Research Report 11/99

Commitment in Agent Cooperation, Applied to Agent-Based Simulation

by

Staffan Hägg

Department of
Software Engineering and Computer Science
University of Karlskrona/Ronneby
S-372 25 Ronneby
Sweden

ISSN 1103-1581
ISRN HK/R-RES—99/11—SE

Commitment in Agent Cooperation, Applied to Agent-Based Simulation

by Staffan Hägg

ISSN 1103-1581
ISRN HK/R-RES—99/11—SE

Printed by Psilander Grafiska, Karlskrona 1999

Commitment in Agent Cooperation, Applied to Agent-Based Simulation[1]

Staffan Hägg

Staffan.Hagg@ide.hk-r.se
http://www.sikt.hk-r.se/~staffanh

Department of Computer Science
University of Karlskrona/Ronneby, Sweden

Active Computing (R)
Ronneby, Sweden

Abstract

We regard simulation as a set of interaction episodes between partaking agents in an agent-based simulation. In order to structure the interaction we use commitments, and the semantics of such commitments is an analogy to a *two-phase commit* scheme used in distributed databases. A commitment is a contract between the involved agents to reach and preserve a specified goal, during a specified time. Here we define the semantics, describe the messages and outline administrative issues.

The full strength of the model is reached when an initiator needs to make commitments with a number of other agents. It then awaits the replies, and then, depending on the replies, it either confirms or rejects proposed commitments. Hereby it accomplishes an analogy to an *atomic action* when a two-phase commit scheme is used.

When the commitment scheme is applied to a simulation, a crucial part of the design is to model the requirements for accepting commitments during simulation. We introduce the concepts of an acceptance function and acceptance domains that are used to express these requirements. An example from battle field simulation is given.

With the proposed model simulated objects may quite easily be replaced by real objects. With the right kind of interface, real trucks and other entities may partake in real time operations, while other entities are simulated.

1. This research is sponsored by the University of Karlskrona/Ronneby, Sweden and Active Computing (R).

1 Introduction

The concept of *commitment* has been studied extensively in agent research. Following the path of Cohen and Levesque [1], commitment has been viewed as a mental phenomenon, (e.g. Singh [6]). More recently, with the emphasis on multi-agent systems, commitment is considered a social property, describing inter-agent relationships (Conte and Castelfranchi [2], Jennings [5], Singh [7]). In our approach, we also regard commitment as a social concept and a means to structure complex and dynamic team formation and interaction patterns in a multi-agent system. Here we take a starting point in a *two-phase commit* scheme, used in distributed databases, and especially the Commitment, Concurrency, and Recovery (CCR) scheme, known as ISO 9804/9805 (or CCITT X.861/871). The idea is to take a well-proven method for controlling inter-dependencies between sub-parts and to generalize it to be valid not only for database operations but for agents, persistently acting to reach and maintain a specified goal. It is also generalized in the way that commitment establishment is not made by request-reply but as a negotiation. In the Societies of Computation (SoC) research programme [8] [3], we are engaged in a number of activities concerning agent-based simulation, e.g. simulation of smart houses in an energy distribution project, and battle simulation with the company Mandator Teknik (R) and the Swedish Defence Department.

Agent-based simulation is a new area for application of agent technology. The basic idea is to model the simulation as an interaction between agents representing objects that are being simulated. As will be shown in the conclusions, it gives a number of advantages, whereas it does not draw any clear line between a simulation and a the use of an on-line system for control and planning as a "real", not simulated system.

In our proposal, a simulation is regarded as a set of interaction episodes, possibly overlapping both in time and in the agent space, and commitments can be used to structure these episodes and the inter-dependencies between agents partaking in them. The paper outlines our notion of commitment, and exemplifies its use in agent-based simulation. First, we define a commitment and outline the administration of commitments in sections 2 and 3. Then we turn to the simulation application in section 4, to draw conclusions in section 5.

2 Commitments

Agents and Goals

An agent A_i is uniquely identified in a global name space. (D1)

An agent A_i can hold goals G_{ij} that are locally uniquely identified and denotes some state of the world. Having a goal G_{ij} *i*) if G_{ij} is not reached, A_i will perform actions in order to reach G_{ij} and restrain from performing actions that would hinder reaching G_{ij}; *ii*) if G_{ij} is reached, A_i will perform actions in order to maintain G_{ij} and restrain from performing actions that would make G_{ij} not true. (D2)

A goal G_{ij} is on the form[1] i*d(parameter*$_1$*, parameter*$_2$*, ... , parameter*$_n$*)*, where *id* is a character string; *parameter*$_i$ is a character string, an integer, or a float number; and *i* >= 0. (D3)

Our cooperation model does not prescribe how agents should be constructed internally. However, agents are required to support the following functions/operations:

1. Internally, agents may have other representations of goals.

- *Communication*: Agents should support message passing, including the primitives for handling commitments.
- *Goals*: Agents should support the setting and maintenance of persistent goals.
- *Commitment management*: Agents should support management of commitments, as described below.
- *Multi-threaded*: Agents should be multi-threaded or otherwise support asynchronous handling of multiple commitments along with its normal operation.

Commitments

A commitment C_{Aij} is a tuple $<A_i, j, A_k, G_{kl}, t>$, where A_i is the *initiator* (owner, creditor) of the commitment, j is an A_i locally unique identifier, A_k is the *responder* (holder, debtor) of the commitment, G_{kl} is a goal that A_k associates with the commitment (below), and t is the duration time (also below). (D4)

From (D1) and (D4) it follows that the commitment C_{Aij} is uniquely identified in a global name space. It is on the form *agentname*:*j*, where *agentname* is the name of the commitment initiator, and *j* is the locally unique identifier. (D5)

A commitment C_{Aij} has a duration time $C_{Aij}(t)$. When t is reached, the commitment responder, A_k, releases C_{Aij}, and the commitment initiator, A_i, unregisters C_{Aij}. $C_{Aij}(t)$ can be given on two forms, absolute time and relative time. (D6)

An agent A_k that makes a commitment C_{Aij} sets a goal G_{kl} and maps G_{kl} onto C_{Aij}. When C_{Aij} is released, A_k retracts G_{kl}. (D7)

Messages

The following messages are defined for handling commitments. Each of them has a message type included in the message.

Propose = (Recipient(s), PROPOSE, C_{Aij}, goal, $C_{Aij}(t)$)
Reply = (Recipient, REPLY, C_{Aij}, goal, $C_{Aij}(t)$)
Reject = (Recipient(s), REJECT, C_{Aij})
Confirm = (Recipient(s), CONFIRM, C_{Aij}, goal, $C_{Aij}(t)$)
Release = (Recipient(s), RELEASE, C_{Aij})
Query = (Recipient(s), QUERY, C_{Aij})
Answer = (Recipient, ANSWER, C_{Aij}, goal, $C_{Aij}(t)$)

“Recipient” denotes a single recipient of the message. “Recipient(s)” says that there may be more than one recipient (using broadcast/multicast, e.g. in [3]). When a message comes in, the receiver can detect which agent sent the message (assumed to be supported by the system).

3 Commitment Administration

Commitment establishment

A commitment always has one agent as its *initiator*. This agent sends a *Propose* message to one or more agents (*responders*) with a proposal to make a commitment. The proposal is presented with its identifier, a goal, and a duration time.

A receiving agent investigates whether it can accept the proposal. It can come to one of three decisions:

(1) The proposal can be accepted.
(2) The proposal can be accepted with modifications.
(3) The proposal can not be accepted.

In (1) the agent sends a *Reply* with the id, goal, and time fields identical to the corresponding fields in the Propose message. It records that there is a commitment proposal and reply outstanding. It also sets the goal, but temporarily, during time t_0. This means that it will restrain from making commitments that would be inconsistent with the current one, but no actions are made in order to reach the goal. The time t_0 is not modifiable by agents, but it can be set for an application. At t_0 the temporary goal is retracted and the record entry for the outstanding commitment propose/reply is erased.

In (2) the agent sends a *Reply* with the id field as in the Propose message, but with either or both of the goal and time fields modified. Hereby, the agent gives a new proposal to the initiator, turning the propose-reply scheme into a negotiation. Records are kept as before, and the t_0 timeout is set on the goal that is replied. Upon receiving the Reply message, the initiator may send another Propose message with the same id, and the procedure is restarted.

In (3) the agent sends a *Reject* message with the id from the Propose message.

The initiating agent may have one or more Propose messages outstanding with the same id. When replies have come in and there is an agreement between the agents, the initiator sends a *Confirm* message to the involved responders and updates its records. The message may or may not be identical for all agents, depending on whether the agreement is symmetric or asymmetric. On the receival of a confirm message, an agent turns the corresponding temporary goal into a proper one, and its record is updated.

If the initiator decides that an agreement should not be confirmed (an agreement has not been reached with all agents), it sends out a *Reject* message to the involved responders. On the receival of a Reject message, a responder retracts the corresponding temporary goal and the record entry for the outstanding commitment propose/reply is erased.

Figure 1 shows some scenarios for commitment establishment. In (a) Propose-Reply-Confirm messages show either that the responder accepted the initiator's proposal or that the initiator accepted the responder's modified proposal. In (b) the responder had a modified proposal that apparently was not accepted by the initiator. Then, either the responder accepted the initiator's second proposal or the initiator accepted the responder's second modified proposal. In (c) the responder did not accept the first proposal, and it could not come up with any modifications. In (d) the initiator did not accept the modified proposal from the responder, and it did not continue with a second proposal.

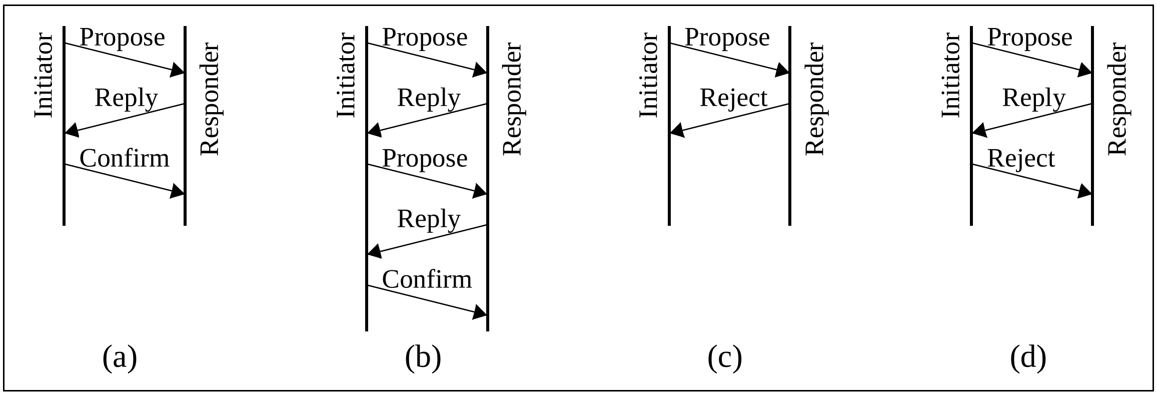

Figure 1. Scenarios for establishing a commitment.

Commitment release

There are three ways of releasing a commitment:

(1) The duration time for the commitment elapses.
(2) The initiator releases the commitment.
(3) The responder releases the commitment.

In normal operation the duration time for a commitment will be reached (1). An agent then retracts the corresponding goal and the record entry for the commitment is erased. The initiator also retracts its corresponding record entry. No messages are sent.

The initiator may release the commitment with responders (2). It then sends a *Release* message with the commitment id to concerned responders and updates its records accordingly. A receiving agent then retracts the corresponding goal and erases the record entry for the commitment.

A responder may also release a commitment (3). The procedure is similar; it sends a *Release* message with the commitment id to the initiator. It retracts the corresponding goal and erases the record entry for the commitment. The initiator updates its records, and it is now up to this agent to decide whether the release of a commitment with one agent affects the rest of its commitments. If so, it may, as a consequence, release commitments with one or more agents.

Commitment supervision

There are two message types reserved for supervision of commitments. The initiator of a commitment may send a *Query* message to responders included in a commitment. Upon the receival of a Query message, the responder checks its tables and replies with an *Answer* message with the corresponding goal and time fields filled in. If no such commitment should exist, these fields are left empty.

Though, the Query-Answer scheme is not reserved for initiator supervision. It may be used by any agent. Hereby, agents can make queries for statistical or other purposes, and therefore the id field in a Query message can be used differently. The id C_{Aij} is on the form *agentname*:*id* which is globally unique (see (D4) above). The query (007, QUERY, M:*) will make agent 007 send an Answer message for every commitment it has with agent M as initiator. The query (007, QUERY, *) will make agent 007 send an Answer message for every commitment it has.

Fault handling

Though we have assumed reliable communication, there may be fault situations that must be taken care of. One type of fault is when an agent crashes (or is retracted by human intervention) during commitment establishment. On the responder side we have solved this with the timer t_0 (above). Similarly, we take care of the situation with responding agents that do not reply, using timers within the initiator. But timers do not only solve problems; they also introduce new fault situations.

In Figure 2 (a) the responder's timer goes off, supposedly solving the problem with a crashed initiator. However, the initiator was merely slow, and the Confirm message arrives too late. As the initiator now believes that there is a commitment established, the responder resolves the fault situation by sending an *Reject* message with the commitment id fetched

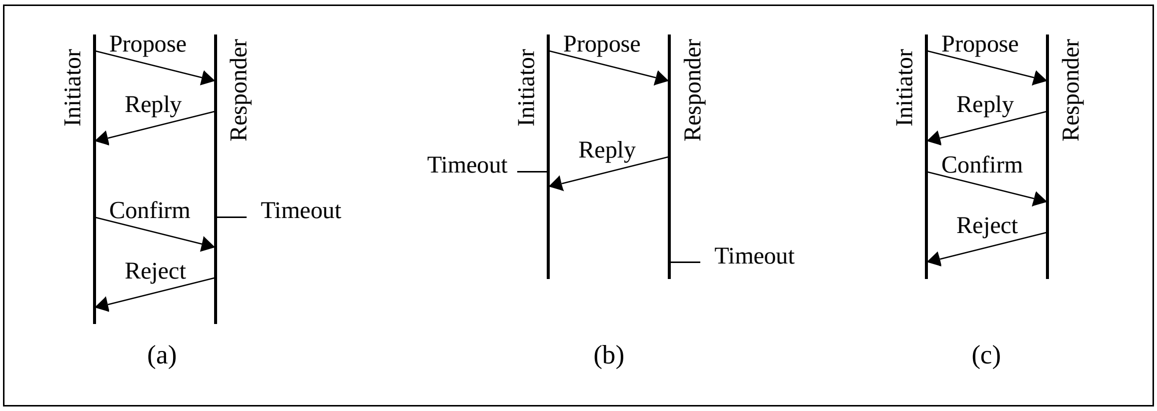

Figure 2. Three fault situations.

from the Confirm message. It is now up to the initiator to decide on how to proceed, but at least it knows that there is no commitment with this responder. In (b) the responder is slow, and its reply comes in too late. Here, the problem is not introduced by a timer. Instead, the timers solve the problem without any messages being sent; the late message is simply discarded. If the initiator wants to repeat its proposal, it may do so. In (c) we see a situation where everything, including the Confirm message, seems to be all right. However, as there is no acknowledgement of a Confirm message and a commitment is considered to be established on the initiator's decision only, the responder needs a way of responding to a misbehaving initiator. By sending a Reject message in response to a Confirm message, the responder says that this was not what we agreed upon.

For a full description of these and other issues, we refer to the longer technical report [4].

4 Agent-Base Simulation Using Commitments

As mentioned, we regard a simulation as a set of interaction episodes, possibly overlapping both in time and in the agent space, and commitments can be used to structure these episodes and the inter-dependencies between agents partaking in them. The example in Figure 3 shows an agent, A_{10}, during a period in time. Some of the commitments are overlapping in time, two of them have the same initiator, and one commitment proposal (the second) was rejected.

Crucial now is the *acceptance function* that determines whether a commitment proposal should be accepted, rejected, or if a modified proposal should be returned. This function is an essential part of the application design to define acceptance functions for agents.

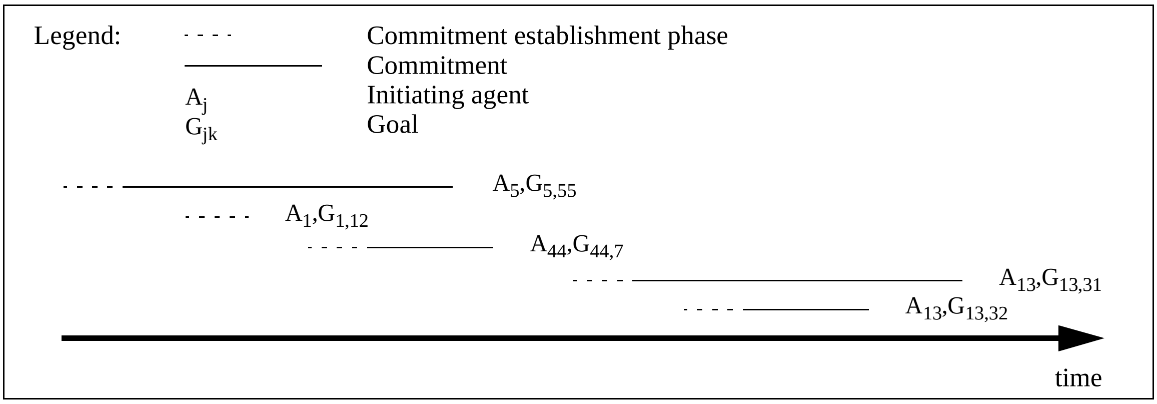

Figure 3. Example: commitments for an agent during a period of time.

Assignment domain: determines if the requested task complies with task assignment strategies.
Location domain: determines if the truck can be at a requested location at a specific time. This may lead to rerouting of planned paths, as long as existing commitments are not violated.
Cargo domain: determines if the truck can be loaded with a requested cargo.
Equipment domain: determines if the truck is or can be equipped to do a requested task at a specific time.
Armament domain: determines if the truck is or can be armed to do a requested task at a specific time.

Figure 4. Commitment domains for an army truck.

The acceptance function

The acceptance function takes a commitment proposal as input and outputs an acceptance, a rejection, or a modified commitment proposal. It takes into account the state of the world, as is recorded by the agent, including what other commitments the agent has made. We have chosen to define commitment domains regarding the nature of the application. The example in Figure 4 shows the commitment domains of an army truck, partaking in a combat simulation.

With this domain information and the commitment mechanism, officers interacting with the simulator can coordinate the activities of simulated entities who, in turn, can recursively make commitments with other entities. For example, if a truck has made a commitment to carry a certain cargo to a field warehouse at a specific time, it may or may not accept a new commitment proposal to carry another load, depending on the domain information, the state of the world, and details of the commitment. A scenario is shown in Figure 5. After the events described in the figure, it is up to the initiator (HQ) to decide on how to continue. If the modified proposal is accepted, there will be a Confirm message sent to Truck105, confirming the modified proposal.

5 Conclusions

Our model is a generalization of the two-phase commit scheme. It can be used not only for database operations, but for applications where agents should be persistent in maintaining a

(1) Agent Truck105 receives a Commitment Proposal:

```
(Truck105, PROPOSE, HQ:1234,
     transport(diesel, 20000, Warehouse1, Tank08), 01/01/1998:12:00)
```

(2) The proposal is checked against the Assignment domain: it is accepted.
(3) The proposal is checked against the Location domain: it is accepted.
(4) The proposal is checked against the Cargo domain: it is not accepted, but it can take 12000 litres, a smaller amount than requested.
(5) The proposal is checked against the Equipment domain: it is accepted.
(6) The proposal is checked against the Armament domain: it is accepted.
(7) Agent Truck105 sets a temporary goal to be timed out after a pre-defined time.
(8) Agent Truck105 responds with a Commitment Reply:

```
(HQ, REPLY, HQ:1234,
     transport(diesel, 12000, Warehouse1, Tank08), 01/01/1998:12:00)
```

Figure 5. Operations performed by the receiver of a Commitment Proposal.

certain state, moving in a specific direction, etc., and the duration is over a specified time. Furthermore, the establishment phase is not a request-reply scheme, but a negotiation. The *logics* of the negotiation is provided by the application specific acceptance function, while the *mechanism* for negotiation is provided by the cooperation model. A major advantage with the model (as with the two-phase commit scheme) is to secure the correct handling of multiple requests. It provides a standardized way of handling interaction situations, as exemplified with the army truck. The full strength of the model is reached when an initiator needs to make commitments with a number of other agents. It then awaits the replies, and then, depending on the replies, it either confirms or rejects proposed commitments. Hereby it accomplishes an analogy to an *atomic action* when a two-phase commit scheme is used.

Another advantage with the proposed model is that simulated objects quite easily may be replaced by real objects. With the right kind of interface, real trucks and other entities may partake in real time operations, while other entities are simulated. The interface may be a mobile terminal, and the acceptance function can have a desired level of automation, from the manual reading of maps, to a local control system that monitors the status, localization, etc. of the entity.

The argument can be taken even further. With real entities (including human operators) interacting with the use of the agent system, this can function as an administration system for resource handling, event reporting, and operation planning and control. The entities are real (but some may be simulated), and the line between a simulator and an operational system is thus no longer clear.

Last, using a system with multiple agents as described here, allows us to design a simulator as a heavily distributed application. Partaking agents (including interfacing real entities) may reside at any locations, relying on conventional communication methods. The cooperation model keeps track of inter-dependencies over time.

6 Future Work

The work reported here is currently used for ongoing work on building a platform for doing agent-based simulations. The example is taken from an application area that uses simulations with high demands on interactivity, flexibility, an security, and future work aim at extending the cooperation model to include issues on uncertainty, fault handling, and security.

References

[1] Cohen, P. R., and Levesque, H. J., Persistence, Intention and Commitment, in Cohen, P. R., Morgan, J., and Pollack, M. (eds.), *Intentions in Communication*, MIT Press, Cambridge, Massachusetts, USA.

[2] Conte, R., and Castelfranchi, C., *Cognitive and Social Action*, Chapter 6, UCL Press, London, UK, 1995, ISBN 1-85728-186-1.

[3] Hägg S., and Ygge F., *Agent-Oriented Programming in Power Distribution Automation - An Architecture, a Language, and their Applicability,* Ph.L. Thesis, LUNFD6/(NFCS-3094)/1-180/(1995), Lund University, Sweden, May 1995.

[4] Hägg S., *Commitment in Agent-Based Simulation*, University of Karlskrona/Ronneby, Research Report xx/98, ISSN 1103-1581 (under preparation).

[5] Jennings, N. R., *Cooperation in Industrial Multi-Agent Systems*, World Scientific Publishing, Singapore, 1994, ISBN 981-02-1652-1.

[6] Singh, M. P., Intentions, Commitments and Rationality, in *Proceedings of the Annual Conference of the Cognitive Science Society (CogSci)*, 1991.

[7] Singh, M. P., *An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts*, unpublished paper, available from Munindar P. Singh, singh@ncsu.edu, or http://www4.ncsu.edu/eos/info/dblab/www/mpsingh/papers/mas/comm-ont.ps.

[8] Societies of Computation (SoC), internet home page, http://www.sikt.hk-r.se/soc/.