LNAI 4327

Matteo Baldoni
Ulle Endriss (Eds.)

Declarative Agent Languages and Technologies IV

4th International Workshop, DALT 2006
Hakodate, Japan, May 2006
Selected, Revised and Invited Papers

Springer

Matteo Baldoni Ulle Endriss (Eds.)

Declarative Agent Languages and Technologies IV

4th International Workshop, DALT 2006
Hakodate, Japan, May 8, 2006
Selected, Revised and Invited Papers

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editors

Matteo Baldoni
Università degli Studi di Torino, Dipartimento di Informatica
via Pessinetto 12, 10149 Torino, Italy
E-mail: baldoni@di.unito.it

Ulle Endriss
University of Amsterdam, Institute for Logic, Language and Computation
Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands
E-mail: ulle@illc.uva.nl

Library of Congress Control Number: 2006938418

CR Subject Classification (1998): I.2.11, C.2.4, D.2.4, D.2, D.3, F.3.1

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-540-68959-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-68959-1 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11961536 06/3142 5 4 3 2 1 0

Lecture Notes in Artificial Intelligence 4327

Edited by J. G. Carbonell and J. Siekmann

Subseries of Lecture Notes in Computer Science

Preface

The workshop on Declarative Agent Languages and Technologies (DALT), in its fourth edition this year, is a well-established forum for researchers interested in sharing their experiences in combining declarative and formal approaches with engineering and technology aspects of agents and multiagent systems. Building complex agent systems calls for models and technologies that ensure predictability, allow for the verification of properties, and guarantee flexibility. Developing technologies that can satisfy these requirements still poses an important and difficult challenge. Here, declarative approaches have the potential of offering solutions that satisfy the needs for both specifying and developing multiagent systems. Moreover, they are gaining more and more attention in important application areas such as the Semantic Web, Web services, security, and electronic contracting.

DALT 2006 was held as a satellite workshop of AAMAS 2006, the fifth International Joint Conference on Autonomous Agents and Multiagent Systems, in May 2006 in Hakodate, Japan. Following the success of DALT 2003 in Melbourne (LNAI 2990), DALT 2004 in New York (LNAI 3476), and DALT 2005 in Utrecht (LNAI 3904), the workshop again provided a discussion forum to both (a) support the transfer of declarative paradigms and techniques to the broader community of agent researchers and practitioners, and (b) to bring the issue of designing complex agent systems to the attention of researchers working on declarative languages and technologies.

This volume contains the 12 contributed articles that were selected by the Programme Committee for presentation at the workshop as well as three invited articles, originally presented as short papers at AAMAS 2006, that were extended by their authors. The volume also includes the article "Producing Compliant Interactions: Conformance, Coverage, and Interoperability" by Amit K. Chopra and Munindar P. Singh. Professor Singh, from North Carolina State University, was the invited speaker for this edition of DALT.

We would like to thank all authors for their contributions, the members of the DALT Steering Committee for their precious suggestions and support, and the members of the Programme Committee for their excellent work during the reviewing phase.

October 2006

Matteo Baldoni
Ulle Endriss

Workshop Organization

Workshop Organizers

Matteo Baldoni	University of Turin, Italy
Ulle Endriss	University of Amsterdam, Netherlands

Programme Committee

Marco Alberti	University of Ferrara, Italy
Natasha Alechina	University of Nottingham, UK
Grigoris Antoniou	University of Crete, Greece
Matteo Baldoni	University of Turin, Italy
Cristina Baroglio	University of Turin, Italy
Rafael Bordini	University of Durham, UK
Keith Clark	Imperial College London, UK
Ulle Endriss	University of Amsterdam, Netherlands
Benjamin Hirsch	Technical University Berlin, Germany
Shinichi Honiden	National Institute of Informatics, Japan
John Lloyd	Australian National University, Australia
Viviana Mascardi	University of Genoa, Italy
John-Jules Ch. Meyer	Utrecht University, Netherlands
Enrico Pontelli	New Mexico State University, USA
M. Birna van Riemsdijk	Utrecht University, Netherlands
Chiaki Sakama	Wakayama University, Japan
Wamberto Vasconcelos	University of Aberdeen, UK
Christopher Walton	University of Edinburgh, UK
Michael Winikoff	RMIT University, Melbourne, Australia

Additional Reviewers

Giovanni Casella
Valentina Cordì
John Knottenbelt

Steering Committee

João Leite	New University of Lisbon, Portugal
Andrea Omicini	University of Bologna-Cesena, Italy
Leon Sterling	University of Melbourne, Australia
Paolo Torroni	University of Bologna, Italy
Pınar Yolum	Bogazici University, Turkey

Lecture Notes in Artificial Intelligence (LNAI)

Vol. 4333: U. Reimer, D. Karagiannis (Eds.), Practical Aspects of Knowledge Management. XII, 338 pages. 2006.

Vol. 4327: M. Baldoni, U. Endriss (Eds.), Declarative Agent Languages and Technologies IV. VIII, 257 pages. 2006.

Vol. 4304: A. Sattar, B.-h. Kang (Eds.), AI 2006: Advances in Artificial Intelligence. XXVII, 1303 pages. 2006.

Vol. 4303: A. Hoffmann, B.-h. Kang, D. Richards, S. Tsumoto (Eds.), Advances in Knowledge Acquisition and Management. XI, 259 pages. 2006.

Vol. 4293: A. Gelbukh, C.A. Reyes-Garcia (Eds.), MICAI 2006: Advances in Artificial Intelligence. XXVIII, 1232 pages. 2006.

Vol. 4289: M. Ackermann, B. Berendt, M. Grobelnik, A. Hotho, D. Mladenič, G. Semeraro, M. Spiliopoulou, G. Stumme, V. Svatek, M. van Someren (Eds.), Semantics, Web and Mining. X, 197 pages. 2006.

Vol. 4285: Y. Matsumoto, R. Sproat, K.-F. Wong, M. Zhang (Eds.), Computer Processing of Oriental Languages. XVII, 544 pages. 2006.

Vol. 4274: Q. Huo, B. Ma, E.-S. Chng, H. Li (Eds.), Chinese Spoken Language Processing. XXIV, 805 pages. 2006.

Vol. 4265: L. Todorovski, N. Lavrač, K.P. Jantke (Eds.), Discovery Science. XIV, 384 pages. 2006.

Vol. 4264: J.L. Balcázar, P.M. Long, F. Stephan (Eds.), Algorithmic Learning Theory. XIII, 393 pages. 2006.

Vol. 4259: S. Greco, Y. Hata, S. Hirano, M. Inuiguchi, S. Miyamoto, H.S. Nguyen, R. Słowiński (Eds.), Rough Sets and Current Trends in Computing. XXII, 951 pages. 2006.

Vol. 4253: B. Gabrys, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part III. XXXII, 1301 pages. 2006.

Vol. 4252: B. Gabrys, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part II. XXXIII, 1335 pages. 2006.

Vol. 4251: B. Gabrys, R.J. Howlett, L.C. Jain (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Part I. LXVI, 1297 pages. 2006.

Vol. 4248: S. Staab, V. Svátek (Eds.), Managing Knowledge in a World of Networks. XIV, 400 pages. 2006.

Vol. 4246: M. Hermann, A. Voronkov (Eds.), Logic for Programming, Artificial Intelligence, and Reasoning. XIII, 588 pages. 2006.

Vol. 4223: L. Wang, L. Jiao, G. Shi, X. Li, J. Liu (Eds.), Fuzzy Systems and Knowledge Discovery. XXVIII, 1335 pages. 2006.

Vol. 4213: J. Fürnkranz, T. Scheffer, M. Spiliopoulou (Eds.), Knowledge Discovery in Databases: PKDD 2006. XXII, 660 pages. 2006.

Vol. 4212: J. Fürnkranz, T. Scheffer, M. Spiliopoulou (Eds.), Machine Learning: ECML 2006. XXIII, 851 pages. 2006.

Vol. 4211: P. Vogt, Y. Sugita, E. Tuci, C.L. Nehaniv (Eds.), Symbol Grounding and Beyond. VIII, 237 pages. 2006.

Vol. 4203: F. Esposito, Z.W. Raś, D. Malerba, G. Semeraro (Eds.), Foundations of Intelligent Systems. XVIII, 767 pages. 2006.

Vol. 4201: Y. Sakakibara, S. Kobayashi, K. Sato, T. Nishino, E. Tomita (Eds.), Grammatical Inference: Algorithms and Applications. XII, 359 pages. 2006.

Vol. 4200: I.F.C. Smith (Ed.), Intelligent Computing in Engineering and Architecture. XIII, 692 pages. 2006.

Vol. 4198: O. Nasraoui, O. Zaïane, M. Spiliopoulou, B. Mobasher, B. Masand, P.S. Yu (Eds.), Advances in Web Mining and Web Usage Analysis. IX, 177 pages. 2006.

Vol. 4196: K. Fischer, I.J. Timm, E. André, N. Zhong (Eds.), Multiagent System Technologies. X, 185 pages. 2006.

Vol. 4188: P. Sojka, I. Kopeček, K. Pala (Eds.), Text, Speech and Dialogue. XV, 721 pages. 2006.

Vol. 4183: J. Euzenat, J. Domingue (Eds.), Artificial Intelligence: Methodology, Systems, and Applications. XIII, 291 pages. 2006.

Vol. 4180: M. Kohlhase, OMDoc – An Open Markup Format for Mathematical Documents [version 1.2]. XIX, 428 pages. 2006.

Vol. 4177: R. Marín, E. Onaindía, A. Bugarín, J. Santos (Eds.), Current Topics in Artificial Intelligence. XV, 482 pages. 2006.

Vol. 4160: M. Fisher, W. van der Hoek, B. Konev, A. Lisitsa (Eds.), Logics in Artificial Intelligence. XII, 516 pages. 2006.

Vol. 4155: O. Stock, M. Schaerf (Eds.), Reasoning, Action and Interaction in AI Theories and Systems. XVIII, 343 pages. 2006.

Vol. 4149: M. Klusch, M. Rovatsos, T.R. Payne (Eds.), Cooperative Information Agents X. XII, 477 pages. 2006.

Vol. 4140: J.S. Sichman, H. Coelho, S.O. Rezende (Eds.), Advances in Artificial Intelligence - IBERAMIA-SBIA 2006. XXIII, 635 pages. 2006.

Vol. 4139: T. Salakoski, F. Ginter, S. Pyysalo, T. Pahikkala (Eds.), Advances in Natural Language Processing. XVI, 771 pages. 2006.

Vol. 4133: J. Gratch, M. Young, R. Aylett, D. Ballin, P. Olivier (Eds.), Intelligent Virtual Agents. XIV, 472 pages. 2006.

Vol. 4130: U. Furbach, N. Shankar (Eds.), Automated Reasoning. XV, 680 pages. 2006.

Vol. 4120: J. Calmet, T. Ida, D. Wang (Eds.), Artificial Intelligence and Symbolic Computation. XIII, 269 pages. 2006.

Vol. 4118: Z. Despotovic, S. Joseph, C. Sartori (Eds.), Agents and Peer-to-Peer Computing. XIV, 173 pages. 2006.

Vol. 4114: D.-S. Huang, K. Li, G.W. Irwin (Eds.), Computational Intelligence, Part II. XXVII, 1337 pages. 2006.

Vol. 4108: J.M. Borwein, W.M. Farmer (Eds.), Mathematical Knowledge Management. VIII, 295 pages. 2006.

Vol. 4106: T.R. Roth-Berghofer, M.H. Göker, H.A. Güvenir (Eds.), Advances in Case-Based Reasoning. XIV, 566 pages. 2006.

Vol. 4099: Q. Yang, G. Webb (Eds.), PRICAI 2006: Trends in Artificial Intelligence. XXVIII, 1263 pages. 2006.

Vol. 4095: S. Nolfi, G. Baldassarre, R. Calabretta, J.C.T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, D. Parisi (Eds.), From Animals to Animats 9. XV, 869 pages. 2006.

Vol. 4093: X. Li, O.R. Zaïane, Z. Li (Eds.), Advanced Data Mining and Applications. XXI, 1110 pages. 2006.

Vol. 4092: J. Lang, F. Lin, J. Wang (Eds.), Knowledge Science, Engineering and Management. XV, 664 pages. 2006.

Vol. 4088: Z.-Z. Shi, R. Sadananda (Eds.), Agent Computing and Multi-Agent Systems. XVII, 827 pages. 2006.

Vol. 4087: F. Schwenker, S. Marinai (Eds.), Artificial Neural Networks in Pattern Recognition. IX, 299 pages. 2006.

Vol. 4068: H. Schärfe, P. Hitzler, P. Øhrstrøm (Eds.), Conceptual Structures: Inspiration and Application. XI, 455 pages. 2006.

Vol. 4065: P. Perner (Ed.), Advances in Data Mining. XI, 592 pages. 2006.

Vol. 4062: G.-Y. Wang, J.F. Peters, A. Skowron, Y. Yao (Eds.), Rough Sets and Knowledge Technology. XX, 810 pages. 2006.

Vol. 4049: S. Parsons, N. Maudet, P. Moraitis, I. Rahwan (Eds.), Argumentation in Multi-Agent Systems. XIV, 313 pages. 2006.

Vol. 4048: L. Goble, J.-J.C.. Meyer (Eds.), Deontic Logic and Artificial Normative Systems. X, 273 pages. 2006.

Vol. 4045: D. Barker-Plummer, R. Cox, N. Swoboda (Eds.), Diagrammatic Representation and Inference. XII, 301 pages. 2006.

Vol. 4031: M. Ali, R. Dapoigny (Eds.), Advances in Applied Artificial Intelligence. XXIII, 1353 pages. 2006.

Vol. 4029: L. Rutkowski, R. Tadeusiewicz, L.A. Zadeh, J.M. Zurada (Eds.), Artificial Intelligence and Soft Computing – ICAISC 2006. XXI, 1235 pages. 2006.

Vol. 4027: H.L. Larsen, G. Pasi, D. Ortiz-Arroyo, T. Andreasen, H. Christiansen (Eds.), Flexible Query Answering Systems. XVIII, 714 pages. 2006.

Vol. 4021: E. André, L. Dybkjær, W. Minker, H. Neumann, M. Weber (Eds.), Perception and Interactive Technologies. XI, 217 pages. 2006.

Vol. 4020: A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi (Eds.), RoboCup 2005: Robot Soccer World Cup IX. XVII, 727 pages. 2006.

Vol. 4013: L. Lamontagne, M. Marchand (Eds.), Advances in Artificial Intelligence. XIII, 564 pages. 2006.

Vol. 4012: T. Washio, A. Sakurai, K. Nakajima, H. Takeda, S. Tojo, M. Yokoo (Eds.), New Frontiers in Artificial Intelligence. XIII, 484 pages. 2006.

Vol. 4008: J.C. Augusto, C.D. Nugent (Eds.), Designing Smart Homes. XI, 183 pages. 2006.

Vol. 4005: G. Lugosi, H.U. Simon (Eds.), Learning Theory. XI, 656 pages. 2006.

Vol. 4002: A. Yli-Jyrä, L. Karttunen, J. Karhumäki (Eds.), Finite-State Methods and Natural Language Processing. XIV, 312 pages. 2006.

Vol. 3978: B. Hnich, M. Carlsson, F. Fages, F. Rossi (Eds.), Recent Advances in Constraints. VIII, 179 pages. 2006.

Vol. 3963: O. Dikenelli, M.-P. Gleizes, A. Ricci (Eds.), Engineering Societies in the Agents World VI. XII, 303 pages. 2006.

Vol. 3960: R. Vieira, P. Quaresma, M.d.G.V. Nunes, N.J. Mamede, C. Oliveira, M.C. Dias (Eds.), Computational Processing of the Portuguese Language. XII, 274 pages. 2006.

Vol. 3955: G. Antoniou, G. Potamias, C. Spyropoulos, D. Plexousakis (Eds.), Advances in Artificial Intelligence. XVII, 611 pages. 2006.

Vol. 3949: F.A. Savacı (Ed.), Artificial Intelligence and Neural Networks. IX, 227 pages. 2006.

Vol. 3946: T.R. Roth-Berghofer, S. Schulz, D.B. Leake (Eds.), Modeling and Retrieval of Context. XI, 149 pages. 2006.

Vol. 3944: J. Quiñonero-Candela, I. Dagan, B. Magnini, F. d'Alché-Buc (Eds.), Machine Learning Challenges. XIII, 462 pages. 2006.

Vol. 3937: H. La Poutré, N.M. Sadeh, S. Janson (Eds.), Agent-Mediated Electronic Commerce. X, 227 pages. 2006.

Vol. 3932: B. Mobasher, O. Nasraoui, B. Liu, B. Masand (Eds.), Advances in Web Mining and Web Usage Analysis. X, 189 pages. 2006.

Vol. 3930: D.S. Yeung, Z.-Q. Liu, X.-Z. Wang, H. Yan (Eds.), Advances in Machine Learning and Cybernetics. XXI, 1110 pages. 2006.

Vol. 3918: W.-K. Ng, M. Kitsuregawa, J. Li, K. Chang (Eds.), Advances in Knowledge Discovery and Data Mining. XXIV, 879 pages. 2006.

Vol. 3913: O. Boissier, J. Padget, V. Dignum, G. Lindemann, E. Matson, S. Ossowski, J.S. Sichman, J. Vázquez-Salceda (Eds.), Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems. XII, 259 pages. 2006.

Table of Contents

Invited Talk

Invited Papers

Contributed Papers

Producing Compliant Interactions: Conformance, Coverage, and Interoperability

Amit K. Chopra and Munindar P. Singh

North Carolina State University

Abstract. Agents in an open system interact with each other based on (typically, published) protocols. An agent may, however, deviate from the protocol because of its internal policies. Such deviations pose certain challenges: (1) the agent might no longer be conformant with the protocol—how do we determine if the agent is conformant? (2) the agent may no longer be able to interoperate with other agents—how do we determine if two agents are interoperable? (3) the agent may not be able to produce some protocol computations; in other words, it may not cover the protocol—how we determine if an agent covers a protocol?

We formalize the notions of conformance, coverage and interoperability. A distinctive feature of our formalization is that the three are orthogonal to each other. Conformance and coverage are based on the semantics of runs (a run being a sequence of states), whereas interoperability among agents is based upon the traditional idea of *blocking*. We present a number of examples to comprehensively illustrate the orthogonality of conformance, coverage, and interoperability.

Compliance is a property of an agent's execution whereas conformance is a property of the agent's design. In order to produce only compliant executions, first and foremost the agent must be conformant; second, it must also be able to interoperate with other agents.

1 Introduction

We investigate the topic of an agent's compliance with a protocol by checking its design for conformance with the protocol and interoperability with other agents. Our agents are set in an open environment, and thus expected to be autonomous and heterogeneous. The interactions of agents are characterized in terms of protocols. The autonomy of an agent is reflected in its policies, which affect how it interacts with others, possibly resulting in deviations from the given protocol.

Deviations complicate the task of determining compliance. To take a simple example, a customer in a purchase protocol may send reminders to a merchant at its own discretion even though the protocol did not encode sending reminders. Some deviations can be flagrant violations. For example, a customer may not pay after receiving the goods it ordered. What can we say about the compliance of these agents? Sending a reminder seems like an innocuous deviation from protocol, whereas not sending the payment appears more serious. One could argue that sending reminders could have been easily incorporated into the protocol. However, when we consider that deviations in protocol are a manifestation of the individual policies of agents, the number of possible deviations from a protocol is potentially infinite. As more deviations are encoded, the resulting

M. Baldoni and U. Endriss (Eds.): DALT 2006, LNAI 4327, pp. 1–15, 2006.

protocol would become large and unwieldy. If each deviant protocol were published as a separate protocol, too many niche protocols would arise. It is better to maintain a smaller number of general protocols and to entertain deviations from such protocols. However, not all deviations are acceptable from the point of view of compliance.

1.1 Compliance: Conformance and Interoperability

For an agent to be compliant with a protocol, first and foremost it must be conformant with the protocol. While agent compliance can only be checked by monitoring the messages the agent exchanges with its peers at runtime, conformance can be verified from its design. The design of an agent involves two primary components: protocols and policies. Protocols are the public part of the design and can be considered fixed for the set of agents that adopt specific roles in the protocol. However, the policies are private to each agent, and potentially unique to each agent. Hence, the design of an agent is a function of its policies. An agent is conformant with a protocol if it respects the semantics of the protocol. A useful criterion when considering conformance is the satisfaction of commitments. Our definition of conformance supports commitments, but it is more general.

The distinction between conformance and compliance is important: an agent's design may conform, but its behavior may not comply. This may be not only because of the agent's failure or unreliable messaging (which do not concern us here), but also because an agent's design may preclude successful interoperation with its peers. In other words, even though an agent is individually conformant, it may not be able to generate compliant computations because of the other agents with whom it interacts, apparently according to the same protocol. Interoperability is distinct from conformance; interoperability is strictly with respect to other agents, whereas conformance is with respect to a protocol.

1.2 Coverage

A protocol may offer a number of alternative execution paths. Some of those paths may be impossible for an agent who deviates from the protocol. Such a reduction in possible paths may be viewed as a reduction in the capabilities of an agent. Conversely, the agent's design may make it possible to interact along paths unforeseen in the protocol. Such an addition may be viewed as an increase in the capabilities of an agent. Informally, we say an agent covers a protocol if it capable of taking any of the paths in the protocol.

This notion of coverage is an important one: if an agent covers a protocol it would appear to be at least as flexible as the protocol. That is, the agent can handle whatever the protocol can "throw" at it. Moreover, in some settings it may be institutionally required that an agent cover a protocol. For example, a tax official must report discrepancies in reviewed filings to the main office; the official cannot ignore them.

1.3 Contributions and Organization

Our contributions include (1) an account of conformance and coverage based on a semantics for protocols suitable for open systems; (2) showing how conformance, coverage, and interoperability are orthogonal concerns; and (3) establishing that in order to only produce compliant interactions, one has to consider both an agent's conformance with the protocol, and its interoperability with other agents.

Section 2 presents the representation of protocols as transition systems. Section 3 discusses the way in which an agent may deviate from protocol. Section 4 defines conformance and coverage. Section 5 discusses the interoperability of agents. Section 6 shows that conformance, coverage, and interoperability are orthogonal; it also discusses the relevant literature.

2 Protocols

We represent protocols as transition systems; the transition systems are similar to those described by $C+$ specifications [5]. The *signature* of a transition system is the set σ of constants that occur in it. Here σ^{act} and σ^{fl} represent the sets of actions and fluents, respectively. Each constant c is assigned a nonempty finite domain *Dom(c)* of symbols. An *interpretation* of σ is an assignment $c = v$ for each $c \in \sigma$ where $v \in Dom(c)$.

Informally, a transition system is a graph with states as vertices and actions as edges. A state s is a particular interpretation of σ^{fl}, the set of fluents; a transition is a triple $\langle s, e, s' \rangle$ where s and s' are states, and e is an interpretation of σ^{act}, the set of actions. In addition, the initial and final states are marked.

Definition 1. A transition system is a $\langle \sigma^{fl}, \sigma^{act}, S, s_0, F, \delta \rangle$, where σ^{fl} is the set of fluents, σ^{act} is the set of actions, S is the set of states such that $S \subseteq 2^{\sigma^{fl}}$, $s_0 \in S$ is an initial state, $F \subseteq S$ is the set of final states, $\delta \subseteq S \times E \times S$ is the set of transitions, where $E \subseteq 2^{\sigma^{act}}$.

Figure 1 shows the transition system of a purchase protocol. The protocol has two roles: *merchant* (*mer*) and *customer* (*cus*) engaging in the steps below:

1. The customer sends a *request* for quotes to the merchant.
2. The merchant responds either by sending an *offer* for the goods for which the customer requested a quote, or by indicating the nonavailability of requested goods in which case the protocol ends. By sending an offer, the merchant creates the conditional commitment *CC(mer, cus, a_price, an_item)* meaning that if the customer pays price *a_price*, then the merchant will send the goods *an_item*.
3. The customer can respond to the offer by either sending an *accept*, or a *reject*. Accepting the quote creates a conditional commitment *CC(cus, mer, an_item, a_price)*, meaning that if the merchant sends the goods, then the customer will pay. If the customer sends a *reject*, the protocol ends.
4. If the customer sends a *payment* to the merchant, then *CC(cus, mer, an_item, a_price)* is discharged and *CC(mer, cus, a_price, an_item)* is reduced to *C(mer, cus, an_item)* meaning that the merchant is now committed to sending the goods. But if the merchant sends *an_item* to the customer, then *CC(mer, cus, a_price,an_item)* is discharged and *CC(cus, mer, an_item, a_price)* is reduced to *C(cus, mer, a_price)* meaning that the customer is now committed to paying for the goods.
5. If the customer has paid in the previous step, then the merchant sends the goods, thereby discharging its commitment. But if the merchant has sent the goods in

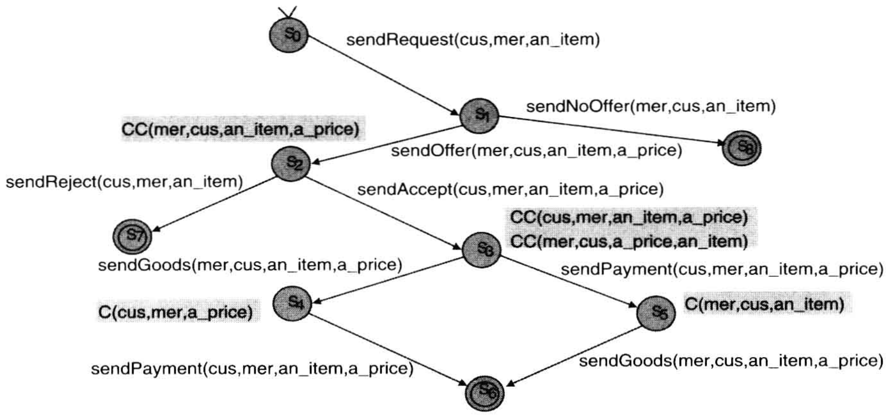

Fig. 1. A purchase protocol

the previous step, then the customer sends the payment, thereby discharging its commitment. In either case, no commitments or conditional commitments hold in the resulting state, which is a final state of the protocol.

Table 1 shows the interpretation of states in the transition system. An action starting with 'send' represents a single message exchange between roles with the *sender* role and *receiver* role as the first and second arguments, respectively. The fluents *initial* and *final* mark the start state and the final states respectively.

We now introduce some definitions related to transition systems.

Definition 2. A path in a transition system is a series of transitions $\langle s_0, e_0, s_1\rangle$, $\langle s_1, e_1, s_2\rangle$, ...,$\langle s_{f-1}, e_{f-1}, s_f\rangle$ such that s_0 is the initial state, and s_f is a final state.

A path may be abbreviated as $\langle s_0, e_0, s_1, e_1, \ldots, e_{f-1}, s_f\rangle$. Given a path $\rho = \langle s_0, e_0, s_1, \ldots, s_i, e_i, \ldots, e_{f-1}, s_f\rangle$, we say $e_i \in \rho$ $(0 \le i < f)$, and $s_i \in \rho$ $(0 \le i \le f)$.

We restrict our attention to two-party protocols. All the actions performed by the agents are communications. We further assume about the transition system of any protocol or agent that (1) only one action is performed along any transition; (2) in any transition $\langle s, e, s'\rangle$, $s \not\equiv s'$; (3) there exist no transitions $\langle s, e, s'\rangle$ and $\langle s, e', s'\rangle$ such that $e \equiv e'$ (in other words, no two distinct actions cause a transition into the same destination state from the same origin state); (4) the transition system is deterministic; and (5) along any path in the transition system, an action is performed at most once.

Definition 3. A *run* in a transition system is a series of states $\langle s_0, s_1, \ldots, s_f\rangle$ such that there exists a path $\langle s_0, e_0, s_1, e_1, \ldots, e_{f-1}, s_f\rangle$ in the transition system.

For example, the protocol of Figure 1 has the runs: $\langle s_0, s_1, s_8\rangle$, $\langle s_0, s_1, s_2, s_7\rangle$, $\langle s_0, s_1, s_2, s_3, s_4, s_6\rangle$, and $\langle s_0, s_1, s_2, s_3, s_5, s_6\rangle$. Note that given the above restrictions, each run maps to a unique path and vice versa.

Definition 4. The t-span $[T]$ of a transition system T is the set of paths in T.

Table 1. States in Figure 1

State	Fluents
s_0	*initial*
s_1	*request(cus, mer, an_item)*
s_2	*request(cus, mer, an_item)*, *offer(mer, cus, an_item, a_price)*, *CC(cus, mer, an_item, a_price)*
s_3	*request(cus, mer, an_item)*, *offer(mer, cus, an_item, a_price)*, *accept(cus, mer, an_item, a_price)*, *CC(cus, mer, an_item, a_price)*, *CC(mer, cus, a_price, an_item)*
s_4	*request(cus, mer, an_item)*, *offer(mer, cus, an_item, a_price)*, *accept(cus, mer, an_item, a_price)*, *goods(mer, cus, an_item, a_price)*, *C(cus, mer, a_price)*
s_5	*request(cus, mer, an_item)*, *offer(mer, cus, an_item, a_price)*, *accept(cus, mer, an_item, a_price)*, *pay(cus, mer, an_item, a_price)*, *C(mer, cus, an_item)*
s_6	*request(cus, mer, an_item)*, *offer(mer, cus, an_item, a_price)*, *accept(cus, mer, an_item, a_price)*, *goods(mer, cus, an_item, a_price)*, *pay(cus, mer, an_item, a_price)*, *final*
s_7	*request(cus, mer, an_item)*, *offer(mer, cus, an_item, a_price)*, *reject(cus, mer, an_item, a_price)*, *final*
s_8	*request(cus, mer, an_item)*, *no_offer(mer, cus, an_item)*, *final*

Notice that t-span is thus defined for protocols, role skeletons, and agents.

For example, $\{\langle s_0, s_1, s_2, s_7\rangle, \langle s_0, s_1, s_8\rangle, \langle s_0, s_1, s_2, s_3, s_4, s_6\rangle, \langle s_0, s_1, s_2, s_3, s_5, s_6\rangle\}$ is the t-span of the purchase protocol of Figure 1.

3 Deviating from Protocol

A role skeleton is a projection of a protocol onto a particular role; it is the transition system of the role. Figure 2 shows the customer skeleton. A customer's policies are combined with the customer role to create a new transition system representing the customer agent. Saying an agent is conformant with a protocol is the same as saying it is conformant with the role it adopts in the protocol; the same holds for coverage. Also note that if the transition system of an agent is identical to the skeleton of the role it adopts, we shall say that the agent *follows* the role.

The policies that go into designing an agent may be such that it follows a protocol. Or, they may be such that the agent encodes deviations from the protocol. Below, we list some common kinds of deviations.

Narrowing. The t-span of an agent is a proper subset of the t-span of the role skeleton it adopts: a typical reason for this would be to simplify its implementation.

Example 1. As shown in the agent's transition system in Figure 3, the customer requires the goods to arrive before it sends the payment. Essentially, the customer has removed a run from the role skeleton, namely, the run in which payment happens before the delivery of goods. ∎

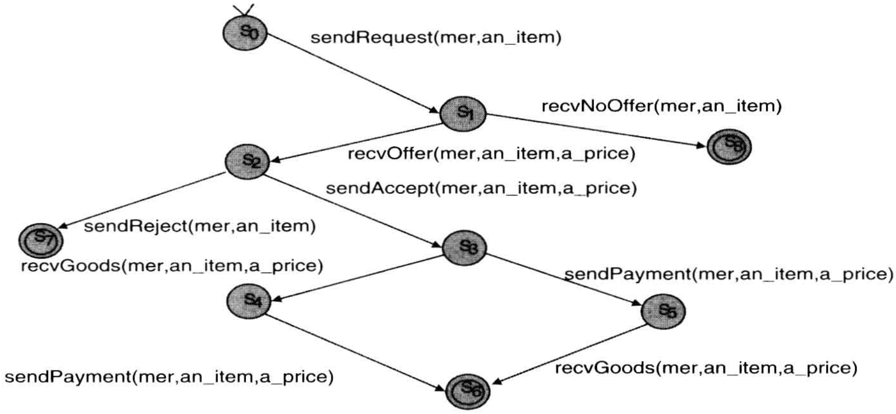

Fig. 2. Customer role skeleton

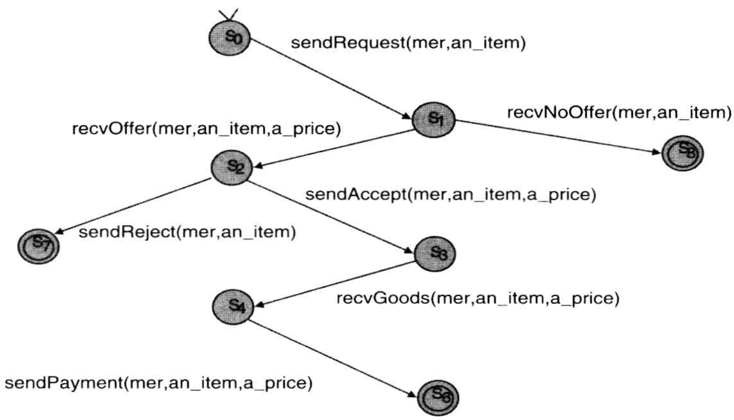

Fig. 3. Customer who sends payment only after receiving goods

Broadening. The t-span of the role skeleton is a proper subset of the t-span of the agent that adopts that role: a typical reason for this would be to handle scenarios not encoded in the protocol.

Example 2. The customer agent sends a reminder to the merchant about its commitment to send goods. Thus, in addition to the original runs, the customer agent includes the run in which it sends a reminder. For the sake of brevity, Figure 4 only shows the additional run; the remaining runs are as in Figure 2. ∎

Lengthening. The t-span of an agent is similar to that of the role skeleton except that some runs in the t-span of the agent are longer than the corresponding runs in the role skeleton: the reason is that additional actions happen along the path corresponding to the run.

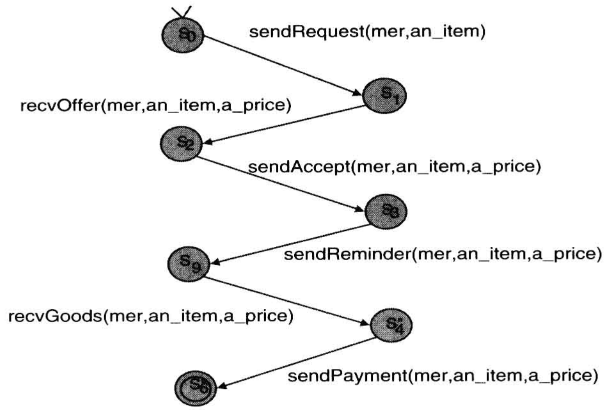

Fig. 4. The run in which customer sends a reminder

Example 3. If we replace the run $\langle s_0, s_1, s_2, s_3, s_4, s_6 \rangle$ in the customer role skeleton (shown in Figure 2) with the run in which a reminder is sent (shown in Figure 4), then it represents an example of lengthening. ▮

Example 4 illustrates the shortening of runs.

Example 4. Consider the customer of Figure 5. After receiving goods, the customer does not send payment for them. State s_4 is a final state for this customer. ▮

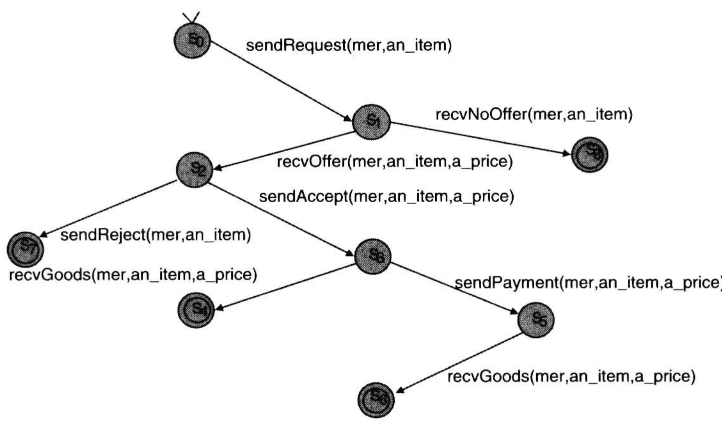

Fig. 5. Customer who does not pay for received goods

Gating. An agent may broaden or lengthen a protocol in such a way that it expects to receive additional messages from its partners in order to proceed.