Model Checking Command Dialogues

Rolando Medellin, Katie Atkinson and **Peter McBurney**
Department of Computer Science
University of Liverpool, UK
{medellin,katie,mcburney}@liverpool.ac.uk

Abstract

Verification that agent communication protocols have desirable properties or do not have undesirable properties is an important issue in agent systems where agents intend to communicate using such protocols. In this paper we explore the use of model checkers to verify properties of agent communication protocols, with these properties expressed as formulae in temporal logic. We illustrate our approach using a recently-proposed protocol for agent dialogues over commands, a protocol that permits the agents to present questions, challenges and arguments for or against compliance with a command.

Keywords: agent communication, command dialogues, CDP, interaction protocols, model checking, NuSMV.

Introduction

The last two decades have seen considerable research on agent communication languages and agent interaction protocols. In the typical formulation, such as the generic agent language *FIPA ACL*, developed by FIPA (FIPA 2002), agent utterances are represented as two-layers: an inner layer of material directly related to the topic of the discussion, and an outer- (or wrapper-) layer comprising a speech act. An example of such a wrapper is the FIPA ACL locution, *inform(.)*, which allows the agent uttering it to tell another agent some statement which the first agent believes to be true. With such a structure, the same set of locutions may be used for dialogues on many different topics, on each occasion wrapping different content. Such generic languages create state-space explosion problems for intending agent dialogue participants, however, and so research attention has also been given to the design of agent interaction protocols. These may be viewed as agent communication languages designed for more specific purposes than is a generic language, in the same way, say, that a standard (human) ascending-price auction protocol is more specific than is a natural human language, such as English. For a recent review of research in agent interaction protocols, see (McBurney and Parsons 2009).

As with any software, verification that agent interaction protocols have desired properties (or do not have undesired properties) is important. In this paper, we explore the use of model checking technologies for verification of properties of agent interaction protocols. In order for model checking approaches to be applied, we need to express the properties in a logical formalism, and we use a branching-time temporal logic for this. We illustrate the approach on an agent protocol designed for arguments over commands, called **CDP** (Atkinson et al. 2009). This protocol was selected because it allows for argument between the participants, and because it is sufficiently complex that an automated approach to verification of protocol properties should prove of value to human software engineers.

The structure of the paper is as follows. The next section summarizes the command dialogue protocol, **CDP**. This is followed by a brief discussion of model checking and of **NuSMV**, the model checker we have used for this work. After that, we present the results of model-checking CDP, showing the graphical representation of the protocol, and the temporal logical representation of the properties we desire to verify. We finish with some concluding remarks and indications of areas for future work.

Command Dialogue Protocol

Commands are instructions issued by one agent to one or more other agents to execute some action (or not), or to bring about some state. Not all commands are issued legitimately, and even those which are legitimate may require subsequent elaboration or explanation before they can be executed. Thus, it is possible for agents to engage in an argumentative interaction over a command. As explained in (Atkinson et al. 2009), the rise of distributed computer systems and rival centres of control of the elements of such systems make commands and agent dialogues over commands increasingly common. Indeed, Hyper-Text Transfer Protocol (HTTP) (Network Working Group 1999) may be viewed as a protocol for two-agent dialogues over commands, although it is rather impoverished in terms of the commands enabled to be represented and the arguments permitted over them. In recent work (Atkinson et al. 2009), a formalism for the representation of commands and a dialogue protocol for argument over commands was presented, making use of an argument scheme for action proposals. In this formalism, the agent issuing the command was called *Commander*, while the intended recipient was called *Receiver*. The Command Dialogue Protocol (**CDP**) allowed Commander

to issue a command to Receiver, and allowed Receiver to question, challenge, refuse or accept this command. If questioned or challenged, Commander could respond with additional information or arguments in support of the original command, and/or re-iterate it, modify it, or retract it.

Command dialogues are not explicitly mentioned in the Walton and Krabbe typology of human dialogues (Walton and Krabbe 1995). In a dialogue where a command has been issued, but not yet refused or accepted, the participants may enter into interactions which resemble those in the Walton and Krabbe typology, for example, Information-seeking, Inquiry, Persuasion Negotiation, Deliberation or Eristic dialogues. Not all command dialogues will have all such interactions, however, and accordingly we believe it appropriate to consider Command dialogues as a type of dialogue distinct from those in the Walton and Krabbe list.

We now present an outline of the Command Dialogue Protocol (CDP) of (Atkinson et al. 2009), which uses an argument scheme for action proposals to specify commands. In an argument scheme, arguments are presented as general inference rules where, under a given set of premises, a conclusion can be presumptively drawn (Walton 1996). The argument scheme presented in CDP states that : *given the social context X, on the current circumstances R, action A should be performed to achieve new circumstances S, which will realise some goal G and will promote some value V*. This scheme allows commands to be justified through the promotion or demotion of some social value or interest, where a certain state or circumstance is achieved. Justification is based on current circumstances and elements of the social context. The CDP specifies the rules to formally represent imperatives in a multi-agent dialogue and provides means by which the participants may question, challenge, justify, accept or reject a command. Commands are represented as action proposals to the Receiver similar to the representation in (Atkinson, Bench-Capon, and McBurney 2004). In contrast with proposals or promises, commands require a set of preconditions in a regulatory environment to be executed validly. A command represents a presumptive argument attacked by a set of critical questions whose answers may defeat the initial argument or command. Critical questions represent questions the Receiver could pose to the Commander either to question or to challenge the command such that more evidence will be needed to justify it. Questions about the appropriateness, suitability, feasibility and normative rightness could be posed to the Commander.

The CDP syntax enables agents to interact using seven locutions: issue, accept, reject, question, challenge, justify and retract (Atkinson et al. 2009). Locutions to issue or retract a command are inherent to the Commander and are comprised of options to state propositions defined in the initial argumentation scheme. As for the Receiver, the protocol defines locutions to respond to a command by accepting, refusing, questioning or challenging it. Expanding the 'question' locution CDP grows to 76 locutions[1] available to Receiver when questioning or challenging a command. Locutions to challenge and provide information can be used by both agents participating in the dialogue.

[1]In case this number of locutions is thought prolix, note that CDP is intended for machine-to-machine communications; for comparison, the machine interaction protocol, Hypertext Transfer Protocol (HTTP), defines 41 standard status-code responses to a GET command, and allows for several hundred additional non-standard codes (Network Working Group 1999).

Model Checking

The verification of multi-agent systems showing that a system is correct with respect to stated requirements is an increasingly important issue (Bordini et al. 2006). Currently, the most successful approach to the verification of computer systems against formally expressed requirements is that of Model Checking (Clarke, Grumberg, and Peled 1999). Model checking is an automatic technique for verifying finite-state reactive systems, such as communication protocols. Given a model of a system M and a formula φ (representing a specification), model checking is the problem of verifying whether or not φ is true in M ($M \models \varphi$). In model checking, the design to be verified is modeled as a finite state machine, and the specification is formalized by writing temporal logic properties.

An efficient search procedure is used to determine whether or not the state-transition graph satisfies the specifications (Clarke, Grumberg, and Peled 1999). The power of model checking is that it is exhaustive, no regions of the operating space are unexplored. Although model checking techniques have been most widely applied to the verification of hardware systems, they also have been used in the verification of software systems, protocols, (Walton 2004), agent dialogues (Endriss 2006; Endriss et al. 2004) and multi-agent-systems (Wooldridge et al. 2002; Bordini et al. 2006).

NuSMV

The possibility of verifying systems with realistic complexity changed dramatically in the late 1980s with the discovery of how to represent transition relations using ordered binary decision diagrams (BDD) (Clarke, Grumberg, and Peled 1999). A BDD is a data structure that is used to represent a Boolean function. The original model checking algorithm, with the new representation for transition relations, is called symbolic model checking. The symbolic model verifier (SMV) system is a tool for checking finite state systems against specifications in the temporal logic CTL (Computation Tree Logic) (McMillan 1999). The input language of SMV is designed to allow the description of finite state systems and allows a rich class of temporal properties, including safety, fairness, liveness and deadlock freedom. NuSMV [2] is a reimplementation and extension of SMV and has been designed as an open architecture for model checking. This

[2]NuSMV is a symbolic model checker developed as a joint project between the Formal Methods group in the Automated Reasoning System division at ITC- IRST, the Model Checking group at Carnegie Mellon University, the Mechanized Reasoning Group at University of Genova and the Mechanized Reasoning Group at University of Trento (Cimatti et al. 2000).

new version is aimed at reliable verification of industrially sized designs, for use as a back-end for other verification tools and as a research tool for formal verification techniques (Cimatti et al. 2000). NuSMV2 uses a technique called Bounded Model Checking (BMC), which uses a propositional SAT solver rather than BDD manipulation techniques. SAT or propositional satisfiability is the problem of determining if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to TRUE (Biere et al. 1999).

Model Checking CDP

Rather than propose a new model checking algorithmic approach to verify agent-communication protocols as in (Bentahar, Moulin, and Ch. Meyer 2006) our aim is to use existing model checkers to validate properties on a dialogue protocol.

In (Walton 2004) a Multi-Agent Dialogue Protocol (MAP) is used to define the communicative process between agents considering complex, concurrent and asynchronous patterns. To verify the MAP protocols Walton uses the SPIN Model checker (Holzmann 2004) translating the MAP representation into the PROMELA language that SPIN uses as input language and then construct LTL formulas to validate against the PROMELA representation. This is probably the most similar approach to what we intended here. The main difference is that the MAP is a generic language to define communicative processes and we are focusing on a single protocol.

Agent dialogue protocols exhibit behaviour characterized in terms of execution traces which can be represented as branching trees. Trees can be represented in terms of a state-transition system and then translated into the NuSMV input language. The NuSMV model checker uses an exhaustive search procedure to determine whether or not a specification or property satisfies the modeled system. We aim to take the advantages of the NuSMV model checker to validate properties of the protocol.

We focus on the CDP (Atkinson et al. 2009) and its desirable properties. The protocol is represented with the NuSMV input language, and properties we want to validate in the model are temporal CTL formulae. CTL formulae can be evaluated in transition systems (Clarke, Grumberg, and Peled 1999) where the states are dialogue states and the transitions are the protocol valid locutions. In case the property is not valid, a counterexample is generated in the form of a sequence of states. In general, properties are classified to "safety" and "liveness" properties. Safety properties express what should not happen (equivalently, what should always happen), and liveness properties declare what should eventually happen.

Among the properties we want to verify for the protocol are:

1. Does any infinite loop or deadlock [3] situation exist in the protocol? If a deadlock or loop does exist, which dialogue sequence leads to that loop or deadlock?
2. Can we reach every outcome state? The motivation behind this property is to ensure the protocol has valid paths in all the possible combinations of the dialogue.
3. Is it possible to utter a particular locution in a particular state? This approach suggests a way to validate locutions in a dialogue.
4. Given a particular state (either an end-state or not), is there a valid dialogue sequence to reach that state?
5. Given a particular state, is there a dialogue sequence which avoids that state? An agent may wish to know if it can enter into a dialogue while avoiding particular states, e,g. concessions to other participants.
6. If the dialogue has reached a particular state, is a particular outcome state still reachable?. It could be the case, for example, that certain intermediate states in a dialogue preclude some outcome states.

[3] A deadlock is a situation wherein two or more competing actions are waiting for the other to finish, and thus neither ever does.

State-transition diagrams

The CDP can be modeled as a high level state-transition diagram where states represent dialogue states and transitions represent valid locutions. The diagrams presented in this section represent a command dialogue in an abstract way, leaving out explicit details about the content of messages, concurrency and the environment. Dialogue states are represented as circles and locutions as directed arrows labelled with valid locutions. Diagrams capture the protocol rules for agents engaged in a command dialogue specifying the path to reach any outcome state.

The dialogue states for the CDP are: *Initial, ReceiverCommanded, CommanderQuestioned, CommanderChallenged, ReceiverwithEvidence1, ReceiverwithEvidence2, CommandRetracted, CommandAccepted and CommandRefused* (we number the ReceiverwithEvidence status because we want to distinguish the state where evidence comes from a question from that where it comes from a challenge). The locutions for the CDP are: *command, question, challenge, provide, refuse, retract and accept*. We are excluding from the model for now the mental states of the agents and the environment state. We also have not yet considered the critical questions from (Atkinson et al. 2009) within our model.

The diagram in Figure 1 represents dialogue states numbered from s0 to s8 and the valid transitions for each state. The diagram shows how locutions are constrained depending upon the dialogue state, for example, we can only access the state where the command has been accepted (s7) from the states { s1, s5, s6 }, where the Receiver has been commanded or has been provided with evidence. From the moment an Agent *C* (the Commander) issues a command a range of valid locutions is available for each agent. Valid Commander locutions are represented with doted arrow-lines and Receiver locutions are represented with normal arrow-lines.

The CDP assumes a strict-turn-taking only for the Receiver that needs to wait for the Commander's locution. Assuming the agent is rational and because of a change in

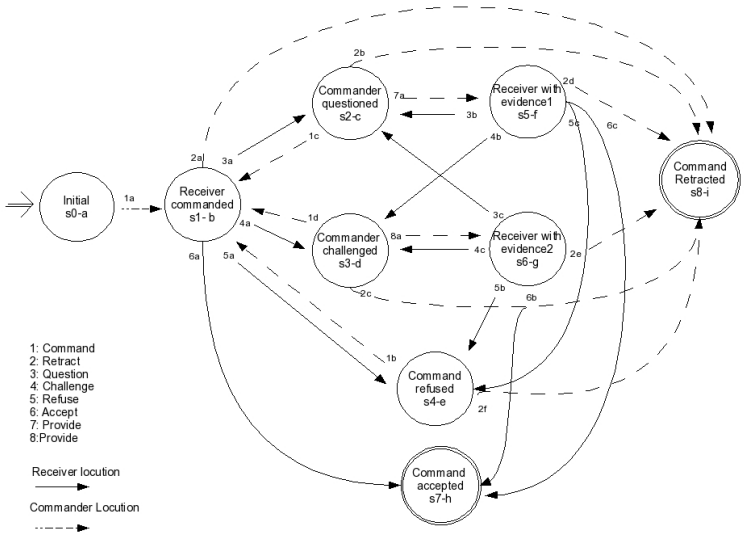

Figure 1: State-Transition Diagram for CDP

the environment the commander could retract or reissue the command at any time. If we assume a strict-turn-taking for the commander arrows, 2a, 2d and 2e would be left out the diagram.

As we have discussed, the finite state transition diagram can be expressed as a tree. We do this transforming outcome-states in final nodes of a tree repeating states as necessary. The tree-diagram representation is presented in Figure 2.

With this second diagram we can visualize all the possible computation paths for the protocol. Instead of representing a state just once, we repeat the state to avoid locutions returning to the same state. Loops are now represented as infinite paths and the paths to reach an outcome state are clearer. Since we are using a branching time temporal logic (CTL) this model is useful to construct temporal formulae to validate. To represent how the dialogue advances we associate a propositional value with each state and specify where the expression is true in each state. For the initial state, for example, we assign propositional variable 'a' and make it true only in that state. In this way we can construct temporal formulae with propositional variables representing each state. We also assign a variable related to the 'turn' of each agent in the dialogue, represented by 'tc' in the case where the Commander is allowed to issue a locution, and 'tr' in the case of the Receiver. These variables allow us to construct temporal formulae related to the turn of an agent to issue a locution.

The properties we want to validate for the protocol could be rephrased as temporal properties related to the tree-model in Figure 2. In Table 1 the properties presented earlier are now rephrased and a temporal formula is associated for each one. CTL is built from path quantifiers and temporal operators. There are two path quantifiers, A and E, where A means "for every path" and E means "there exists a path" in the tree. A path is an infinite sequence of states such that each state and its successor are related by the transition relation. CTL has four temporal operators presented as follows: $\bigcirc\phi$ meaning "ϕ holds at the next time step" (where ϕ is a propositional formula), $\Diamond\phi$, "ϕ holds at some time step in the future", $\Box\phi$, "ϕ holds at every time step in the future" and $\phi U \psi$, "ϕ holds until ψ holds" (Biere et al. 1999).

Tree-oriented property	Temporal property
1. Do infinite paths exist in the tree-diagram?	$A\Diamond(A \bigcirc h \vee A \bigcirc g)$
1a. Which is the path?	Counterexample from 1
2. Is there a valid path to reach every outcome state?	$E\Diamond h$
3. Is a transition valid from a specific node?	$A\Box(c \rightarrow E \bigcirc d)$
4. Given a node, is there a path which leads to that node?	$E\Diamond g$
5. Given a node, is there a path which avoids that node?	$A\Box(\neg c \rightarrow E\Diamond i)$
6. If a command has been issued and questioned can the dialogue still reach a state where the command is accepted?	$A\Box(c \rightarrow E\Diamond i)$

Table 1. Properties and Temporal formulae.

NUSMV implementation

We use NuSMV for model checking because the input language allows us to represent the dialogue as a finite-state diagram and we can verify temporal properties in it. If the property specified does not satisfy the NuSMV model, the model checker offers a counterexample specifying the path where the formula fails to be true. Input to NuSMV is via a file which describes the state transition system in terms of

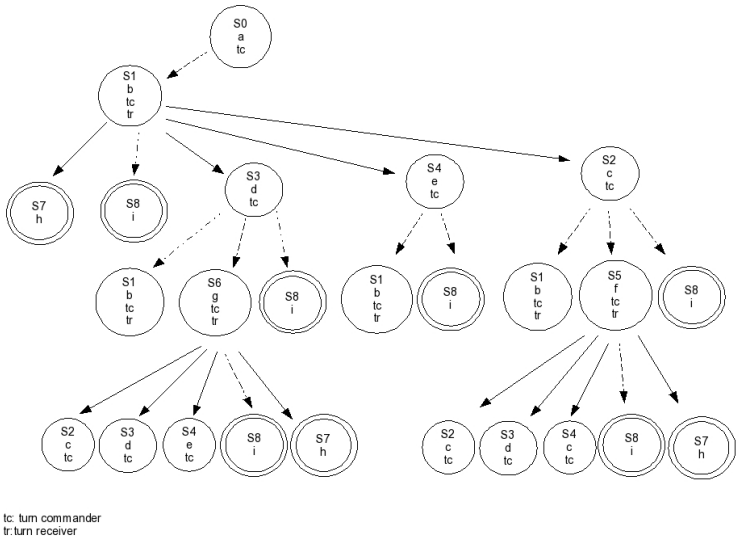

Figure 2: Tree diagram for CDP

variables and value assignments. Dialogue states are represented with a variable *state* that can obtain the value of any of the 9 states defined in the CDP, plus an error state to specify non-valid moves. Transitions are represented using the NuSMV *case* expression. For each state we define a set of possible next states that represent the valid transition relations. Expressions are evaluated sequentially and the first one that is true determines the resulting value.

We use the keyword *SPEC* in NuSMV to specify CTL properties to be checked. For example, to express if it is true that at some path there is a case where the command is retracted, we use the CTL formula $E\Diamond i$. Variable i represents the retracted state. In the NuSMV input language it is represented as *SPEC EF (i)*.

As for the variables related to the turn-taking we can specify a property to check if there is an option to issue a locution at every state (except on final states) $A\Box((tc|tr)|(\neg tc\&\neg tr\&h)|(\neg tc\&\neg tr\&i))$.

Preliminary Results

All the properties presented were translated to the NuSMV language and validated against the model. For the first property we are trying to check if there exists any infinite path on the model. The idea is to construct a formula that represents that eventually on all paths the final nodes could be reached. The property constructed in the NuSMV input language is $SPECAG(AXh \mid AXi)$. The formula is false for the model since the protocol allows the participants to engage in an infinite loop in several situations. Another way to construct this property without making reference to a particular state is $AGEX\top$ which states that there exist a path such that every node on that path still has some immediate successor.

Property number two ("Is there a valid path to reach every outcome state?") is True. The protocol allows to reach an outcome state in all paths. Property number three ("Is a transition valid from a specific state?") depends on the state we are choosing. In the example we are validating if issuing a locution from state "c" (commander questioned) is valid to a state "d" (commander challenged), in this case is False. This seems obvious if we analyse the diagram, but human visual inspection will not scale to larger and more complex protocols, nor operate at runtime. Property four ("Given a node is there a path which leads to that node?") tries to confirm if a valid path exists to reach a specific state. In the example the formula is true for state "g". Property number five ("Given a state is there a path which avoids that state?") is true for state "i" avoiding state "c". Finally property six ("If a command has been issued and questioned can the dialogue still reach a state where the command is accepted?") is True for the specified states.

Properties are closely related to the CDP protocol and the states that emerge from it; a more generic set of formulae may be desirable to develop. Nevertheless, we need to take into account that for dynamic verification, on-the-fly models need to be constructed and validated.

Conclusions

In this paper we have explored the possibility of using model-checking methods to automatically verify that a complex agent interaction protocol using argumentation has desired properties (or does not have undesired properties). Our key contribution has been to show by example that this is possible, using the model checker **NuSMV** to verify specific properties of the command dialogue protocol, **CDP**. Because this protocol supports multi-agent argumentation, it is reasonably complex and thus the value of automated verification approaches is likely to be considerable. Such verification could take place well prior to implementation, for example, as part of the human-led protocol design process.

Or it could take place at run-time just prior to invocation of the protocol, if agents were enabled to select and verify protocols on-the-fly at the moment before they enter into dialogue, as in (Miller and McBurney 2008). For agents having dynamic goals, on-the-fly verification of protocols will be important to ensure that protocols they use to engage in dialogue are able to achieve states currently desired or avoid states currently not desired.

In future work we intend to extend our model to account for the critical questions associated with the argument scheme as given in **CDP** since we have not considered them here. Our approach would be much more complex if we add rules and states considering the critical questions where more states and variables need to be added to the model. We also hope to investigate how our model can be extended to handle different types of dialogue in addition to CDP. For example, in (Atkinson, Bench-Capon, and McBurney 2004) a protocol is given for persuasion dialogues based on a similar argument scheme that is used for CDP, so this would be a good candidate protocol to model next. Additionally, some recent work (Atkinson and Bench-Capon 2009) has looked at how the argument scheme for practical reasoning discussed here can be formalised in terms of action-state semantics (Reed and Norman 2007). It would be also interesting to see how our approach to model checking dialogues could be applied to this representation.

Acknowledgements

Rolando Medellin is grateful for financial assistance from CONACYT of Mexico. We thank Clare Dixon and the anonymous reviewers for their comments. A similar version of this work was presented at the 2009 European Agent Systems Summer School in Italy in September 2009.

References

Atkinson, K., and Bench-Capon, T. 2009. Action-state semantics for practical reasoning. Proceedings of 2009 Fall Symposium on The Uses of Computational Argumentation.

Atkinson, K.; Bench-Capon, T.; and McBurney, P. 2004. A dialogue game protocol for multi-agent argument over proposals for action. *Autonomous Agents and Multi-Agent Systems* 11(2):153–171.

Atkinson, K.; Girle, R.; McBurney, P.; and Parsons, S. 2009. Command Dialogues. In Rahwan, I., and Moraitis, P., eds., *Argumentation in Multi-Agent Systems*, Fifth International Workshop, 93–106. Berlin, Heidelberg: Springer-Verlag.

Bentahar, J.; Moulin, B.; and Ch. Meyer, J.-J. 2006. A new model checking approach for verifying agent communication protocols. *Canadian Conference on Electrical and Computer Engineering, CCECE '06* 1586–1590.

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without bdds. In *TACAS '99: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, 193–207. London, UK: Springer-Verlag.

Bordini, R. H.; Fisher, M.; Visser, W.; and Wooldridge, M. 2006. Verifying multi-agent programs by model checking. *Autonomous Agents and Multi-Agent Systems* 12(2):239–256.

Cimatti, A.; Clarke, E.; Giunchiglia, F.; and Roveri, M. 2000. NuSMV: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer* 2:2000.

Clarke, E. M.; Grumberg, O.; and Peled, D. A. 1999. *Model Checking*. Springer.

Endriss, U.; Maudet, N.; Sadri, F.; and Toni, F. 2004. Logic-based agent communication protocols. In *Advances in Agent Communication Languages*, volume 2922, 91–107. Springer-Verlag.

Endriss, U. 2006. Temporal logics for representing agent communication protocols. *Agent Communication II: International Workshops on Agent Communication* 3859/2006:15–29.

FIPA. 2002. Communicative Act Library Specification. Standard SC00037J, Foundation for Intelligent Physical Agents.

Holzmann, G. 2004. *The SPIN Model Checker. Primer and Reference Manual*. Addison-Wesley.

McBurney, P., and Parsons, S. 2009. Dialogue games for agent argumentation. In Rahwan, I., and Simari, G., eds., *Argumentation in Artificial Intelligence*. Berlin, Germany: Springer. chapter 13, 261–280.

McMillan, K. L. 1999. The SMV system. *Cadence Berkeley Labs*.

Miller, T., and McBurney, P. 2008. Annotation and matching of first-class agent interaction protocols. In Padgham, L.; Parkes, D.; Mueller, J. P.; and Parsons, S., eds., *Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*. New York, NY, USA: ACM Press.

Network Working Group. 1999. Hypertext Transfer Protocol — HTTP/1.1. Technical Report RFC 2616, Internet Engineering Task Force.

Reed, C., and Norman, T. J. 2007. A formal characterisation of Hamblin's action-state semantics. *Journal of Philosophical Logic* 36:415–448.

Walton, D. N., and Krabbe, E. C. W. 1995. *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*. SUNY Series in Logic and Language. Albany, NY, USA: State University of New York Press.

Walton, D. N. 1996. *Argumentation Schemes for Presumptive Reasoning*. Mahwah, NJ, USA: Lawrence Erlbaum Associates.

Walton, C. 2004. Model checking agent dialogues. In *Declarative Agent Languages and Technologies II*, Lecture Notes in Computer Science, 132–147. Springer.

Wooldridge, M.; Fisher, M.; Huget, M.-P.; and Parsons, S. 2002. Model checking multi-agent systems with MABLE. In *AAMAS '02: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, 952–959. New York, NY, USA: ACM.