to communicate or express information in real circumstances of language use participate in collective actions. All this may seem quite obvious but we cannot ignore the fact that some philosophers and linguists have expressed doubts about the possibility of articulating the sort of systematic theory of discourse that we are seeking. Our work is motivated in part by the will to respond to earlier skeptical views expressed by Wittgenstein, Searle [17] and Mann [10]. The study of communication in a game-theoretical setting is still in a very early stage of growth.[3]

There are some specific conditions that must be satisfied in order for a dialogue to exist and even more conditions must be met for a dialogue to be successful. We will use the notion of *joint coordinated activity* to name the type of collective action that best describe dialogical activity. To interpret these, we make use of Bratman's conditions for joint cooperative activity. Using Bratman's theory of collective intentionality, it becomes possible to formulate the success conditions for the existence of a minimally rational conversation. Using an analogy proposed by Johan van Benthem, language use can be compared to a vast amusement park in which you can play a number of different games. In an amusement park, when you finish some game you get in line to start an entirely new one with different features. This is a telling analogy. A given amusement park is a collection of exciting experiences. So are the strategic games that you can play in conversation. But in order to play strategic games *in* a conversation, you must first satisfy the requirements of cooperative activity, you must accept *to make* conversation and try to make it successful. In a way, you could compare this to the need to buy an admittance ticket to be allowed to play the games in the amusement park. You have to get a day-pass before you can enjoy the rides. It turns out that our understanding of dialogic interaction is irreducibly a two-tiered structure. We will argue that only some joint coordinated speech activity can be gamified. Basically, there must be a will to play the game of conversation with others and to account for this, there is no need to bring in

[3] See van Benthem's postface to Rubinstein [16].

the machinery required for the analysis of strategies and conflicts. The term "analysis of conflict", by the way, is another name for game theory.[4] We claim that there is no need to bring in the machinery of game theory to define under what conditions a *joint coordinated activity* is successful because, *by definition*, there is no strategic agenda *at this level* and the cooperation of the participants is presupposed. Of course, verbal interaction can occur in a wide variety of social contexts and external circumstances can determine many aspects of the structure of a verbal exchange. We do not attempt to account for all the psychological or sociological aspects that might be relevant for the interpretation of a dialogue. We simply assume that there is such a background and that knowledge of this background matters.

Our starting point is a discourse typology that builds on illocutionary logic to build a semantics of dialogues. ILDT provides a recursive definition of the set of possible conversations and a framework that allows for an exact definition of the notion of "successful dialogue". We take ILDT as our starting point because it is general, explicit and semi-formal.[5] A theory of success and satisfaction for discourses should include the conceptual apparatus necessary to describe *strategic interactions* in conversations. For this purpose, it is natural to use some concepts from game theory. For those of our readers who are unfamiliar with game theory we can only offer a brief description. Game theory is best viewed as "a bunch of analytical tools", to use the apt words of Osborne and Rubinstein [12]. This manner of introducing game theory avoids the pitfall of thinking about game theory as a unified and universal set of fixed principles that reveal the deep structure of human interactions. We do not know that such a universal theory might exist. It is perhaps best to think of game theory as a social science. The relevance of game-theoretical concepts for the study of human dialogue should be clear. Here are two basic assumptions of game theory: (1) when they play games, agents are decision-makers who pursue well-defined exogenous objectives and in so doing they are *rational* (2) players have some knowledge and some beliefs that allow them to form expectations about other players behavior. We say that players reason *strategically*. Since our goal is to bridge the gap between a certain theory of dialogues, namely ILDT, and certain tools from the game-theoretical toolbox, it should be clear that what we will be doing here is applied game theory and that are we are making use of a small fraction of what is already known. The study of cooperation in rational interaction is a topic about which there remain many open questions. We only thread our way among the most general ones.

[4] in reference to the subtitle of Meyerson's well-known textbook[11].

[5] Here, we do not discuss the relation of our own efforts with the works of Aher and Lascarides [1] and Parikh [14]. Asher and Lascarides's Segmented Discourse Representation Theory (SDRT) is based on Kamp and Reyle's discourse semantics and Parikh's theory makes use of situation theory. The links between these approaches and illocutionary logic are not all clear which makes intertheory comparison difficult and orthogonal to the orientation of the present paper.

2 Searle and Vanderveken's Illocutionary Account

The logical basis of our account is the formal speech act theory of Searle and Vanderveken found in [**?**]. According to FIL, "the minimal units of human communication are speech acts" and in general "an illocutionary act consists of an illocutionary force F and a propositional content P". These statements are well known and often quoted but they are not always duly understood. We must stress the point that these claims do not imply that illocutionary logic is "sentential" and could not be concerned by the relations between utterances.[6] Firstly, just as there are logical relations between propositions, there are many logical relations that speech acts have among themselves in virtue of their form and content. Illocutionary logic offers a formal and systematic account of those. Moreover, in the use of language in particular circumstances, there are logical relations between the various illocutionary acts performed by different agents who try to express or communicate various propositional contents by the coordinated use of language in those circumstances. Secondly, the purpose of FIL is to provide an analysis of the logical forms of actions performed by speakers. Illocutionary acts have propositional contents without being themselves reducible to sentences and propositions. Speech act theory is coherent with its basic tenet by maintaining that the proper semantic value of illocutionary acts is not *truth* but *success*. In fact there are three distinct and irreducible semantic values in the formal semantics of illocutionary logic : *truth*, *success* and *satisfaction*. Of course, we are aware that some respectable logicians would disagree with this ramification of semantic values.[7] The resulting formal semantics for speech act theory has been investigated thoroughly by Vanderveken in [24].

Vanderveken establishes the groundwork for the theory of discourse ILDT by putting forward a complete account of the types of possible discursive goals that speakers can attempt to achieve by way of conversing. His framework provides resources by which the conversation types in terms of previously defined concepts of illocutionary logic: the mode of achievement of a discursive goal, the thematic conditions, the background conditions and the sincerity conditions. These aspects of dialogue are in line with similar components from the analysis of speech acts in FIL. The typology of possible discourses is also grounded on familiar notions. The basic uses of language are few in number, in fact they can be separated according to four possible directions of fit: downward or word-to-world, upward or world-to-word, the double direction of fit such appropriate for declarations and the empty declaration of fit characteristic of expressive discourses.

3 The Analysis of Intelligent Dialogues in ILDT

We start with the basic units. We want a concept of intervention in a dialogue that will coincide with the idea of a move in a game. For that reason it seems best to construe

[6] The contrary seems implied by the authors in Asher and Lascarides[1] p.74.

[7] This account of the semantic value of performative sentences is in line with views shared by Frege, Austin and Searle, In contradistinction see, Lewis [8] "example 7". Some arbitration can be found in Belnap [4].

interventions as sets of illocutionary acts. By doing so, we do not rule out the possibility of an intervention consisting of a single illocutionary act. Nor do we suppose that an intervention is made up of a number of locutionary events that are contiguous in time. Interventions can be embedded in nested structures, they can be complex in the sense of being structured units having structured parts. Some interventions are speech acts of a higher level, such as the opening statement in a judicial trial. For example, the opening of a court session is an intervention that contains another intervention, the preliminary announcement. The preliminary announcement often proceeds according to a fixed ritual consisting of a predictable series of directives and declaratives:

– *Silence.*
– *All rise.*

(After the judge enters the room and is seated.)

– *The Court is now in session.*
– *Please be seated.*

In view of its fixed protocolar structure, a judicial trial is not a very typical form of discourse, but it offers clear examples of nested interventions. Viewing discourse as made up of interventions gives us the precision we need to provide a game-theoretical analysis of conversations because it permits us to construct interventions that correspond to strategic moves. It also gives us the freedom we want to sieve out irrelevant *noise* with respect to discursive goals. We can thus speak of interventions such as describing an object, answering a question, concluding an argument or summing up a debate. Since the analysis of dialogues in ILDT is centered on the achievement of discursive goals, interventions are defined as speech acts or collection of speech acts that bring the participants closer to the achievement of these discursive goals.[8] By linking the speech act concept of intervention to the game-theoretical concept of a move in a game, we are taking the first step towards a formulation of ILDT *in* game theory. To construct a reasonable account of dialogue games, we must take notice of the fact that a move in a dialogue game may require the performance of more than one illocutionary act and that some illocutionary acts that occur in a conversation may not count as a move with respect to a particular discursive goal. Moreover, we make the assumption that each intervention corresponds to a deliberate attempt to make a discursive action. This attempt results from a choice made by a participant at some point in the history of the dialogue. From the standpoint of each participant in a dialogue, the problem of choosing interventions can be viewed as a problem of individual choice under uncertainty. So we can proceed to define the concept of rational behavior for this type of choice in the context of social interaction. Indeed, we state that a formal dialogue game can be defined as a (partial) description of a sequence of interdependent Bayesian decision problems.[9] We take these basic notions as the first components of our dialogue game theory.

[8] More precisely, these collections are families of equivalent sequences of speech acts.

[9] The Bayesian form of a game is a generalization of the extensive form, the tree representation. The extensive form should not be considered to be essentially different from the strategic, tableau form. See Myerson [11] p. 37, Stalnaker [21] p. 5 and Stalnaker [19].

To proceed further, we need to explain the concept of strategy for a participant in a conversation. We want to understand how conversations can be viewed as games where plans and strategies mesh, compete or collide. We have to be mindful of the applicability of these concepts in the case of dialogues. In game theory, a *plan* is made up of choices and a *strategy* is defined as a *complete plan*. The concept of *strategy*, under the standard interpretation of game theory, seems to impose too much structure on dialogues. This difficulty has been noted by the economist A. Rubinstein in his book on economics and language, Rubinstein [16]. We must consider seriously the possibility of being misguided by the rhetoric of game theory. A strategy specifies what choices will be made in every possible situation that might turn up in the history of the conversational exchange, according to the pattern of information that the logical structure of the game indicates expressly for some agent in a particular case. This concept of strategy is the one needed in order to define the concept of *solution* for a game, a solution being a systematic description of the possible outcomes that may emerge in a family of games. A solution spells out the possible outcomes of a game type in the form a method, which assigns to each game a set of *profiles* of strategies satisfying certain conditions of stability and rationality.[10] Plans and strategies, if they are interpreted cautiously, can be essential elements of a game-theoretic account of the type strategic interactions that take place *inside* dialogue games. All this looks promising. Nevertheless, we must worry that the concept of a *game of strategy*, as introduced by Von Neumann and Morgenstern and expanded by Kuhn and others, carries with it a certain number of assumptions that bring in too much structure to fit the simple and abstract idea of a dialogue with an internal discursive goal.[11] We must ask how the idea of "winning" can be applied in the analysis of different types of dialogues and we must also worry about what counts as a "payoff function" in the general case. Upon reflection, some conceivable dialogues may contain no strategic interaction among the participants and there may not be a winner in any clear sense. This is the problem that we address by taking as our starting point the obvious fact that all dialogues must contain some basic cooperation among the participants. To make use of this obvious fact, we introduce the notion of joint coordinated activity as the most general feature of intelligent dialogue.

4 Interlocution as joint activity and collective intentionality

We want to consider more closely a leading idea ILDT that "speakers contribute to conversations with the collective intention of achieving a discursive goal".[12] We face a problem when we try to formulate this key principle in game-theoretical parlance. The basic notion is that of a type of interaction where some discursive goal is aimed at and where participants contribute in an attempt to achieve a common goal. It would be a downright misnomer to call this a "cooperative game" even if the term sounds intuitively right. We propose to use the descriptive phrase *the dialogue as a joint coordinated activity* to name this interaction. We recommend refraining from calling such

[10] Rubinstein [16] p. 86.
[11] See Von Neumann and Morgenstern [26] p. 49, and Kuhn [7].
[12] Vanderveken [25].

activity "a game", and also recommend that we try to steer clear from the terminological confusion of calling it a "cooperative game". Game theory has a copyright claim on this vocabulary and provides exact definitions. True enough, following an analysis first proposed by John Nash, it is possible to model cooperative behavior as the result of noncooperative bargaining between cooperating players. In a sense, it becomes a special case of noncooperative interaction.[13] This reduction may be formally correct but we refrain from using it to model intelligent dialogues.[14] It is true that preliminary negotiations are possible and can be very elaborate in form and content. But, it seems wrong to suppose that some form of bargaining must always take place before a dialogue can exist. We concur with W. C. Mann that strategic interaction doest not always fit actual dialogues.[15]

One of the most general feature of conversational interlocution is that by willingly taking part in a dialogue, the participants make it manifest that they accept to play a part in a joint coordinated activity with the collective intention to achieve some internal discursive goal or some set of discursive goals. In so doing, the participants in a dialogue must commit themselves to the joint activity with the appropriate collective intention. In recent years, philosophers such as J. Searle, R. Tuomela and M. Bratman [5] have proposed interesting analysis of collective intentions and joint activities. Bratman offers a review of previous attempts to define collective action and offers an improved analysis. He carefully delineates a set of essential attitudes that explain what is required in a joint coordinated activity aimed at a common goal. According to his explanation, it is necessary but not sufficient that (1) the participants both have the intention of achieving the common goal. It is also required (2) that each participant intend to achieve the goal in a way that implies the other participants action, in accordance with subplans that mesh. The subplans need not be identical, but (3) they must be compatible and (4) jointly permit the achievement of the goal. Moreover, (5) it must be common knowledge that both participants intend to achieve the common goal. These conditions are sufficient for the collective intention of achieving a discursive goal. From a philosophical point of view, pending the formulation of new arguments to the contrary, we may consider that they express necessary conditions. It can be argued that a well worked-out theory of human dialogues needs a well worked-out account of collective intentional actions. For our purpose, this analysis of joint coordinated activities has three very desirable features. First, it does not rest on an essential use of "we-intentions" that would exist over and above individual intentions. Secondly, it explains joint action as more that the sum of individual action, as can be seen in (2). Thirdly, this analysis of joint coordinated activity can be incorporated in a formal logic of action in a straightforward manner. For these reasons we adopt it here as an explanation of cooperation in the logic of dialogue. The four conditions are taken as necessary conditions for the existence of a coordinated dialogue.

---

[13] See Myerson [11], p. 370.

[14] See Grice [6] p. 29 for a criticism of such "quasi-contractual basis" in cooperative conversation.

[15] Mann has coined the term "Single Comprehensive Model Fallacy" for the view that any idea about how something works must cover all cases. See Mann[10].

5 Common knowledge and strategic knowledge

Common knowledge and collective intentions are pervasive in dialogues. They help us explain what it means to share a common discursive goal. The shared knowledge of participants is sometimes closely related to the explanation of rational behavior in social interaction. Part of the common knowledge of participants belongs to linguistic competence; someone who understands the word "negotiation" can form some expectations about what is supposed to happen if he is going to take part in a negotiation. The basic knowledge that is required for achieving a successful dialogue — if it is at all possible for participants to achieve a given goal in given circumstances — must include some common or shared knowledge. In game theory, it is usual to suppose that the players know the rules of the game they play, that they can make inferences to anticipate the other player's next moves. In connection with the concept of strategy for a game of a given type, it is often necessary to assume that each player has a detailed "mental model" of the possible unfoldings of the game, a representation that is at least as complete as required by the analysis of the solution concept for that type of game. On the other hand, we would like to acknowledge that when game playing is goal-driven, a large number of the "possible unfoldings" are not necessarily known to the player. For these reasons, dialogical strategic interactions are best viewed as extensive games with imperfect information.

As we have stated at the beginning, our analysis of dialogic interaction is irreducibly a two-tiered. We find echos to our concern in the work of Rubinstein. He notes that a conversation "differs from [a] debate in that participants pursue common rather than opposed interests".[16] For that reason, Grice's *cooperative* principles seem more appropriate for conversation in general than for debates in particular. Our distinction between dialogue as a joint coordinated activity and dialogue as a strategic game expands this difference. There are indeed many forms of dialogues where the speaker and hearer pursue "different" if not "opposed" interests. So to speak, they use different preference relations over outcomes. Clearly, dialogue game theory must account for this basic fact. Speakers and hearers *do* need to agree to participate in a joint coordinated activity. Nevertheless, they need not agree about content, subgoals and means that characterize the games that are taking place within a conversation. As we have indicated, we believe that the conceptual apparatus of game theory is well suited for the task of describing the strategic interactions that take place in those possible dialogue where participants have opposed or at least different interests. This suggests that we need a two-tiered account to describe the logical structure of dialogues: on one level, we have the dialogue as a joint coordinated activity with its own success conditions and on another level we have the strategic games that take place inside the joint coordinated activity, with their own success conditions. The other option would be to define the basic concept of a game so that the elementary case would coincide with the idea of a joint coordinated activity. This type of unification does not blend nicely in the conceptual framework of standard game theory.

If the basic notion of joint coordinated activity is going to do some work for us, it should be clear that it must account for the fact that each participant is going to form

[16] Rubinstein [16] p. 112.

some expectations about the typical scenario of a dialogue. For example, if I am going to take part in a debate, just because I know that the interaction is going to take the form of a debate, I expect that two parties who disagree over some issue will oppose arguments.[17] Quite clearly, this is not the only type of knowledge that is required from the participants in a dialogue. A more general form of knowledge is also involved. This more general form of knowledge provides expectations about the strategic interaction that should take place in the typical scenario for a dialogue of a given type. Thus, in our example, we expect debaters to put forward their best arguments — rather than the less persuasive ones — and we expect them to refrain from indicating loopholes and shortcomings that they may be aware of in their positions. Of course, we want to refrain from saying that all the knowledge required to contribute to a successful dialogue is either part of semantic competence or part of strategic knowledge. In order to interpret the propositional content of speech acts, the partners in a dialogue must also possess an unspecified and *unspecifiable* amount of knowledge about the world in general and about the immediate surroundings. This knowledge about the world helps the hearer in his attempt to understand the acts of reference and acts of predications that are made by the speaker. Some may be inclined to see an insuperable difficulty in the problem of spelling out exactly what this knowledge amounts to. May they be reminded that in order to have a successful dialogue with someone, you do not need to know much about what he knows or believes about our external world. So a reasonable logic of dialogues can be constructed without venturing in vague cognitive and doxastic assessments.

There is little doubt that our general strategic knowledge derives from our accumulated experience. Indeed, we acquire strategic skills as we take part repeatedly in similar dialogue games and we can learn from the experience of our ancestors that is passed on by culture and education. We can also take a relevant clue from evolutionary game theory, a type of game-theoretic analysis that seeks to explain the structure of social institutions in terms of repeated games. To become a good negotiator, one has to learn the finer points of a type of strategic interaction. This knowledge goes well beyond the mere understanding of the word "negotiation" and well beyond the ability to understand the various illocutionary acts that are performed as the actual dialogue unfolds. The general knowledge of a good negotiator evolved through experience with negotiations. There is, at least in principle, a clear distinction between the basic common knowledge that must be shared by participants who engage in a joint coordinated activity and the more general knowledge, the *strategic knowledge*, that is acquired by taking part repeatedly in dialogues with similar logical structure and formally identical discursive goals. Beyond that, as Wittgenstein pointed out, our language games are embedded in forms of life in which we have all sorts of other goals, that we cannot attain by discursive means. Such is, for instance, the goal of repairing a car. In order to achieve such goals, we need knowledge that does not relate to language or strategy. We need knowledge about how things work, such as car mechanics.

There are some paradigm cases of strategic interactions in dialogues with the descriptive goal (debates, interviews, interrogations) and in dialogues with the deliberative goal (negotiations, bargaining sessions). It would be difficult to imagine strategic interaction taking place in those declaratory and expressive discourses where the speakers

[17] See Rubinstein [16] p. 42 for a game-theoretic account of debates.

put forward the same propositional content such as when a group of people takes an oath or pledges allegiance or when people jointly express the same mental states as in a collective prayer. In order to act according to a strategy, one must be able to choose a course of actions, to select the general principles governing his choices freely and act in accordance to some personal agenda. It is a postulate of the logic of discourse that each participant is agentive with respect to his interventions. The set of interventions that lead to the accomplishment of a discursive goal is governed by a basic logic of action. To gain a better understanding of these issues, we need to be more specific about strategies, strategy profiles and personal utility functions.

6 The components of dialogue structure in game theory

We proceed to assign their intended interpretation to the components of the game. Since the terms "speaker" and "hearer" designate particular roles in conversational interaction that change during a conversation, we use the term *participants* to refer to the players in a dialogue game. We have already introduced *interventions* as structured sets of speech acts. We now turn to the problem of providing a reasonable interpretation for the concept of strategy. Above, we hinted that this may not be a simple issue. The informal definition of a strategy in game theory is not far from the intuitive idea of "a set of instructions". But the formal concept of strategy can be at variance with the intuitive idea of a plan of action. As Rubinstein points out repeatedly, the standard game-thoretical definition of strategy in an extensive game is counter-intuitive in that it requires a completely defined plan of action. It is not simply a *tuple* of plans, one for each individual. In some problems, the formal definition of a strategy will spell out the course of action for every history after which it is the participant's turn to move, even for moments that would never be reached if the strategy is followed. This is the issue of over-specification of strategies.[18] Another problem associated with the interpretation of the concept of strategy is associated with random acts or acts regulated by probabilistic rules. One way of constructing a mathematical model for this situation is to suppose that the adoption of a mixed strategy means that a participant will use some randomness in his behavior, for instance, using some random device to decide of his action. Outstandingly clever solutions have been advanced by game theorists such as J. C. Harsanyi and R. J. Aumann to circumvent this difficulty. In the present paper, following Stalnaker, we adopt a special interpretation for the concept of mixed strategy. This interpretation, called the belief — or epistemic — interpretation, may be thought of as less counter-intuitive with respect to the issues of completion and random acts.[19] It replaces the over-specification of the participant's plan by lack of certainty on the part of the *other* players. According to this interpretation, the players in a game do not randomize; they choose definite actions. But other players need not know which action they choose. So the mixt acts represent their uncertainty, their partial belief about his choice. This interpretation eliminates the need for using some random device in the process of decision making. The belief interpretation is usually considered favorably by those who raise doubts about the applicability of the concept of strategy. To be more specific, under this interpretation, a mixed strategy

[18] See Osborne and Rubinstein [12] p. 92 and Rubinstein [16] p. 77.
[19] See Aumann, [2] and Rubinstein, [16] p. 79.

becomes a belief held by all other players concerning a player's action.[20] The fact that this epistemic interpretation is available for Nash equilibrium can be taken as an independent argument in favor of using Nash equilibrium as a solution concept for simple games.

In a dialogue with an internal discursive goal, the definition of the dialogue must determine the class of possible strategies available. The utility function of each participant will motivate his choices among strategies. For simplicity, we occasionally talk as if we were concerned only with two-person dialogues. The beliefs of a participant include all the situations such that the other participant may be in these situations for all the participant knows. It would not be reasonable, and is not required, that absolutely all the knowledge of the participants be common knowledge. It is only the knowledge that pertains to relevant components of the dialogue game that need to be common.

To interpret the utility function associated with each participant, *at the stratospheric height of simple games*, we assume that the participants are rational in the sense of being relevance and efficiency maximizers. Of course, it is proper action to make idle talk in some dialogues. It is also possible to exploit the occasion of a conversation to obtain irrelevant information with respect to the thematic conditions associated with a discursive goal. We simply ignore these complications in the formal representation of simple dialogue games. The more complex form of dialogues can be explained as variations on the basic structure of simple dialogues. Larger subsets of the class of all possible dialogues can be defined inductively by defining a class of functions that construct dialogues from simple dialogues. We claim that these basic components are sufficient to construct the concept of solution for a simple dialogue. We also claim that they provide a sufficient conceptual apparatus to spell out the parameters of a family of dialogues as games. To support these claims, the parallel between the definition of success for a conversation and the concept of solution for a game must be established on firmer ground.

7 Success in dialogues and solutions for simple dialogue games

In Vanderveken's theory, "speakers succeed in holding a conversation of a certain type in making their successive utterances in a speech situation if and only if first, the theme of their conversation satisfies the thematic conditions of their discourse type, secondly, they achieve the discursive goal of that discourse type on the theme with the required mode of achievement, thirdly, they presuppose that the required background conditions obtain and finally they express all the mental states required by the sincerity conditions of their discourse type".[21] The goal is to take this success-value semantics from the illocutionary logic of dialogue and interpret it in terms of a general concept of solution from game theory. We have noted earlier that the general idea of a solution is a systematic description of the outcomes that may emerge in a family of games.[22] The definition of success and the specification of strategies that define a solution are elements of the theoretical description of a possible dialogue game. Both are *abstract* in that they leave

[20] This interpretation and others are reviewed in Rubinstein [16] p. 77 sq.

[21] Vanderveken [25] p. 253.

[22] See Osborne and Rubinstein [12] p. 2.

aside many aspects of real dialogues. In both cases, the analysis starts from the end point, the goal. In the present paper, we put forward the view that in the case of conversations that involve strategic interactions, the concept of success and the concept of solution coincide. In other words, the question "What is required for a conversation of a given type to be successful?" is essentially equivalent to the question "How can a game with a given structure be played optimally by rational agents?" Both the concept of success and the concept of solution are put to the test with the same form of reasoning. To the question "Why should people be relevant and efficient and respect thematic conditions?" the proper answer is "They ought to, if they want to achieve the goal efficiently." Recall that the definition of success contains an "if and only if". Note the "only if" part and apply *modus tollens*. The same reasoning is appropriate and familiar in the context of game theory. To the question "Why should people behave optimally and play the Nash equilibrium?", the proper answer is "In real life, people deviate from optimal paths, but the game-theoretical analysis predicts that if they deviate, either they gain nothing by doing so or their action profile will not be an equilibrium after all. In either case, the speaker's profile is self-defeating with respect to the goal, incoherent with respect to the speaker's beliefs, or simply irrational with respect to the speaker's utility function." We need to keep in mind that a solution is an abstract description of the logical structure of a game in strategic form. The intended goal of this abstract description is to provide a systematic formulation of the necessary features that successful dialogues possess.

8 Nash equilibrium as a solution concept for simple dialogues

It seems natural to consider the application of Nash equilibrium as a concept of solution in the context of dialogue game theory. The general idea of a Nash equilibrium is to describe an action profile for each participant such that every participant acts optimally given the other's action. There is a simple fact about Nash equilibrium that is relevant for our application to dialogues. Consider the following assumptions: (1) each participant is rational, (2) each knows his own payoff function and (3) each knows the strategy choices of others. From this it follows by definition that the players choices are a Nash equilibrium.[23] This simple fact indicates how little is required for a simple game —a simple dialogue— to have a solution, in other words, to be *playable*. As we have seen in the previous sections, there is a close analytical link between the concept of success for a dialogue type and the concept of solution for a game. This modest claim is the central point of our contribution and the starting point indicating that a lot of future work is needed. In his remarkable paper listed as [20], R. Stalnaker explained in great detail how to link the concept of model for a type of epistemic logic and the concept of solution for a game. We make use of his analysis in the next section in order to show how the elements of his framework can be applied to dialogues along the lines of the interpretation indicated in the previous sections of this paper. Stalnaker presents two characterization theorems that show the adequacy of his framework, in a way that is analogous to what soundness and completeness show for a logical system. We will

[23] See Aumann and Brandenburger [3] for further discussion of this observation and a treatment of interactive belief systems.

review and discuss these theorems will be stated in the closing section of the present paper. Both theorems can be applied to dialogue games with an internal discursive goal. However, this claim must be set in perspective. The logical structures of dialogue games under consideration are very abstract and our approach neglects many details. This is why we talk about these models as solutions for *simple* games. In order to work out a detailed game-theoretical pragmatics of dialogues on the basis of models of this type, we would need to build a more complex theory of dialogues and include more elements to represent discursive goals, mental states expressed by participants and modes of achievement of discursive goals.

At this point, the main stumbling block that we have met concerns the concept of strategy. It is only on very rare occasions that the participants in a dialogue will pick out a strategy that they will play deterministically, i.e. as if following a set of instruction. In game theory, such a plan is called a *pure strategy*. Therefore, we find it more appropriate to consider the mixed extension of a strategic game. In our discussion of the concept of strategy, we mentioned a difficulty with the interpretation of mixed strategies. Recall that since a player does not know what the other player will do or say, the standard analysis describes him as randomizing over the set of feasible strategies that are available to him. This feature of the standard interpretation of mixed strategies has been criticized by many as being artificial and counter-intuitive. Since we adopt the belief interpretation strategic games, we use the less problematic interpretation that is available. Apart from this difficulty that is not peculiar to the analysis of dialogue games, the interpretation of dialogue game theory can be built using this formal setting as the basic structure.

9 Stalnaker's epistemic models with application to dialogue games

We now turn to the basic ingredients of the logical semantics for a dialogue game. Skipping over many features, we review and reinterpret many definitions of the framework of Stalnaker [20].[24] A dialogue game, or more precisely, a dialogue interpreted as a game in strategic form Γ is defined as a structure $\langle N, \langle C_i, u_i \rangle_{i \epsilon N} \rangle$ where N is the set of participants, C_i is a finite set of alternative strategies available to participant *i*, and u_i is participant *i*'s utility function, a function from strategy profiles (members of $C = \times_{i \epsilon N} C_i$) into real numbers representing utility values.[25] The definition of a dialogue game determines the set of possible strategies available. A model for a dialogue game is a structure $\mathscr{M} = \langle W, \mathbf{a}, \langle R_i, P_i, S_i \rangle_{i \epsilon N} \rangle$ where W is a non-empty set, the set of possible worlds; **a** is a designated member of W, the actual world of $\mathscr{M}$; each R_i is a binary relation on W that determines for each possible world which possible world are compatible with participant i's beliefs in that world. The function P_i, a distinctive feature of this otherwise familiar model structure, is an additive measure function on the subsets of W, that determines i's partial beliefs in each world. P_i is defined over the full algebra of subsets $\mathscr{P}(W)$; by definition we know

[24] See also Stalnaker [20] and [19]. Note that Stalnaker constructs a semantics for games *in general*. He makes no reference to *dialogues* or anything remotely resembling what we call here *dialogue games*.

[25] The definition of a dialogue game in strategic form would be much longer.

1. $\Lambda \epsilon \mathscr{P}(W)$ (Λ is the empty set)
2. $\mathscr{P}(W)$ is closed under finite unions, i.e.
   for $x_n^{'} \epsilon W$, and $i = 1 \rightarrow n$, $\cup x_i^{'} \epsilon W$
3. for all $X^{'} \epsilon \mathscr{P}(W)$, $(W - X^{'}) \epsilon \mathscr{P}(W)$

Moreover, P_i is a real-valued *additive* function; i.e. it verifies

$$\text{for all } X, Y : X - Y = \Lambda \text{ and } X, Y \epsilon \mathscr{P}(W)$$

$$P_i\,(X \cup Y) = P_i(X) + P_i(Y)$$

In Stalnaker [19], a further proviso is made explicit to insure that P_i is nonzero for all w. It is assumed that the models are finite and that the measure functions assign nonzero probability to every non-empty subset in $\mathscr{P}(W)$.

$S_i(w)$ is a function that represents player i's strategy choice in world w as a function from W into C_i. The set of propositions that a participant i believes in world x is the set of those propositions that are true for all y such that $x\ R_i\ y$. Note that i believes that ϕ is equivalent, in the present context, to i *is certain that* ϕ in x. This amounts to defining the beliefs of i as the set of situations that, for all i knows, another participant might be in. More explicitly, for any proposition ϕ, the proposition that *i fully believes that* ϕ is the set

$$\{x \epsilon W : \{y \epsilon W : (x\ R_i\ y)\} \subseteq \phi\}$$

The set of possible worlds that are compatible with a participant's belief in world w is the set $(x : w\ R_i\ x)$. The degree to which i believes proposition ϕ in a situation *w*, in notation $P_{i,w}(\phi)$, is defined as

$$P_{i,w}(\phi) = \frac{P_i(\phi \cap \{x : w\ R_i\ x\})}{P_i(x : w\ R_i\ x)}$$

This formula is an instance of a familiar principle, the quotient rule for conditional probability. Simply put, the formula tells us how to compute the probability of ϕ relative to i's beliefs in w; i's beliefs is the set $(x : w\ R_i\ x)$. In our view, this concept is central for the semantics of dialogues because dialogues should be analyzed as interacting belief systems. The compatibility relations R_i are not reflexive, but serial, euclidean and transitive. It is not desirable that R_i be reflexive since an agent's beliefs may not be consistent with the world in which he has those beliefs. Seriality $[(\mathrm{x})(\exists \mathrm{y}) x R_i y)]$ amounts to the assumption that each agent-indexed belief set is consistent and permits that some of what the agent believes may be false in the world in which he has those beliefs. The euclidean axiom $[(x)(y)(z)((xR_iy)\ \&\ xR_iz) \supset yR_iz]$ and the transitivity of R_i encodes the assumption that agents have introspection with respect to their own beliefs. These properties of R_i validate the axiom set for belief from the basic epistemic logic known as KD45.

Rationality is defined in the usual way as maximizing expected utility over outcomes, where the expected utility of a strategy choice for a participant i at a world w is the weighted sum of the utilities of the possible outcomes of the choice. In the case of a dialogue, as we have stated above, maximizing "utility" can be interpreted as the basic

quest for relevance and efficiency in communication but it may also be interpreted in terms of any other personal agenda that participants may have. A participant is rational if the expected utility of his chosen strategy is equal or greater than any other possible strategy choice he might have made given the strategy choice of the other participants.

$$A_i : \text{Participant } i \text{ is rational} =_{\text{df}}$$

$$\{x \epsilon W \ : \ \text{for all } s \ s \epsilon C_i, \sum_{y \epsilon W} P_{i,x}(y) \ \times \ u_i(S(y))$$

$$\geq \ \sum_{y \epsilon W} P_{i,x}(y) \ \times \ u_i(s, S_{-i}(y))\}$$

The utilities of the possible outcomes are weighted by i's degree of belief in the alternative states of the world. The proposition *that everyone is rational* is the intersection of the $A_i^{'}$

$$A = \cap_{i \subset N} A_i$$

The concept of *common belief* (all agents believe that ϕ, all believe that all believe that ϕ, and so forth) is defined as the $transitive\ closure\ R^*$ of the R_i relations. This can be said explicitly: xR^*y is true if and only if there is a sequence $\langle w_1, ..., w_m \rangle$ such that $w_1 = x$ and $w_m = y$ and for all j from 1 to $m - 1$ there is a player i such that $w_j R_i w_{j+1}$. The proposition *that there is common belief that everyone is rational* is defined likewise

$$Z = \{x \ \epsilon \ W : \{y \ \epsilon \ W : x \ R^* \ y\} \subseteq A\}$$

These are the basic definitions needed to work out an account of dialogue game theory. Within this framework it is possible to formulate the set of Nash equilibrium strategies for a n-person dialogue in model-theoretic terms.

The adequacy of this semantics can be established in connection with a pair of theorems that bridge the gap between game theory and the model theory of epistemic logic. These theorems show how solution concepts such as rationalizability and Nash equilibrium for a game are characterized by a class of models. Before we state the theorems we need to introduce a few more more concepts. We need to explain a standard method for constructing a solution for a game known as strategy elimination.[26] To explain this notion, we want to generalize the definition of a *strictly dominated action*, a familiar notion from decision theory. We say that a choice option in a decision problem is strictly dominated if for any set of beliefs of the chooser, that option could not be optimal.[27] This can be expressed in the framework of Savage-style decision theory with states S_1 to S_m and acts A_1 to A_n. Let $j, j'...(k, k'...)$ be such that $1 \geq j, j'...(k, k'...) \geq m, (n)$; we say that an act A_k is strictly dominated by another act $A_{k'}$ if and only if $A_{k'}$ is a better response to every probability distribution over states S_j. We want to apply the relation of strict dominance to strategies and probability mix of alternative actions; to

[26] For a detailed formulation, see Myerson [11] p. 57 sq.

[27] Myerson [11] p. 28.

do this, let $A_k\,'s$ range over sets of acts and identify each act A_k with the set of which A_k is the only member. The notion of strict dominance is therefore applicable to sets of acts, it can be applied to strategies relative to each player i.

We now turn to the notion of strategy elimination. The iterated elimination of strictly dominated strategies is, as the name indicates, a process of testing pairs of strategies in C_i, successively eliminating dominated strategies (for player i).[28] This process yields residual games up to the game Γ^* such that for every i in N^*, the set C_i^* contains only strategies that are best response for i to probability distributions over $\times_{j \epsilon N^* - i}\, C_j^*$, the strategies chosen by players other than i. In the analysis of intelligent dialogues, as in game theory, the expectations formed by the participants with respect to the predictable behavior of other participants is the key element that explains how strategic social interaction can be successful and efficient. Stalnaker's first theorem shows that common belief in rationality is sufficient to characterize a solution concept known as rationalizability. It follows in part from the following proposition, first proved by G. W. Pearce: [29] *An action is a best response to some probability distribution over states S_1 to S_m if and only if it is not strictly dominated.*

Stalnaker's Theorem 1 *For any strategic form game, the set of strategies that survive the iterated elimination of strictly dominated strategies is characterized by the class of models in which the players have common belief that all player's choose strategies that maximize expected utility.*

Assumptions about participants in a dialogue game can be viewed as conditions that restrict the class of models for a game with a given structure. Such is the proposition that each participant in a dialogue game is rational or that there is common belief among players in rationality. More precisely, we say that a proposition restricts the class of models for a game structure to those models in which the proposition is true in the actual world of the model. In the first theorem, the clause that "players have common belief that all players choose strategies that maximize expected utility" is such a restriction. The proof of the theorem shows that a game in which the set of strategies is restricted to those that survive iterated elimination of dominated strategies will satisfy the condition. It also shows how to correlate strategies (for a player i) that survive iterated elimination with probability distributions. If we apply the theorem to a playable dialogue game, we can take it to mean that common belief in the fact that all players are rational will guarantee the existence of set of sequences of interventions, one for each player, such that each is a series composed of best responses to the interventions of other players. The theorem reinforces our view, stated in previous sections of the present paper, that the possibility of success in a dialogue game, or "playability", can be grounded on the assumption that participants are interacting systems of beliefs, that they are rational, and that they form intelligent expectations about others. It also corroborates a guiding principle of our approach to the analysis of dialogues, an overall conjecture that has proven useful in game theory. This guiding principle predicts that the amount of beliefs, knowledge and inference capabilities that is assumed of the participants in a

[28] An exact formulation must account for the fact that the order in which strategies are eliminated may matter.

[29] See Pearce [15].

dialogue game is fairly limited and does not exceed what is required to build a formal analysis of the possibility of success of the dialogue game. The next theorem is also a characterization claim for a solution concept, but this time, it is about capturing the arguably more substantial solution concept of Nash equilibrium. The standard definition of this equilibrium is as follows. A *mixed strategy Nash equilibrium* for a finite strategic game is a mixed strategy profile $S(x)$ realized at world x with the property that for every player i every action in support of $S(x)$ is a best response to $S_{-i}(x)$, the action profile of the participant(s) other than i.

Stalnaker's Theorem 2 *For any two-person game, the set of Nash equilibrium strategies, interpreted as belief profiles, is characterized by the class of models in which each player knows the other's beliefs about his strategy choice, and each knows that the other is rational.*

In order to work with the belief interpretation of mixed strategies, we need to have, in addition to a strategy profile for each player, a belief profile defined as a sequence of probability distributions defined over the partial strategy profiles of other players. The proof of this theorem uses strategy pairs, $\langle m_1, m_2 \rangle$, to represent belief profiles. We note that this second theorem is formulated with a restriction to a two-person game and this is due to a technical property of the belief interpretation for mixed strategies. When the number of players is greater than two, the belief profile of a given player is not a mixed action but a probability distribution on *tuples* of actions of other players. [30] The proof of this theorem uses mainly the definition of Nash equilibrium and proceeds by constructing a model in which the probability distribution P_i has been defined so that the players have the same belief profiles in all possible worlds of the model. This entails that each player knows what the other player believes about his strategy choice. What is remarkable in this theorem is how little is required — only two rather weak conditions on the class of models — to obtain the characterization of a significant solution concept. As we have said earlier, Nash equilibrium can be understood as a very general description of success conditions for strategic interactions. Therefore, this solution concept applied to strategic interactions in dialogue games is a special instance of the general case.

References

1. Asher, N., Lascarides, A.: Logics of Conversation. Cambridge University Press (2003)
2. Aumann, R.J.: Correlated equilibrium as an expression of bayesian rationality. Econometrica 55, 1–18 (1987)
3. Aumann, R.J., Brandenburger, A.: Epistemic conditions for nash equilibrium. Econometrica 63(1161–1181) (1995)
4. Belnap, N.: Declaratives are not enough. Philosophical Studies 59, 1–30 (1990)
5. Bratman, M.E.: Faces of Intention. Cambridge University Press (1999)
6. Grice, H.P.: Logic and conversation. Harvard University Press (1989)
7. Kuhn, H.W.: Extensive games and the problem of information. In: Kuhn, H, W., Tucker, A.W. (eds.) Contributions to the Theory of Games, vol. II. Annals of Mathematics Studies, Princeton University Press (1953)

[30] Aumann and Bradenburger [2] underlines that different epistemic conditions are associated with two-person and n-person games.

8. Lewis, D.: Scorekeeping in a language game. Journal of Philosophical Logic 8, 339–359 (1979)
9. Mann, W.C.: Dialogue games, conventions of human interactions. Argumentation 2, 511–532 (1988)
10. Mann, W.C.: A single theory of dialogue. Tech. rep., SIL international (2001)
11. Meyerson, R.B.: Game Theory: Analysis of Conflict. Harvard University Press (1991)
12. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press (1994)
13. Paquette, M.: Solutions for simple dialogue games (2002). In: Vanderveken, D., Vernant, D. (eds.) Logique et Dialogue. College Publications (Forthcoming)
14. Parikh, P.: The Use of Language. CSLI, Stanford (2001)
15. Pearce, D.G.: Rationalizable strategic behavior and the problem of perfection. Econometrica 52, 1029–1050 (1984)
16. Rubinstein, A.: Economics and Language: Five Essays. The Churchill Lectures in Economic Theory, Cambridge, Cambridge University Press (2000)
17. Searle, J. R. Parret, H., Verschuen, J.: (On) Searle on Conversation. John Benjamins (1992)
18. Searle, J.R., Vanderveken, D.: Foundations of Illocutionary Logic. Cambridge University Press (1985)
19. Stalnaker: Extensive and strategic forms: Games and models for games. Research in Economics 53, 293–319 (1999)
20. Stalnaker, R.: On the evaluation of solution concepts. In: Bacharach, M.O.L., Gérad-Varet, Mongin, P., Shin, H.S. (eds.) Epistemic Logic and The Theory of Games and Decisions, vol. 20, pp. 345–364. Kluwer Academic (1997)
21. Stalnaker, R.: Knowledge, belief, and counterfactual reasoning in games. In: Bicchieri, C., Jeffrey, R., Skyrms, B. (eds.) The Logic of Strategy, chap. 1, pp. 3ñ38. Oxford University Press (1999)
22. Van Benthem, J.: Rational dynamics and epistemic logic in games. International Game Theory Review 9(1, 2), 1:13–45, 2:377–409 (2006)
23. Van Benthem, J.: Logical Dynamics of Information and Interaction. ILLC, Amsterdam (2010)
24. Vanderveken, D.: Meaning and Speech Acts, vol. I and II. Cambridge University Press (1990–91)
25. Vanderveken, D.: Illocutionary logic and discourse typology. Revue Internationale de Philosophie pp. 243–255 (2001)
26. Von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Princeton University Press, third edn. ((1944) 1953

Author Index

Cristina, Baroglio, 2

Elisa, Marengo, 2

Jamal Bentahar, 53
Jan Corfixen, Sørensen, 37

Matteo, Baldoni, 2
Michel A. Paquette, 73
Mohamed El-Menshawy, 53
Munindar P., Singh, 19

Nørregaard, Jørgensen, 37

Rachida Dssouli, 53

Scott N., Gerard, 19

Vanderveken, Daniel, 1

Wei Wan, 53