Conflict Management with an Authority/ Deniability Model

Mihai Barbuceanu and Mark S. Fox
Enterprise Integration Laboratory
Department of Industrial Engineering
University of Toronto
4 Taddle Creek Road, Rosebrugh Building, Toronto, Ontario, M5S 1A4
email: {mihai, msf}@ie.utoronto.ca, tel: 1-416-978-6823 or 1-416-978-0910,
fax:1-416-978-3453

Abstract: When a reasoning system encounters a contradiction like p&q=>false, it may try to eliminate it by retracting some belief that supports either p or q. This paper addresses the issue of how to determine which of the possible supporting beliefs to retract. The problem is studied in the context of a heterogenous distributed environment in which we can not make assumptions about how agents derive their beliefs. The presented model considers two properties of beliefs: the authority of the agents that provided the belief in the first place and the costs incurred upon other agents that already used the belief (directly or indirectly) for their own decision making or action, if the belief is to be retracted. The model is implemented in an information infrastructure we have created for enterprise integration.

Keywords: cooperative information systems, enterprise integration.

1.0 Introduction

When a reasoning system encounters a contradiction like *p&q=>false*, it may try to eliminate it by retracting some current belief that supports either *p* or *q*. This paper addresses the issue of how to determine which of the possible supporting beliefs to retract. The problem is studied in the context of a heterogenous distributed environment in which we can not make any assumptions about how agents derive their beliefs. The model considers two properties of beliefs: the *authority* of the agents that provided the belief in the first place and the *costs* incurred upon other agents that used the belief for their own decision making or action, if the belief is to be retracted. The model is implemented in an agent-based information infrastructure we have created for enterprise integration. We consider two main advantages of this model. First, it provides a general method for belief revision that is accurate, as it depends on the current views of all involved agents, rather than on a pre-defined scheme. Second, it minimizes the communication and negotiation overhead by identifying situations in which negotiation can be avoided or reduced.

The paper presents first the agent-based information architecture that forms the context of our work. Then, it discusses the proposed authority/deniability model and reviews a number of issues that occured in the implementation phase.

2.0 An information infrastructure for enterprise integration

Our research approaches the construction of heterogenous collaborative environments for enterprise integration [Pan and Tenenbaum 91, Roboam and Fox 92] by relying on an infrastructure consisting of a class of agents called *information agents (IAs)* [Barbuceanu and Fox, 1994]. IAs are sophisticated knowledge and data management systems that allow other (functional) agents from the computerized organization to be *consistently and selectively aware* of relevant information by providing communication and information services supporting:

- Persistent storage of information to be shared among the multiple functional agents in the computerized organization.
- Deductive capabilities allowing new information to be inferred from existing information.
- Automatic, content-based routing and distribution of information to the agents that need it.
- Automatic retrieval, processing and integration of information that is relevant to agents.
- Checking and maintaining various forms of consistency of the information. We address terminological consistency, assertional consistency, and specific to this approach, temporal consistency.

A given IA will service a number of agents (both functional and other IA-s) by providing them with a layer of shared information storage and the above mentioned services for managing it. Functional agents maintain their own local information store. Agents periodically volunteer some of their information to the IA (and keep it up to date) or just answer the queries sent to them by the IA.

The IA uses its own knowledge together with the supplied information to determine which information needs of other agents can be satisfied. It processes the information in order to determine the most relevant content and the most appropriate form for the needs of these agents. In the process, it may uncover various forms of inconsistency among the supplied information and take action to remove them. *Terminological consistency* refers to the coherence of the conceptual vocabulary employed by agents and is handled by the T-Box services of the underlying description logic used by the IA. *Assertional consistency* refers to the coherence of the infornation exchanged by agents and is handled by the mechanisms presented in this paper. Finally, we have added *temporal consistency* as the coherence of exchanged information wrt to time intervals during which beliefs are held. The approach presented here handles this as well, as will be shown later on.

Essentially, the IA is composed of two components: a knowledge and data management system and an agent program, as shown in figure 1.

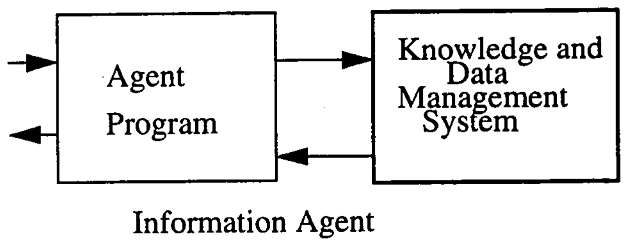

FIGURE 1. Architecture of the Information Agent.

The Knowledge and Data Management System provides knowledge representation and processing services based on a description logic language. They support: (i) a general inferential capability, in order to cope with the requirements for deductive query processing and content-based information routing, (ii) mechanisms for checking and enforcing the consistency of information, (iii) mechanisms for change management, in order to cope with the unexpected and unforseeable nature of the events happening in real corporate networks.

The Agent Program provides an interface to the external environment, allowing external messages to reach the IA and the IA to send out its own messages [Shoham 93]. The Agent Program relies on the KQML [Finin et. al. 92] agent communication language as the protocol for information exchange and translates from this protocol into the language of the knowledge management system.

For the knowledge management system we use a description logic language [Barbuceanu 93] that provides the usual concept-forming operators - *conjunction, value restrictions, number restrictions* - roles and subroles, disjointness declarations, primitive and defined concept specifications. The language T-Box provides the usual services of constructing the *complete form* of concepts and *automated classification* based on *subsumption checking* [Borgida et al. 89, Brachman and Schmolze 85, McGregor and

Bates, 87, etc.] The language A-Box is essentially a *constraint propagation* engine that makes instances conform to the various constraints asserted about them. It uses a propositional representation of instances and roles.

Unlike usual A-Boxes, ours is also a full *time-map management system* [Dean and McDermott, 87] that (i) records the *time intervals* during which propositions are true or false [Allen,1983], (ii) provides a *query language* that supports temporal interrogations, (iii) provides for the definition and application of *rules that perform forward reasoning* extending the information in the time mapped data base and (iiii) provides a *temporally extended* boolean truth maintenance system that records dependencies and supports asserting and retracting time-mapped propositions.

3.0 Conflict management with the authority/deniability model

3.1 Context and terminology

Assume we have a collection of functional agents serviced by an information agent in an enterprise environment. The functional agents may carry out enterprise functions such as marketing, design, manufacturing, purchasing, etc. The information agent provides the information and communication services supporting their consistent interaction. Agents communicate by sending and receiving messages consisting of *propositions*. Propositions are formed with concepts from a *common ontology* shared by all agents. The IA represents this ontology as a *concept* and *role* taxonomy encoded in a description logic. No assumption is made about how functional agents represent it. If `e1` is an automobile engine used by the enterprise, then the proposition `(v6engine e1 (at 13 march 94))` expresses the fact that `e1` is believed to be a v6 engine at 13 march 94, while `(l4engine e1(at 14 march 94))` expresses the fact that `e1` is belived to be an l4 engine at 14 march 94. Properties of objects are described by propositions such as `(power e1 132 (at 13 march 94))` - the power of engine e1 is believed to be 132 at the given time - or `(torque e1 150(at 13 march 94))` - the torque of e1 is believed to be 150 at the specified time. For the IA, `v6engine` and `l4engine` are concepts, while `power` and `torque` are roles. The IA maintains two kinds of propositions. *Premises* are propositions sent to the IA by other agents that consider them true. The IA has no access to whatever justification the sending agent may have for the proposition. *Derived propositions* (or simply propositions) are propositions inferred by the IA based on the available premises and on the IA's knowledge of the domain. For example, being told that `(v6engine e1(at 13 march 94))`, the IA may derive `(heavy e1(at 13 march 94))` based on domain knowledge. Agents that supplied propositions to the IA are named *producers* of the information. Agents that have been supplied information by the IA are named *consumers* of the information.

The operation of the IA can be described as consisting of functional agents sending information to and posting their information interests with the IA and the IA trying to answer these interests by integrating new information as it arrives from agents. An information interest (or topic of interest) is a persistent query whose answers are important to an agent. For example, assuming that engines are puchased from other companies, the purchasing agent may post as its interest the query `engine`, signaling that it must be informed of any entity representing an engine. In this case, propositions like `(v6engine e1(at 13 march 94))` and `(l4engine e1(at 14 march 94))` are amongst the information the purchasing agent will receive from the IA in response to its posted interest.

3.2 Authority and deniability

When information is integrated from multiple sources, contradictions can easily occur. For example, the marketing agent may have determined that for a new automotive product a v6 engine would sell better. Hence marketing will sent the IA a message telling that the engine should be a v6: `(v6engine e1(starting 13 march 94))`. From different requirements, design may later determine that only a l4 engine can be used: `(l4engine e1 (starting 14 march 94))`. Using domain knowledge that the `v6engine` and `l4engine` concepts are disjoint, the IA will derive a contradiction (for the common time interval during which both beliefs are held):

```
(and
   (v6engine e1(starting 13 march 94))
   (l4engine e1(starting 14 march 94))
   => false.
```

The conflict management service of the IA will try to remove this contradiction by considering two properities of information, authority and deniability. *Authority* is defined as a kind of priority agents may have wrt the truth of the information they deliver. If marketing has established a clear trend in the market favoring v6 engines, then it may deliver this information to the IA specifying an increased authority, e.g.: `((v6engine e1(starting 13 march 94)) :authority 9)`. Our model assumes that agents will honestly assess their authority for each piece of information they supply.

After information is delivered to the interested agents, these consumers will use it to make decisions and take action. For example, if marketing was first to determine that the engine must be a v6, the IA sent this information to the purchasing agent (whose interest was matched by it). The purchasing agent used the information to order v6 engines from another company. Later, design discovered that the engine must be a l4. If the design view is accepted, purchasing will have troubles in cancelling the order (paying penalties, etc.). This shows that information that has been consumed may be costly to retract later. We define the *deniability* of consumed information as a measure of the cost to retract it. We often use *undeniability* as the inverse of deniability. Undeniability (or deniability) is determined by the consumers of information. The same assumption about honest assessment of undeniability is made.

In conclusion, when a contradiction *p&q=> false* is encountered, we need to retract either *p* or *q* in order to remove the contradiction. The decision of what to retract considers the authority of the producers of the *premises* from which *p* and *q* were inferred as well as the cost incurred upon the other agents that have consumed any propositions derived from the premise to be retracted. To ensure accuracy, we assume that producers honestly assess authority and consumers honestly assess deniability.

3.3 The a-u space

The conflict resolution process has the goal of retracting one or several *premises* so that the contradiction can not be derived any more. When a premise is considered for retraction, both the authority of the agent producing it and the deniability of the propositions that are supported by the considered premise (and hence will be retracted with the considered premise) are considered. Suppose we have determined a set $\{p_i\}$ of premises which supports a *p&q=>false* contradiction. To each p_i we can attach an *authority measure* - the authority of its producer - and an *undeniability measure* - derived from the sum of deniability costs of all propositions that would have to be retracted if p_i is retracted. A high authority means that the proposition is more difficult to retract since a high authority has to be contradicted. A high undeniability means that the proposition is more difficult to retract because the costs of retraction incurred upon consumer agents will be great.

We can represent these two values in a diagram having authority on the *x*-axis and undeniability on the *y*-axis. Such a diagram is called an *a-u space* and is illustrated in fig. 2.

Propositions from the a-u space that have both low authority and low undeniability are easy to retract because no significant authority is violated and no significant costs are incurred. Propositions that have high authority and high undeniability are hard to retract exactly for the opposite reasons. An aggregated measure of both authority *and* undeniability is the distance *r* to the origin. If a proposition with high autority or undeniability is considered for retraction (e.g. because low authority and undeniability propositions do not exist), the IA must negotiate retraction with the producer and/or consumers. If authority is high, the producer must approve the retraction. If undeniability is high, consumers must be pooled and a measure of their approval must be computed. We can represent regions in the a-u space for the classes of propositions that can be retracted with or without negotiation. To do this, we introduce a threshold value of authority, a_t, and a threshold value of undeniability, u_t, such that propositions having authority and respectively undeniability higher than the threshold value can be retracted only after negotiation. Fig. 2 shows the regions defined by these thresholds in the a-u space. [Sycara, 89] and [Zlotkin and Rosenschein 89] are examples of work exploring negotiation as a means to mediate among conflicting agents.

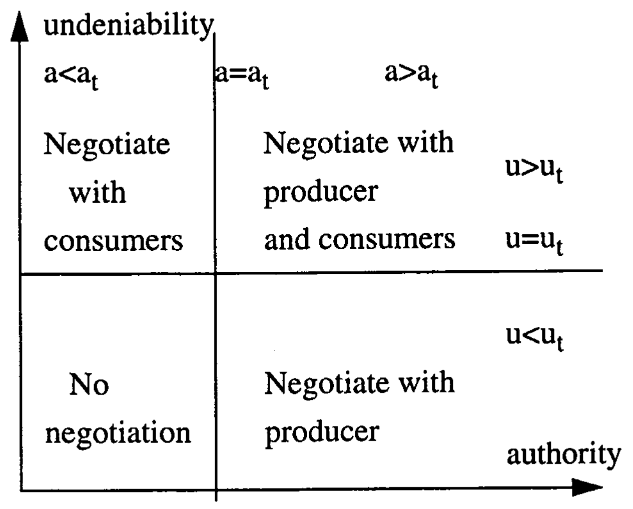

FIGURE 2. Negotiation regions in the a-u space.

3.4 The model

Putting together the above discussed elements we can formulate the following model for contradiction removal:

Given: a contradiction of the form: *p&q=>false*

1. Determine the support set of *p*, that is the set of premises *p* is derived from, and the support set of *q*, that is the set of premises *q* is derived from.

2. Group the propositions from both support sets into 4 sets corresponding to the 4 negotiation regions. In each region, order the component propositions in increasing order of the value of $r = (a^2+u^2)^{1/2}$

3. Considering the 4 regions in the order: (1) no-negotiation, (2) negotiation-with-producer, (3) negotiation-with-consumers, (4) negotiation-with-both, take each proposition in order and try to retract it:

- If the premise falls in the no-negotiation region, retract it

- If the premise falls into a negotiation region, negotiate with the required agents.

4. Retract the first proposition that passes the above tests and check if after retraction the contradiction can be rederived. If so, repeat the procedure. If no proposition can be retracted, report failure.

3.5 Implementation of the model

We have implemented this model in the prototype IA we have built for the TOVE project [Fox, 1993]. The IA uses a description logic language for the knowledge management component. The proposition base of the system is managed by a boolean constraint propagation TMS [McAllester 90]. The TMS was extended to incorporate the contradiction resolution model presented. The following issues arose during this implementation:

1. *When is authority specified?* In our implementation each producer specifies authority with each proposition the moment the proposition is sent. This ensures that the authority measure reflects the current producer's beliefs and may be more accurate than the deniability measures (see bellow why). Because of this feature, we choose to attempt to retract the propositions from the negotiate-with-producer region before those from the negotiate-with-consumers region.

2. *When is undeniability specified?* This raises some problems. The "most appropriate" time for specifying undeniability is when the proposition has to be retracted. This would imply that any retraction has to be done with negotiation (because undeniability is not known beforehand). We have chosen a solution that allows retraction without negotiation by allowing agents that query for information to specify how undeniable is the information to be supplied as a response. Hence our queries (and the persistent interest specifications) can specify the undeniability of the information that will constitute the response (e.g. "whatever answer you give to this query, it will be hard for me to deny it later"). The problem with this approach is that it may be hard for the consumer to correctly estimate in advance the costs of retraction. However, this scheme can be improved by allowing consumers to send to the IA incremental updates of retraction costs later, as they become known. In this way costs will be accurate and the IA can use them when conflict management is invocked.

3. *How is authority specified?* Any totally ordered set of values are acceptable. We currently use integer authorities (e.g the 1..10 range).

4. *How is undeniability specified?* The same response as above can be given. In the implementation we have versions using either boolean measures (*false* = can deny, *true* = better don't deny) or numerical measures (like the 1 .. 10 range).

5. *How are undeniability costs aggregated?* The measure of a proposition's undeniability must be aggregated from the deniability costs provided by each consumer. If a proposition has *n* consumers (directly or indirectly, through propositions derived from it), then a normalized cost can be obtained by averaging the costs reported by the consumers.

6. *Equivalence classes.* Propositions with the same values for authority and undeniability form equivalence classes. For propositions in negotiable regions the implemented algorithm tries *all* propositions in an equivalence class in order to retrive the "most retractable" proposition (the one reported as most acceptable for retraction by its producer and consumers). For propositions in the non-negotiable region, a random choice is made.

7. *Handling time mapped propositions.* Our representation language allows time mapped propositions. These propositions mention a time interval during which they are true or false. In this case, a contradiction *p&q=> false* exists iff *p* and *q* are true on a common subinterval. When retracting a premise, it is enough to guarantee that either *p* or *q* will be retracted on the common subinterval causing the contradiction. The TMS we use is extended to handle time-mapped propositions and can retract propositions on subintervals.

8. *Early detection of contradictions.* It is important to detect contradictions as early as possible. With a reasoning system that reasons backwards (inferring a proposition only when asked to do so) contradictions amongst propositions that are never queried may remain undetected. Because of this, we are using a truth-maintenance system that works forward, inferring all possible propositions whenever new premises are added.

4.0 Concluding remarks

The model of belief revision presented here has three main advantages. First, it is accurate because the selection of the retracted belief is based on the the views of all involved parties at the moment the contradiction is detected. This means that the selection implicitly relies on domain knowledge and on the current state of the global problem-solving effort. Second, by estimating costs and identifying negotiation regions, the model takes advantage from situations in which negotiation may not be required or in which a smaller amount of negotiation may suffice. Third, the model handles the new time-dependent inconsistencies that arise in time-mapped data-bases.

Although implemented, the model suffers however from the lack of experimental results that might demonstrate its effectiveness in applications. We hope to improve on that in the near future, when the distributed enterprise intergration environment will be experimented with.

As similar research is concerned[1], [Petrie 87] was the first to introduce reasoning about retraction in a TMS, in a manner that introduced domain dependent criteria. His work was carried out in the context of Doyle's JTMS [Doyle 79] that allows non-monotonic justifications. Our work uses a McAllester TMS that forbids non-monotonic justifications and has a more efficient (linear) labeling algorithm. Both of these use single context TMSs. It would be interesting to see how the authority/deniability model can be integrated in a multiple-context TMS as the ATMS [DeKleer 86]. Since the ATMS stores with each datum its complete set of prime implicants, the reasoner can symultaneously consider all candidate premises for retraction. In the single context case we can only use the current justifications and whatever premises support them (although other premises and justifications may exist as well). Because of this, in the multiple-context case we may

1. Other efforts have tried to maintain consistency in distributed TMS environments by devising distributed labeling algorithms. Examples are [Brigeland and Huhns 90] for the JTMS and [Mason and Johnson 89] for the ATMS [DeKleer 86]. Our problem however is not distributing the TMS algorithm as we make no assumption about the problem-solving mechanisms of the agents.

be able to avoid step 4 of our retraction algorithm (trying to rederive the contradiction and repeating the procedure for any new justifications of the contradiction) by retracting everything that has to be retracted from the beginning.

Acknowledgements

This research is supported, in part, by the Manufacturing Research Corporation of Ontario, Natural Science and Engineering Research Council, Digital Equipment Corp., Micro Electronics and Computer Research Corp., Spar Aerospace, Carnegie Group and Quintus Corp.

5.0 References

1. Allen, J., F., (1983) Maintaining Knowledge About Temporal Intervals, Communications of the ACM, 26 (11), pp 832-843, 1983.

2. McAllester, D., (1990) Truth Maintenance, Proceedings AAAI-90, pp. 1109-1116.

3. Barbuceanu, M. (1993) Models: Toward Integrated Knowledge Modeling Environments, Knowledge Acquisition 5, pp. 245-304.

4. Barbuceanu, M., Fox, M.S., (1994) The Information Agent: An Infrastructure Agent Supporting Collaborative Enterprise Architectures, to appear in Proceedings of the 3-rd Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises, IEEE Computer Society Press (to appear).

5. Borgida, A., Brachman, R. J., McGuiness, D. L., Resnick. L. A., (1989) CLASSIC: A Structural Data Model for Objects, Proc. 1989 ACM SIGMOD International Conference on Management of Data, june 1988, pp. 59-67.

6. Brachman, R. J., Schmolze, J. G., (1985), An Overview of the KL-ONE Knowledge Representation System, Cognitive Science 9(2), April-June 1985, pp. 171-216.

7. Bridgeland, D. M., Huhns, M., N. (1990) Distributed truth maintenance, Procedings of AAAI-90, pp 72-77

8. DeKleer, J. (1986) An Assumption Based TMS, Artificial Intelligence, 28(2), pp 127-244, march 1986

9. Doyle, J., (1979) A truth maintenance system, Artificial Intelligence, vol 12, no. 3, pp 231-272

10. Finin, T., et al. (1992) Specification of the KQML Agent Communication Language, The DARPA Knowledge Sharing Initiative, External Interfaces Working Group.

11. Fox, M. S., (1993) A Common-Sense Model of the Enterprise, Proc. of Industrial Engineering Research Conference, 1993.

12. Mason, C., Johnson, R.,R., (1989) DATMS: a framework for distributed assumption based reasoning. In Les Gasser and Michael N. Huhns, editors, Distributed Artificiall Intelligence, Volume II, pp. 293-317, Pitman Publishing, London, 1989

13. McGregor, R., Bates, R., (1987) The LOOM Knowledge Representation Language, ISI IRS-87-188, USC/ISI Marina del Rey, CA

14. Luck, K., Nebel, B., Peltason, C., Schmeidel, A., (1987) The Anatomy of the BACK System, KIT Report 41, Fachbereich Informatik, Technische Universitat Berlin, Berlin.

15. Pan, J.Y.C., Tenenbaum, J. M., (1991) An Intelligent Agent Framework for Enterprise Integration, IEEE Transactions on Systems, Man and Cybernetics, 21, 6, pp. 1391-1408, 1991.

16. Petrie, C., (1987) Revised dependency-directed backtracking for default reasoning, Proceedings of AAAI-87, pp 167-172

17. Roboam, M., Fox, M. S., (1992) Enterprise Management Network Architecture, A Tool for Manufacturing Enterprise Integration, Artificial Intelligence Applications in Manufacturing, AAAI Press/MIT Press, 1992.

18. Schmolze, J.G., (1985) The Language and Semantics of NIKL, BBN Inc., Cambridge, MA.

19. Shoham, Y., (1993) Agent-Oriented Programming, Artificial Intelligence 60 (1993) pp 51-92.

20. Sycara, K., (1989) Multi-agent compromise via negotiation, In Les Gasser and Michael N. Huhns, editors, Dis-

tributed Artificiall Intelligence, Volume II, pp. 119-137, Pitman Publishing, London, 1989

21. Zlotkin, G., Rosenschein, J.S. (1989) Negotiation and task sharing among autonomous agents in cooperative domains, Proceedings of IJACI-89, pp. 912-917, Detroit, MI, aug. 1989