Beliefs and Conflicts in a Real World Multiagent System

Benedita Malheiro[1], **László Zsolt Varga**[2], **Eugénio Oliveira**[3]
[1] DEE, ISEP, Rua de S. Tomé, 4200 Porto, Portugal, Email: `mbnm@fe.up.pt`
[2] Informatics Department, MTA SZTAKI, POB 63, H-1518 Budapest, Hungary
Email: `lvarga@lutra.sztaki.hu`
[3] DEEC, FEUP, Rua dos Bragas, 4099 Porto CODEX, Portugal
Email: `eco@fe.up.pt`

Abstract

In a real world multiagent system, where the agents are faced with partial, incomplete and intrinsically dynamic knowledge, conflicts are inevitable. Frequently, different agents have goals or beliefs that cannot hold simultaneously. Conflict resolution methodologies have to be adopted to overcome such undesirable occurrences.

In this paper we investigate the application of distributed belief revision techniques as the support for conflict resolution in the analysis of the validity of the candidate beams to be produced in the CERN particle accelerators.

This CERN multiagent system contains a higher hierarchy agent, the Specialist agent, which makes use of meta-knowledge (on how the conflicting beliefs have been produced by the other agents) in order to detect which beliefs should be abandoned. Upon solving a conflict, the Specialist instructs the involved agents to revise their beliefs accordingly.

Conflicts in the problem domain are mapped into conflicting beliefs of the distributed belief revision system, where they can be handled by proven formal methods. This technique builds on well established concepts and combines them in a new way to solve important problems. We find this approach generally applicable in several domains.

1 Introduction

The complexity of real world problem domains often results in the development of multiagent systems where the individual agents do not have complete knowledge about the domain problem although the community of agents is capable of solving the overall goal. Due to the lack of complete knowledge, the individual agent may have beliefs that are not coherent with the findings of other agents. As long as the agent is alone, its inferred propositions and beliefs are based on its own local knowledge base. When the agents form a community and try to solve common problems, then they exchange beliefs which have to be integrated into their own local knowledge bases. To perform this integration the agents must have mechanisms to determine whether the external beliefs and the local knowledge bases are coherent and, if not, must know how to find the best solution to achieve the common goals.

As a result, there is a need for conflict detection and conflict resolution methods within the knowledge base of a single agent. The conflict detection can be solved by extending the inference mechanism of the single agent with general integrity rules. The integrity rules state the condition of a conflict by specifying the beliefs that cannot hold simultaneously in the knowledge base of an agent. These integrity rules can be either domain independent or specific to the given application. When an integrity rule is fired, the inference mechanism of the agent activates the belief revision system to retrieve all the beliefs related with the conflicting beliefs and, thus, resolving the conflict within a single agent.

In order to achieve the common goal in an optimal way, it may not be enough to restore the integrity of the knowledge base in the agents, but it may also be necessary to modify the plans and actions of the agents. One possible solution is to use a supervisor or specialist agent containing the necessary meta-knowledge to resolve the conflict between the plans and actions of the agents. Every time a conflict is detected by the integrity rules, the involved agents notify the specialist agent, who, in turn, instructs the conflicting agents on how to revise their plans.

In this paper we investigate these issues in a real multiagent system by proposing a distributed belief revision system as a model for conflict detection and resolution. Section 2 briefly overviews the CERN multiagent system application, Sections 3 and 4 introduce the reader to distributed belief revision, Section 5 discusses a real world scenario showing conflicting beliefs and the way they are resolved, Section 6 analyses the specific scenario and abstracts the generic method of adapting distributed belief revision to conflict resolution, in Section 7 conclusions are drawn.

2 Problem Domain

CERN is an European research institute and its particle accelerator compound is one of the world's most sophisticated high energy research centers. The accelerator operation and the maintenance of the underlying computing control system are complex tasks, difficult to survey, and where the necessary knowledge and the control system are naturally distributed.

Because these facts suggested the application of DAI methods, CERN joined the ARCHON project [Jennings and Wittig, 1992], [Wittig, 1992] as an application partner providing a large accelerator control system and two expert systems as a test-bed for the development and evaluation of the methodologies and software produced within this project. Problems, insights and experiences gained whilst deploying ARCHON technology in real applications are described in [Jennings *et al.*, 1996]. These investigations [Jennings *et al.*, 1993] were among the first experiences in creating operational DAI systems by transforming existing intelligent systems of a real world application into members of a multi-agent community. Several possibilities for cooperation were detected at CERN including the cooperation between accelerator setup and diagnosis, as well as the cooperation between the different aspects of timing diagnosis [Skarek and Varga, 1996a].

In this paper we will use a new cooperation scenario centered around an expert system developed at CERN using database technology [Lewis *et al.*, 1995], [Skarek and Varga, 1996b] and mentioned in this paper later as BCD Checker. This scenario is particularly well suited for the investigation of conflict resolution because conflicts and conflict resolution are inherent and necessary for the everyday operation of the accelerators. If the agent that creates the beam schedule for the accelerators had complete knowledge about the physical constraints of the accelerators and their control system, then it would be too complex. As a result, the BCD Editor agent has incomplete knowledge - when it creates plans to achieve its goals, it creates beliefs that may be inconsistent with the beliefs of the BCD Checker agent. The BCD Checker agent incorporates the knowledge about the accelerators and control system constraints. In order to create an executable beam schedule for the accelerators, the agents may have to revise their beliefs and create new plans. In this paper we will apply the distributed belief revision model to solve these problems and we will study the relationship between belief revision and conflict resolution.

3 Distributed Belief Revision

Reasoning with incomplete, inaccurate knowledge can be achieved by performing belief revision, i.e., abandoning some of the existing beliefs in order to accommodate new evidences or recent findings.

Standard knowledge representations do not handle beliefs adequately. Beliefs are intrinsically non-monotonic: their belief status may change according to newly perceived world changes, received beliefs, or inferred propositions.

The approach we use for modelling belief is based on justifications, also referred as foundations theory of belief. According to this theory, belief revision consists, first, in giving up all beliefs that no longer have a satisfactory justification and, second, in adding new beliefs that have become justified [Gärdenfors, 1992].

An agent with belief revision skills which becomes aware of the existence of counterparts with whom to exchange beliefs relevant to the undergoing activity is said to perform distributed belief revision.

The ability of sending and receiving beliefs poses an important set of questions to agents: in which circumstances should an external belief be accepted? how to represent an external belief locally? should an external belief be allowed to trigger the revision of internal beliefs? how to accommodate different belief status regarding the same data item?

Acceptance of External Beliefs

When should an external belief be added to an agent knowledge base? The two fundamental approaches concerning the inclusion of communicated beliefs are [Malheiro and Oliveira, 1996a]:

- local consistency of the shared propositions - internal beliefs prevail over external beliefs. An external belief is only used in the absence of an internal belief regarding the same proposition. As soon as an agent infers a justification for a previously communicated proposition, only the locally inferred belief will be used by the agent.
- global consistency of the shared propositions - every existing belief (internal or external) is unconditionally used.

We chose global consistency for the shared propositions.

Accommodation of Different Perspectives

Upon accepting an external belief, an agent may find itself with conflicting belief status for the same proposition (*believed a* and *unbelieved a*) or with contradictory beliefs (*a* and *NOT a*). Although there is no guarantee of solving every conflict, different methodologies can be adopted: they range, among others, from the adoption of static synthesis criteria [Malheiro and Oliveira, 1996b] to dynamic approaches such as argumentation [Verheij, 1995] or negotiation [Sycara, 1989].

The synthesis criteria we adopted is static. The assignment of an unique belief status to every shared proposition is achieved trough the application of a conjunctive (*AND*) synthesis criterion: a shared proposition is believed, if and only if, it is believed by every agent that shares the proposition. With this *AND* criterion only the consensus among the involved agents makes a shared proposition believed by the system.

4 The Agent Architecture

The cooperating deliberative agents implemented have an ARCHON-like architecture [Wittig, 1992] made of a domain level layer and a cooperation layer. In our case, the

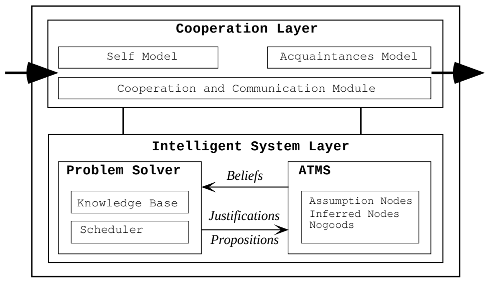

Figure 1: Agent Architecture

domain level system, containing the domain expertise, is an assumption based belief revision system with a problem solver and an assumption based truth maintenance system (ATMS) [de Kleer, 1986]. The problem solver is a data driven production rule based system which works in tight connection with the ATMS.

The functionality related to cooperation is represented as a distinct layer, the cooperation layer. The cooperation layer has the following components [Wittig, 1992]: (i) a cooperation module responsible for engaging in cooperative actions (mostly result sharing); (ii) a communications module responsible for sending/receiving asynchronous messages; (iii) a self model representing the information about the intelligent system layer; and (iv) an acquaintances model containing the relevant information regarding the acquaintances with whom the agent is expected to interact. Based on the self and acquaintances models each agent divides its knowledge base in two separate sets, the set of private beliefs - beliefs for exclusive internal agent use, and the set of shared beliefs - beliefs shared with at least one acquaintance. Private beliefs are revised automatically by the local ATMS and shared beliefs are revised according to methodology adopted for the shared propositions belief revision.

4.1 Assumption based Belief Revision

The operation of any ATMS is based on a special type of propositions – the system assumptions. Assumptions represent knowledge that is assumed to be compliant with the current system state and, therefore, believed. The assumptions are dynamic beliefs which may have to be dropped in face of new findings. Propositions are inferred from the existing assumptions, facts and rules. As a result, when an assumption is abandoned the propositions depending on the assumption have to be revise accordingly. Internally, the ATMS handles nodes and justifications. Nodes represent information (propositions and rules) and justifications represent inferences relating existing nodes. The ATMS computes and keeps updated every set of assumptions (support set) from which the registered propositions were inferred. This way the system is able to attribute justified belief status to each proposition: (i) *BEL or believed* – when the proposition has at least one valid support set; (ii) *UBEL or unbelieved* – when the proposition has no support set whatsoever; (iii) *UKNO or unknown* – when the proposition is not represented in the ATMS.

In an assumption based belief revision system the problem solver contains two sets of rules: (i) inference rules – which, once fired, provide justifications for believing in the consequent propositions, and (ii) integrity rules – that detect the existence of integrity violations implying, as a result, abandoning previously adopted beliefs. The inference engine is a two step data driven engine: the arrival of a new input triggers first the integrity rules and last the inference rules.

Before firing any rule (integrity or inference rule) the problem solver queries the ATMS about the belief status of the rule antecedents. If the rule antecedents are currently believed then the rule is fired and the rule outcome is communicated to the ATMS:

- If the rule is an integrity rule, the ATMS receives the detected inconsistency or *nogood* (*INCONSISTENT* with the agent's knowledge base, and therefore *unbelieved*). The ATMS records the nogood and removes the inconsistent set of assumptions from every support set;
- If the rule is an inference rule, the ATMS receives the new proposition together with its justification (the rule and rule antecedents) and computes the proposition new support set (*CONSISTENT* with the agent's knowledge base, and therefore *believed*).

4.2 Knowledge Representation

The node structure is defined through the following BNF grammar:

```
<NodeDef>  ::= Proposition <Type> <Status> <Scope>
               <Label> OwnerAgent
<Type>     ::= <Belief> | Fact
<Scope>    ::= SharedInternal | SharedExternal | Private
<Status>   ::= Believed | Unbelieved | Unknown
<Label>    ::= {{}} | {<NodeSet>}
<NodeSet>  ::= <NodeDef> | <NodeSet>
<Belief>   ::= Assumption | InferBelief
```

The node structure slots have the following meaning:
Proposition - the identifier of the proposition;
Type - the type of the proposition (assumption, fact or inferred node);
Scope - the scope of the proposition (private, shared internal or shared external);
Status - the belief status of the proposition (believed or unbelieved);
Label - the sets of assumptions from which the proposition was inferred (the support sets);
OwnerAgent - the agent that owns the node.
The inference and integrity rules are defined by the following BNF grammar:

```
<InferRule> ::= RuleId IF <Cond> THEN <Cons>
<IntegRule> ::= RuleId IF <Cond> THEN <InCons>
<Cond>      ::= <OperTerm> | <OperTerm> AND
                <Cond>
<OperTerm>  ::= <Operator> <Term> | <Term>
<Operator>  ::= BEL | UBEL | UKNO
<Cons>      ::= CONSISTENT <Term> | CONSISTENT
                <Term> AND <Cons>
<InCons>    ::= INCONSISTENT <Term> |
                INCONSISTENT <Term> AND
                <InCons>
<Term>      ::= Predicate | NOT Predicate
```

5 The Scenario with Conflict

The adequate management of a critical resource such as the CERN particle accelerator compound needs a multiagent system capable of setting up valid beam schedules. This scenario introduces a multiagent system made of 3 coarse grain deliberative agents: the BCD Editor, the BCD Checker and the Specialist agents. The common goal of the BCD Editor and BCD Checker agents is to create a valid BCD (a BCD is a beam schedule diagram described later). The BCD Editor creates preliminary plans for the creation of the BCD, while the BCD Checker validates these BCDs. A conflict occurs whenever the BCD created by the BCD Editor is not accepted by the BCD Checker. It is the Specialist agent who has the knowledge of what to remove from the preliminary BCD in order to solve the conflict.

5.1 Agents

The actual multiagent system consists of the following agents:

Agent 1: BCD Editor – This agent represents the operator setting up a beam schedule on the BCD Editor. A beam schedule is a sequence of beam acceleration cycles executed by the interconnected accelerators in a time sharing manner. Beam schedules are called BCDs, meaning Beam Coordination Diagrams. The aim of the operator in this scenario is to create, according to the physicists request, a BCD containing two machine cycles which produce lepton particles. The operator creates the BCD with the BCD Editor and stores it in the Oracle database. In order to achieve this goal, the operator has to keep to the following rule:

RU1:
IF BEL cycle($Cycle_a$, lepton)
AND BEL cycle($Cycle_b$, lepton)
AND member($Cycle_a$, BCD)
AND member($Cycle_b$, BCD)
THEN CONSISTENT ok(BCD)

Agent 2: BCD Checker – The role of the BCD Checker is to verify if the BCD created by the BCD Editor satisfies the constraints posed by the accelerators and their control systems. These constraints are represented by several rules defined by specialists of the different parts of the accelerator system.

One of the rules relevant for our scenario states that if a cycle is preceded by a higher energy cycle, and the higher energy cycle is not long enough (the magnets have hysteresis and enough time must be provided to set them to the right value by driving them through a so called up-min-max transition) then the BCD is not executed correctly. The formal representation of this rule is the following:

RC1:
IF BEL consecutive($Cycle_a$, $Cycle_b$)
AND BEL destination($Cycle_a$, X)
AND BEL destination($Cycle_b$, X)
AND BEL duration($Cycle_a$, D_a)
AND smallerThan(D_a, LongValue)
AND greaterThan(energy($Cycle_b$),energy($Cycle_a$))
AND member($Cycle_a$, BCD)
AND member($Cycle_b$, BCD)
THEN CONSISTENT NOT ok(BCD)

The other constraint relevant for our scenario is related to the optimal operation of the accelerators. During each lepton cycle a costly operation, the activation of the 114 MHz cavities, occurs. To minimize the number of activation/deactivation operations within a BCD, lepton cycles must be positioned consecutively – with consecutive lepton cycles only one activation and one deactivation is executed within a BCD. This constraint is represented in the BCD Checker by the following rule:

RC2:

IF BEL cycle($Cycle_a$, lepton)
AND BEL cycle($Cycle_b$, lepton)
AND BEL cycle($Cycle_c$, not_lepton)
AND BEL consecutive($Cycle_a$, $Cycle_c$)
AND BEL consecutive($Cycle_c$, $Cycle_b$)
AND member($Cycle_a$, BCD)
AND member($Cycle_b$, BCD)
THEN CONSISTENT NOT ok(BCD)

Agent 3: Specialist – The specialist has meta-knowledge about the meaning and importance of the rules of the BCD Checker, establishing a rule preference order. For example it knows that the RC1 rule of the BCD Checker always has to be satisfied, otherwise the accelerators will not produce the necessary beams, but - if necessary - the RC2 rule of the BCD Checker can be ignored, because even if the accelerators do not operate optimally, then they still produce the necessary beams. This knowledge is represented by the following meta-rules:

Meta-R1: IF fired(RC1, BCD)
THEN INCONSISTENT ok(BCD)
Meta-R2: IF fired(RC2, BCD)
THEN INCONSISTENT NOT ok(BCD)
Meta-R3: IF NOT fired(RC2, BCD)
THEN CONSISTENT NOT finantialLoss
Meta-R4: IF fired(RC2, BCD)
THEN CONSISTENT finantialLoss
Meta-R5: IF BEL ok(BCD)
THEN CONSISTENT produceBeam
Meta-R6: IF BEL produceBeam
AND BEL finantialLoss
THEN CONSISTENT satisfiedUsers
Meta-R7: IF BEL produceBeam
AND BEL NOT finantialLoss
THEN CONSISTENT verySatisfiedUsers

The multiagent system presented exhibits the following features: (i) the agents are benevolent and thus truthful from their own point of view; (ii) a shared proposition is represented by as many nodes as there are agents holding a belief regarding it (multiple node representation); (iii) a node can only be revised by the agent it belongs to or by the Specialist agent; (iv) the existing beliefs (internal and external) regarding any proposition are accommodated through an *AND* synthesis criterion.

5.2 Steps of the Scenario

Now we are going to demonstrate with the help of the scenario how the CERN multiagent system achieves its goal of guaranteeing that only beam schedules compliant with the existing constraints are executed. The scenario is started when the BCD editor submits a new beam schedule:

Step 1 – The BCD Editor creates the new BCD_X with consecutive lepton cycles $cycle_a$ and $cycle_b$. The description of BCD_X will be represented in the Truth Maintenance System of the BCD Editor by several assumptions and facts.
Rule RU1 provides the BCD Editor with a justification for believing in *ok(BCD_X)*. The BCD Editor communicates this finding and BCD_X to the BCD Checker and to the Specialist agents.

Step 2 – The BCD Checker is able to fire RC1, concluding that *NOT ok(BCD_X)* is believed. The BCD Checker communicates its belief on *NOT ok(BCD_X)* both to the Specialist and to the BCD Editor agents, and informs the Specialist that rule RC1 has been fired. A conflict between *NOT ok(BCD_X)* and *ok(BCD_X)* has been generated.

Step 3 – Through Meta-R1 the Specialist agent concludes that the belief in *ok(BCD_X)* is inconsistent. The Specialist notifies the BCD Editor agent to revise its belief on *ok(BCD_X)* accordingly, and to create a BCD for which RC1 won't fire.

Step 4 – The BCD Editor revises BCD_X and creates BCD_Y by inserting $cycle_c$ in between $cycle_a$ and $cycle_b$ to satisfy the RC1 rule. Unfortunately the energies of $cycle_c$ and $cycle_b$ are such that RC2 is not satisfied.
Based on RU1, the proposition *ok(BCD_Y)* becomes believed by the BCD Editor. The BCD Editor communicates BCD_Y and its belief in *ok(BCD_Y)* to the BCD Checker and to the Specialist agents.

Step 5 – The BCD Checker fires rule RC2, and finds that *NOT ok(BCD_Y)* is believed. The belief in *NOT ok(BCD_Y)* is communicated to the Specialist and the BCD Editor agents, and the Specialist is informed that rule RC2 fired for BCD_Y. The conflict created by the simultaneous belief on *NOT ok(BCD_Y)* by the BCD Checker and on *ok(BCD_Y)* by the BCD Editor can be solved by the Specialist.

Step 6 – From Meta-R2 the Specialist concludes that the belief in *NOT ok(BCD_Y)* is inconsistent with the current context. This outcome of the conflict is communicated both to the BCD Editor and to the BCD Checker. As a result *NOT ok(BCD_Y)* becomes unbelieved and *ok(BCD_Y)* remains believed. From Meta-R3 the Specialist concludes that *produceBeam* is believed. The Specialist accepts BCD_Y.

6 Belief Revision as Conflict Resolution

According to [Zlotkin and Rosenschein, 1991] and [Zlotkin and Rosenschein, 1990], a conflict exists when two agents cannot achieve their goals simultaneously: only one of the agents, not both, is capable of achieving its goal. When both agents achieve their goals simultaneously, optimally or not, then a cooperative or compromise situation occurs.

In the distributed belief revision model used, a conflict occurs, when the same proposition is assigned different belief status by different agents (*BEL a* and *UBEL a*) or when contradictory beliefs coexist (*BEL a* and *BEL NOT a*).

In the CERN scenario, the goal of the BCD Editor agent is to guarantee the existence of two lepton cycles in the BCD. This is possible in many different ways, so the goal of the BCD Editor agent can be represented by a set of BCDs, hereby called Editor Set. Any BCD that falls into the Editor Set is acceptable from the BCD Editor agent's point of view. If the BCD Editor agent achieves its goal, then it believes *ok(BCD_X)*. The goal of the BCD Checker agent is to test if the BCD is compliant with the constraints of the accelerators and their control systems. This is represented by the BCD checking rules RC1 and RC2. Only those BCDs which fail to trigger both RC1 and RC2 are acceptable for the BCD Checker and thus belong to the Checker Set. If the BCD Checker agent achieves its goal without firing either RC1 or RC2, then it also believes *ok(BCD_X)*.

Fortunately, the Editor Set and the Checker Set are usually not exclusive: there is at least one BCD which is member of both sets. From the point of view of goals, this is not a conflict situation, because both agents can achieve their goals, but they have to cooperate to do so. However, when the agents plan their actions, they sometimes generate plans which contain conflicting beliefs. The conflicting belief, in this case, indicates that the goal state of the BCD Editor agent is not in the intersection of the Editor Set and the Checker Set. When conflicting beliefs are detected the agents have to modify their plans. The absence of conflicting beliefs in the modified plan means that the goal state of the BCD Editor agent is in the intersection of the Editor Set and the Checker Set.

At the starting point of the scenario presented, the agents are in a situation where they have conflicting goal states, indicated by conflicting beliefs. As a consequence, they try to modify their plans, in order to resolve the conflicting beliefs. When the Specialist agent verifies that only the RC2 rule has been triggered in the BCD Checker, it concludes, via meta-rules, that RC2 is not that important. By doing so, the Specialist agent extends the Checker Set to include the goal state of the BCD Editor agent into it. With this procedure the Specialist agent turns the conflicting situation into a compromise situation.

In the CERN scenario the conflict occurred between the belief in *ok(BCD_Y)*and the belief in *NOT ok(BCD_Y)*. From the analysis of the set of possible situations we can verify that belief revision acts as a conflict resolution methodology:

- If neither RC1 nor RC2 were fired then *ok(BCD_i)* is believed by the BCD Editor agent and there is no conflict;
- If only RC1 was triggered then *ok(BCD_i)* is believed by the BCD Editor agent and *NOT ok(BCD_i)* is believed by the BCD Checker agent. Meta-R1 from the Specialist agent is fired, and, as a result, *ok(BCD_i)* becomes unbelieved. The Specialist communicates this outcome to the BCD Editor. The BCD Editor revises its belief in *ok(BCD_i)* accordingly;
- If only RC2 was fired then *ok(BCD_i)* is believed by the BCD Editor agent and *NOT ok(BCD_i)* is believed by the BCD Checker agent. meta-R2 from the Specialist agent is triggered, and, as a result, *NOT ok(BCD_i)* becomes unbelieved. The Specialist communicates this outcome to the BCD Checker, who revises its belief in *NOT ok(BCD_i)* accordingly;
- If both RC1 and RC2 were triggered then *ok(BCD_i)* is believed by the BCD Editor agent and *NOT ok(BCD_i)* is believed by the BCD Checker agent. Both meta-R1 and meta-R2 fire. Consequently, both *ok(BCD_i)* and *NOT ok(BCD_i)* become unbelieved. The BCD Editor and the BCD Checker revise their beliefs accordingly.

7 Conclusions

We investigated conflict resolution in a real multiagent system by applying distributed belief revision techniques. We combined existing concepts and techniques in a new way to resolve conflicts. The key factors of conflict resolution in this approach are the distributed belief revision system and the meta-knowledge of the Specialist agent. The distributed belief revision system detects the occurrence of conflicting beliefs or belief statuses in the plans of the agents and initiates the modification of their plans towards a jointly acceptable goal state. The meta-knowledge of the Specialist agent (how the conflicting beliefs have been produced) is used to extend the set of goal states of one of the agents and to choose which belief status to change.

The investigations also showed that, although in a strict theoretical sense there can be no conflict between the agents, because in the end they find a common goal state, in practice, conflicting plans and beliefs do occur due to the incomplete knowledge of the agents. Distributed belief revision system and meta-rules help the agents coordinate their actions and overcome this problem.

8 Acknowledgments

This work has been partially funded by JNICT and OMFB under contracts 423/OMFB and P-5/95, respectively. We are thankful to Paul Skarek from CERN for identifying and providing the valuable real world scenario on which the investigations of the paper are based.

References

[de Kleer, 1986] J. de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, 1986.

[Gärdenfors, 1992] P. Gärdenfors. Belief revision: An introduction. In *Belief Revision*, Cambridge Tracts in Theoretical Computer Science, pages 1–28. Cambridge University Press, 1992.

[Jennings and Wittig, 1992] N. R. Jennings and T. Wittig. ARCHON: Theory and practice. In N. M. Avouris and L. Gasser, editors, *Distributed Artificial Intelligence: Theory and Praxis*. Kluwer Academic Publishers, 1992.

[Jennings *et al.*, 1993] N. R. Jennings, L. Z. Varga, R. P. Aarnts, J. Fuchs, and P. Skarek. Transforming standalone expert systems into a community of cooperating agents. *Engineering Applications of Artificial Intelligence*, 6(4):317–331, 1993.

[Jennings *et al.*, 1996] N. R. Jennings, E. H. Mamdani, J. M. Corera, I. Laresgoiti, F. Perriollat, P. Skarek, and L. Z. Varga. Using ARCHON to develop real-world DAI applications. *IEEE Expert*, 11(6):64–86, 1996.

[Lewis *et al.*, 1995] J. Lewis, P. Skarek, and L. Z. Varga. A rule-based consultant for accelerator beam scheduling used in the CERN PS complex. In *International Conference on Accelerator and Large Experimental Physics Control Systems, ICALEPS'95, Chicago, USA*, pages 703–707, 1995.

[Malheiro and Oliveira, 1996a] B. Malheiro and E. Oliveira. Consistency and context management in a multi-agent belief revision testbed. In M. Wooldridge, J. P. Müller, and M. Tambe, editors, *Intelligent Agents Volume II — Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)*, Lecture Notes in Artificial Intelligence, pages 361–375. Springer-Verlag, 1996.

[Malheiro and Oliveira, 1996b] B. Malheiro and E. Oliveira. Intelligent distributed environmental decision support system. In D. L. Borges and C. A. A. Kaestner, editors, *Advances in Artificial Intelligence (SBIA'96)*, Lecture Notes in Artificial Intelligence, pages 171–180. Springer-Verlag, 1996.

[Skarek and Varga, 1996a] P. Skarek and L. Z. Varga. Multi-agent cooperation for particle accelerator control. *Expert Systems With Applications*, 11(4):481–487, 1996.

[Skarek and Varga, 1996b] P. Skarek and L. Z. Varga. Rule-Based Knowledge Representation Using a Database. In J. Zarka, editor, *Proceedings of the Conference on Artificial Intelligence Applications (EXPERSYS-96)*, IITT Technology Transfer Guide Book Series, pages 199–206, 1996.

[Sycara, 1989] K. P. Sycara. Multiagent compromise via negotiation. In L. Gasser and M. N. Huhns, editors, *Distributed Artificial Intelligence*, volume II of *Research Notes in Artificial Intelligence*, pages 119–137. Morgan Kaufmann, 1989.

[Verheij, 1995] B. Verheij. Arguments and defeat in argument-based nonmonotonic reasoning. In C. Pinto-Ferreira and N. J. Mamede, editors, *Porgress in Artificial Intelligence (SBIA'96)*, Lecture Notes in Artificial Intelligence, pages 213–224. Springer-Verlag, 1995.

[Wittig, 1992] T. Wittig, editor. *ARCHON: An Architecture for Multi-agent Systems*. Ellis Horwood Series In Artificial Intelligence. Ellis Horwood, 1992.

[Zlotkin and Rosenschein, 1990] G. Zlotkin and J. S. Rosenschein. Negotiation and conflict resolution in non-cooperative domains. In *Proceedings of the AAAI*, 1990.

[Zlotkin and Rosenschein, 1991] G. Zlotkin and J. S. Rosenschein. Incomplete information and deception in multi-agent negotiation. In *Proceedings of the IJCAI*, pages 225–231, 1991.