

Managing Conflicts in Reflective Agents

Frances M.T. Brazier and Jan Treur

Vrije Universiteit Amsterdam
Department of Mathematics and Computer Science
Artificial Intelligence Group
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
Email: {frances, treur}@cs.vu.nl
URL: <http://www.cs.vu.nl/~frances, ~treur>

Abstract

This chapter addresses management of conflicts in an agent by means of reflective reasoning. A structure for reflective agents is proposed within which reasoning about observation, assumption making and communication; an agent's own information state and reasoning processes; other agents' information states and reasoning processes, and combinations of these types of reflective reasoning are explicitly modelled. The types of knowledge needed to detect, analyse and resolve conflicts that arise by meta-reasoning within the agent are discussed. The knowledge and interaction between agents required to model the wise men's puzzle is used to illustrate the approach.

1. Introduction

Although not all distributed intelligent systems are designed as multi-agent systems, many are. The metaphor of autonomous agents in interaction with each other and the external world, provides a conceptual basis for the design of distributed systems for which interaction is of primary importance. The intelligent systems themselves can often be described in terms of characteristics associated with the notion of weak agency (Wooldridge and Jennings, 1995). The characteristics of this notion of agency: autonomy, pro-activeness, social ability and reactivity, provide a means to characterise the behaviour of an intelligent system. Pro-activeness and autonomy are related to a system's ability to reason about its own processes, goals and plans, and to control these processes. Reactivity and social ability are related to the ability to be able to communicate and co-operate with other systems and to interact with the external world. In this chapter such distributed intelligent systems are viewed to be multi-agent systems. Each individual system is viewed to be an autonomous agent.

Autonomous agents are often reflective agents: agents capable of reasoning, for example, not only about the behaviour of the external world, but also about their own behaviour, and other agents' behaviour. More specifically, reflective agents are able to reason about:

- their own information states
- their own assumptions
- the control of their own reasoning processes (e.g., which strategy to follow and when)
- their observations (e.g., which observations to perform and when)
- their actions (e.g., which actions to perform and when)

- their communication with others (e.g., which communication to perform, with which other agents)
- other agents' processes (their information states, assumptions, reasoning processes, observations, communication and actions in the external world)
- interaction between agents (e.g., the extent to which co-operation is successful)
- their own tasks

Reasoning about reasoning, *meta-reasoning*, is essential to most problem solving behaviour. Reasoning about reasoning includes reasoning about conflicts. Conflicts occur continually, at all levels within a reasoning process, not only due to unexpected events, but often as an explicit part of a reasoning solving process, on purpose, to learn from the management and evaluation of the conflict. When monitoring and guiding a reasoning process, for example, an agent needs to decide which choices to make and when (given conflicting options), which choices to re-consider, which to accept. Which diagnostic strategy to employ is, for example, a question with which human doctors are confronted, but also automated diagnostic systems. Often the choice of strategy depends on the availability, quality and cost of relevant information. For medical diagnosis the benefit and cost of the acquisition of additional information includes consideration of a patient's comfort, risk, estimated information value, and financial implications - often conflicting factors. Which choices are made depends on the strategy deployed, but often need to be adapted continually, depending on the information acquired and the state of an agent's knowledge.

Another example of a situation in which an agent needs to be able to continually reason about conflicts:

Centralised air traffic control has resulted in limited use of the total available space: a limited number of "highways" have been defined within which all aircraft are scheduled. Currently the concept of free flight is being explored: aircraft are free to fly the route they themselves determine with very limited interaction with other aircraft. New traffic rules are being devised to this purpose. One of the main aspects involved is that to be able to determine his/her own course, a pilot needs to be able to reason about the expected behaviour of other aircraft: a pilot needs to be able to reason about a specific situation from the perspective of another pilot given limited information such as the characteristics of the aircraft, its position, its destination. On the basis of this information the pilot can determine his/her own strategy to adapt his/her own course, if and when conflicts occur.

Reasoning about conflicts necessitates reasoning about

- uncertainty of facts and/or inferences
- inconsistency of facts and/or inferences
- availability and adequacy of information in a given situation
- types of interaction needed
- (default) assumptions and the current information state

Often such meta-reasoning is needed at different levels. The pilot, for example, needs to decide whether his/her own past experience with specific airlines should influence his/her decision with respect to the best course to take, given the analysis of the other aircraft's expected behaviour. If the other airline has, in the past, shown to be reliable, a pilot may decide to rely on his/her own initial analysis. If not, the pilot may decide to adjust his/her own course to minimise the chance of conflicting courses. The pilot's analysis is based on the pilot's own observations, but may also be influenced by additional information acquired from other sources. Modelling this type of reasoning requires non-trivial nesting of reasoning. An arbitrary number of meta-levels may be needed, depending on the intricacy of the reasoning process.

In the literature on reflection such as (Weyhrauch, 1980; Maes and Nardi, 1988; Attardi and Simi, 1994) a restricted number of the types of reflective reasoning distinguished above, are modelled. Non-trivial combinations of different types of reflective reasoning, however, have not been studied extensively. In the literature on modelling in the context of multi-agent systems, most often the types of reflective reasoning agents are capable of performing is limited.

In this chapter an agent model is introduced that models non-trivial combinations of reflective reasoning, to model management of conflicts. This model has been used to model distributed air traffic control, as discussed above. A generic agent model is introduced in Section 2 and refined in Section 4 for a reflective agent capable of performing the types of reasoning about conflicts listed above, illustrated for the specification of the wise men's puzzle (introduced in Section 3). An analysis of conflict management within the example domain is presented in Section 5. The role of meta-reasoning and reflection in the context of conflict management is discussed in Section 6.

2. Reflective reasoning in a generic agent model

To design a generic structure for autonomous agents capable of reflective reasoning, the types of reasoning agents can be expected to perform, must be distinguished. In (Brazier, Jonker and Treur, 1997) a compositional generic agent model was introduced which distinguishes seven main processes, modelled by components (see Figure 1). The types of reflective reasoning performed in each of these components are briefly discussed in this section.

Reasoning about *the external world* (MWI) is a basic type of reasoning reflective agents are assumed to be capable of performing. A reflective agent needs to be able to reason about a specific situation, extending its own knowledge, by, for example, confirming or rejecting assumptions about the world made in previous reasoning processes.

As autonomous agents capable of interacting with the external world, reflective agents must also be capable of reasoning about *interaction with the external world* (WIM): about, for example, the types of information that can be observed in the external material world, when and how, but also which actions are to be performed, when and how.

Autonomous reflective agents need to be capable of reasoning about their *own processes (OPC)*. Reflective agents need to be able to reason about their own characteristics, capabilities and goals, of their success or failure in achieving these goals, about assumptions which need to be or have been made and when, about information which has been sought and not yet found, about information which has not yet been explored, about strategic preferences, about control, and about all other aspects of their own reasoning and acting.

Reflective agents also need to be capable of reasoning about *other agents' processes (MAI)*. Reflective agents need to be able to reason about the information available to other agents, about their (reasoning) capabilities, their goals and success (or lack thereof), their strategic preferences, their assumptions, et cetera.

To interact with other agents, reflective agents must be capable of reasoning about *interaction between agents(AIM)*. Agents not only need to be able to reason about which information can be obtained by communication with which other agents, but also about how and when this communication has to be initiated.

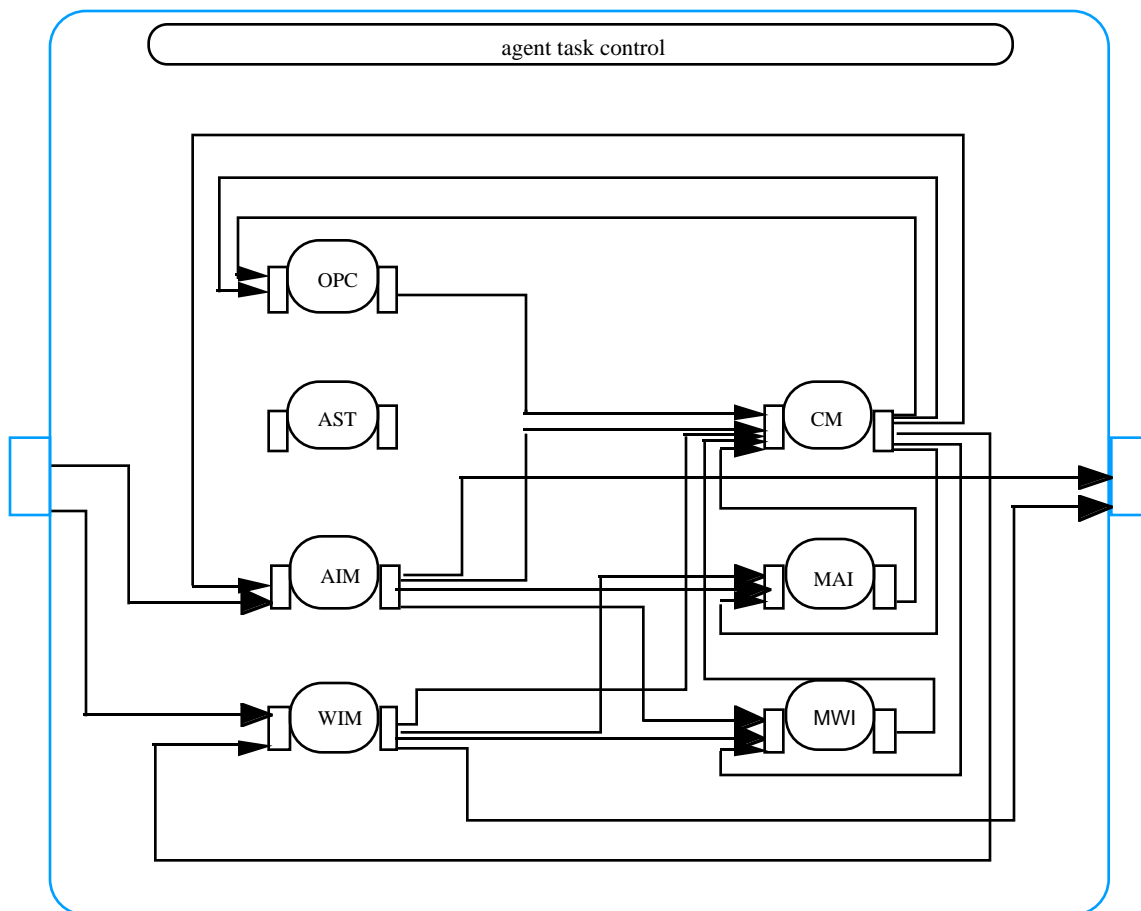


Figure 1. A generic agent model

In situations in which *co-operation between agents (CM)* is required a reflective agent needs to be able to reason about the type of co-operation required, its success, its failure and appropriate actions to take.

Last, but not least, a reflective agent's *agent specific tasks* (AST) require reasoning, but also often include reasoning about the tasks the agent is to perform: about the way in which a task is to be approached, about assumptions which can be made, for example.

The seven types of reasoning distinguished above correspond to the seven generic components depicted in the generic agent model presented in Figure 1. These tasks are generic in the sense that all autonomous agents are assumed to be capable of performing these tasks. The corresponding components are most often composed. The number of levels of reasoning involved depend on the complexity of the tasks for which they are designed.

Within each of these components, an agent must be able to reason about conflicts. Conflicts within *own process control*, may be explicitly modelled as conflicts in beliefs, desires and intentions. A generic model for reasoning about beliefs, desires and intentions, in which such conflicts are explicitly modelled is proposed in (Brazier, Dunin-Keplicz, Treur, Verbrugge, 1998). The concepts applied in this model are strongly related to the concepts distinguished by Castelfranchi (1998). Conflicts within *co-operation management* are addressed in the co-operation model applied in (Brazier, Jonker, Treur, 1996); for a more detailed specification, see (Brazier, Jonker, Treur, 1997). Conflicts in the *agent specific task* design are discussed in (Brazier, Langen, Treur, 1995). This chapter focuses on a refinement of the generic agent model for a reflective agent, and how specific types of conflicts are deliberately introduced and managed in this model. Note that this chapter focuses on conflicts within an agent and not on the management of conflicts between agents (which is addressed in, for example, Brown (1998)).

3. An example reflective reasoning process

To illustrate the different levels involved in an example of reflective reasoning about conflicts, a simple version of the wise men's puzzle is used. This puzzle requires two wise men (A and B) and two hats. Each wise man is wearing a hat, of which the colour is unknown. Both wise men know that:

- hats can be white or black
- there is at least one white hat
- they can observe the colour of each other's hat
- they both reason fully logically.

Assume, for example, that both men have a white hat and that wise man A is asked whether he knows the colour of his hat. Wise man A must answer that he is incapable of drawing a conclusion about the colour of his own hat. On the basis of this knowledge wise man B can then reason that its own hat is white. This reasoning process is depicted below in Figures 2 and 3.

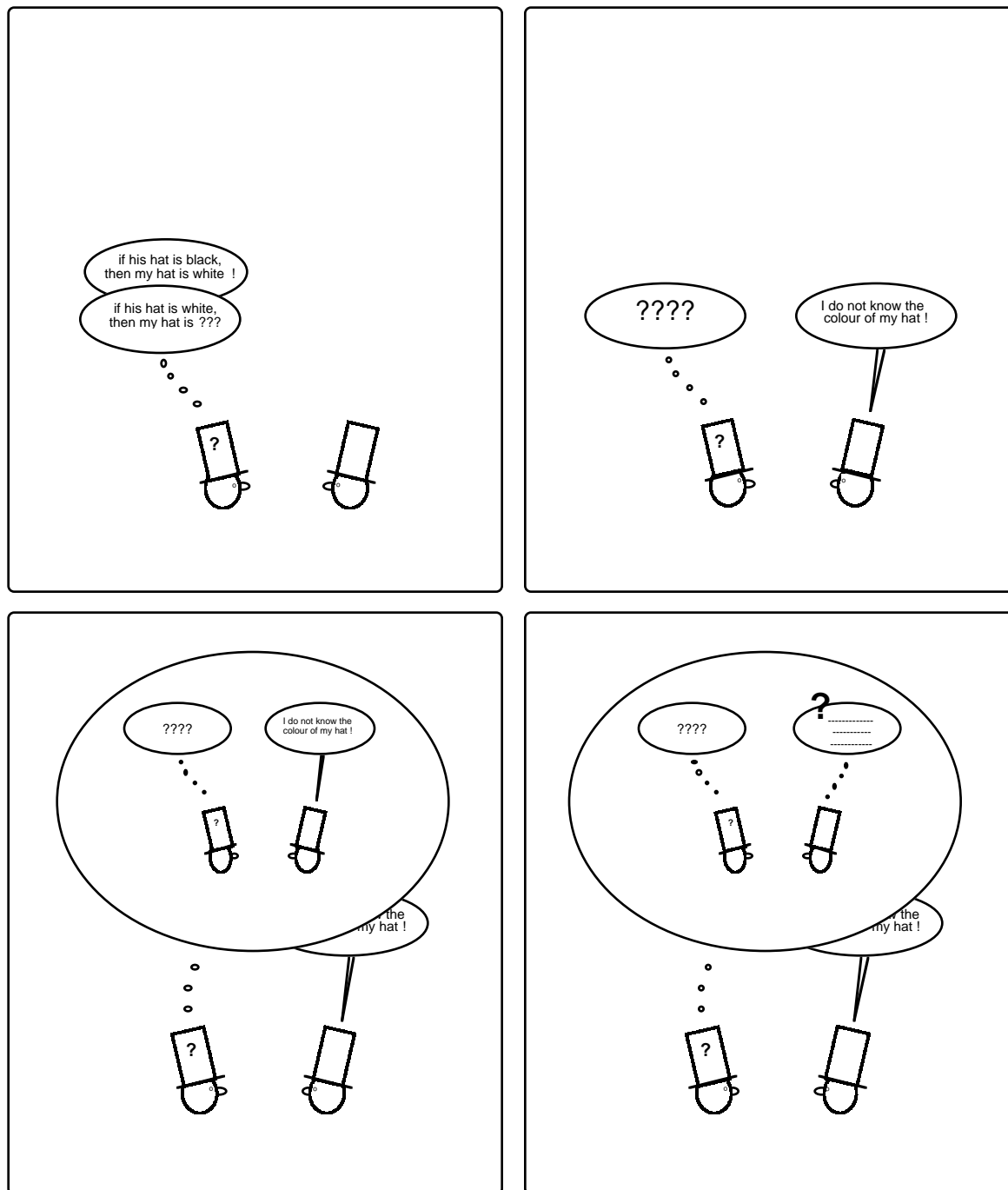


Figure 2 A reflective reasoning process: part 1

Wise man B not only reasons about its own state but also about A's reasoning processes. B reasons about the observations A could have made and the conclusions A would have drawn on the basis of these observations.

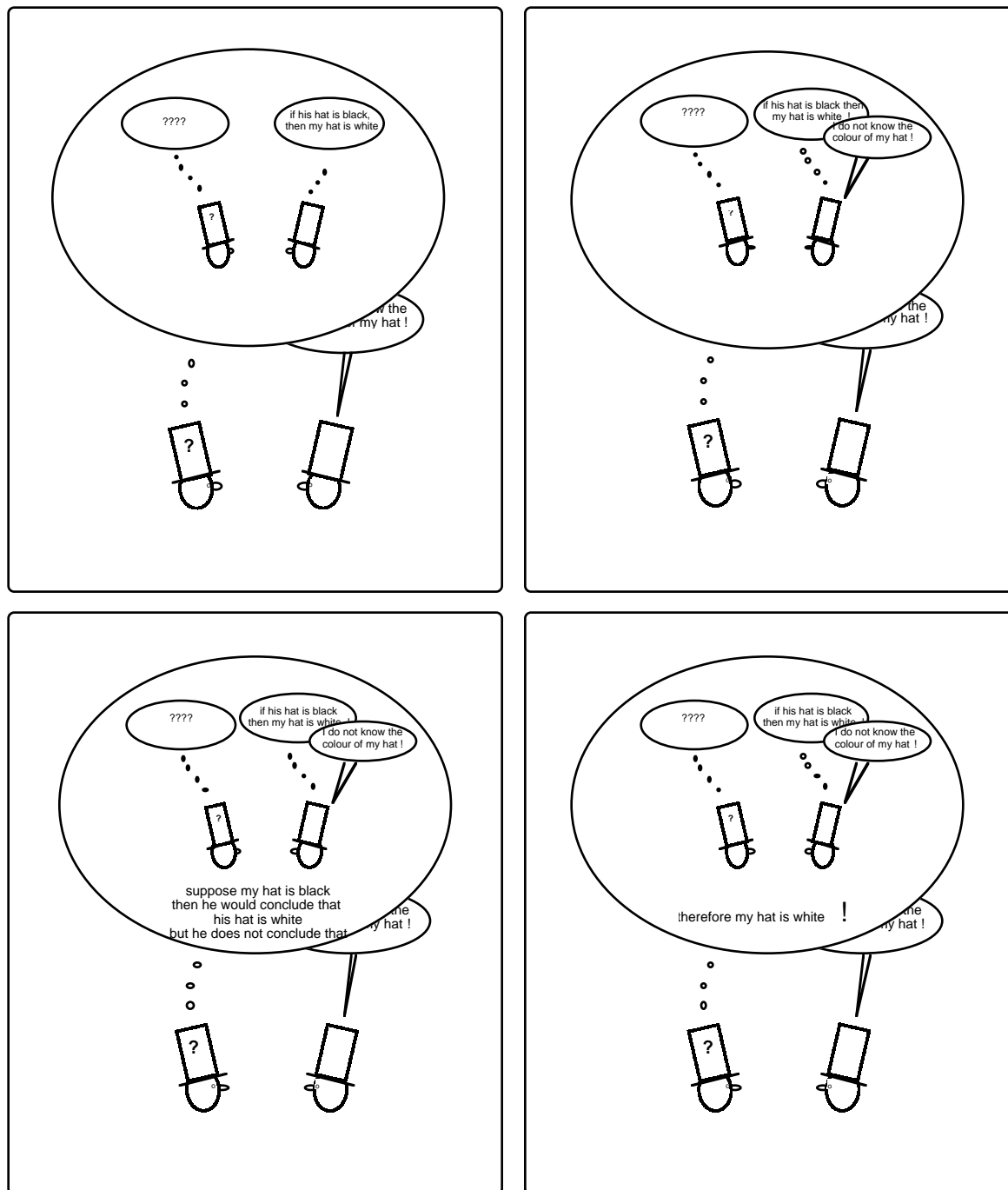


Figure 3 A reflective reasoning process: part 2

The generic agent model briefly presented in Section 2, is refined (specialised and instantiated) to model an agent that is able to perform the reflective reasoning about conflicts needed to solve this puzzle. The resulting model can be used to model both wise men: wise man A as Agent A and wise man B as Agent B. For the purpose of explanation, however, the model is described from the perspective of Agent B: the concepts and specifications involved are illustrated from Agent B's point of view. Reflective elements in B's reasoning include reasoning about:

- observations (e.g., the decision to observe the colour of A's hat),
- A's reasoning (which conclusions should A have reached on the basis of specific information),
- B's own information state and B's assumptions (e.g., about the colour of B's own hat),
- control of B's reasoning and actions, and
- communication of B with A (which information can and should A provide and when).

Note that for convenience sake quotes to denote an object-meta naming relation have been omitted.

4. A Specific Model of a Reflective Agent

In this section the generic agent model is refined. For three of the generic agent components more specific compositions are introduced (see Figure 4). The most illustrative generic component of an autonomous agent in this example is the component devised to perform the agent's specific task of determining the colour of its own hat.



Figure 4 Process abstraction levels for the reflective agent

4.1 Processes at different levels of abstraction

The processes modelled within the agent model for a reflective agent are depicted in Figure 4. All but one of the processes distinguished for the generic agent, are applicable to the reflective agent in this example. Due to the simplicity of the example, co-operation management has not included. As in the generic agent model the processes involved in controlling an agent (e.g., determining, monitoring and evaluating its own goals, plans) are the task of the component own process control.

Maintaining knowledge of other agents' abilities and knowledge (the task of the component maintain agent information) involves two processes: interpreting available information on other agents (the task of the component interpret agent info), and keeping this information up to date (the task of the component update agent info). Comparably, the processes involved in maintaining information on the external (material) world (the task of the component maintain world information) are two-fold: interpreting available information on the external world (the task of the component interpret world info), and keeping this information up to date (the task of the component update world info).

The reflective agent's specific task in this example domain (the process performed by the component determine hat colour) is to determine the colour of the other agent's hat. This task involves three subtasks for which three components are distinguished: determine method of information acquisition, evaluate process state and determine assumptions.

The component determine method of information acquisition is responsible for the choice of one of the three options: (1) observe the colour of agent A's hat hoping this will provide the information required, (2) communicate with agent A on A's conclusions concerning the colour of agent A's own hat and (3) make assumptions on the colour of his own hat and reason about the conclusions A should have drawn.

The component evaluate process state determines the results of having tried one of the methods: whether the method provided the colour of agent B's hat. To be able to reason about the results of the analysis process (the task of the component analyse process state) a separate component cwa analyse process state is needed.

The component determine assumptions determines which assumptions to make during reasoning. To this purpose determine assumptions first generates a possible assumption (the task of determine assumption's component generate assumptions). It then evaluates this assumption by reasoning about the consequences of the assumption (derived by the component interpret agent info in the component maintain agent info): the task of determine assumptions' second component, validate assumptions.

4.2 The processes at a lower level of abstraction

In this section the interface knowledge structures for each of the refinements of the top-level processes distinguished above, are presented together with the applicable (meta-)level of the knowledge structures with respect to the encompassing component (note that this is not the level within each of the component's themselves), and task control knowledge.

Refinement of the agent specific task: determine hat colour

The component evaluate process state receives three types of information:

1. information on the agent's own observations (from the component own process control),
2. information on conclusions agent A has reached and communicated (from the component manage agent interaction), and

3. information on the best assumption (if the component determine assumption's component generate assumptions has been able to make a best assumption).

On the basis of this information the component analyse process state determines the state of the problem solving process (e.g., whether observations have been made, whether a definite conclusion on the colour of the hat can be drawn). The knowledge with which the state of the process is determined includes both knowledge on which positive conclusions can be based, and knowledge on which negative conclusions can be based. Positive conclusions on the state of the process can be drawn, given that information has been acquired from observation, communication and/or assumption determination. Negative conclusions are based on the lack of positive conclusions; they are drawn by a closed world assumption on the output atoms, explicitly specified at a higher (third) meta-level in the component cwa analyse process state.

The information on the state of the reasoning process is transferred from the component evaluate process state to the component determine method of information acquisition. The specifications for the knowledge structures used within the component evaluate process state are shown below for the components analyse process state and cwa analyse process state . The knowledge structures for the component evaluate process state, are comparable.

component analyse_process_state

input atoms:

known_to_me_based_on_obs(hat_colour(A, C:Colour))	(** meta-level 1 **)
communicated(A, concludes(A, hat_colour(A, C:Colour)))	(** meta-level 2 **)
communicated(A, cannot_reach_a_conclusion(A))	(** meta-level 2 **)
best_assumption(observed(A, hat_colour(I,C:Colour)))	(** meta-level 2 **)

output atoms:

performed(observation)	(** meta-level 2 **)
performed(communication)	(** meta-level 2 **)
performed(assumption)	(** meta-level 2 **)
colour_known	(** meta-level 2 **)

knowledge base:

```
if known_to_me_based_on_own_obs(hat_colour(A, C:Colour))
then performed(observation) ;

if communicated(A, X:Comms)
then performed(communication) ;

if best_assumption(observed(A,hat_colour(I, C:Colour))
then performed(assumption) ;

if best_assumption(observed(A,hat_colour(I, C:Colour))
then known_to_me_based_on_comm(hat_colour(I, C:Colour)) ;

if known_to_me_based_on_own_obs(hat_colour(I, C:Colour))
then colour_known ;

if known_to_me_based_on_comm(hat_colour(I, C:Colour))
```

```
then colour_known
```

```
component cwa_analyse_process_state
```

```
input atoms:
```

```
true(X:OA) (** meta-level 3 **)
```

```
output atoms:
```

```
to_assume(X:OA, false) (** meta-level 3 **)
```

```
knowledge base:
```

```
if not true(X:OA)  
then to_assume(X:OA, false) ;
```

Based on the status information provided by evaluate process state the component determine method of information acquisition determines which method to follow: observation, communication or assumption. The conclusions of this component are transferred to the output interface of determine hat colour.

```
component determine_method_of_information_acquisition
```

```
input atoms:
```

```
performed(observation) (** meta-level 2 **)  
performed(communication) (** meta-level 2 **)  
performed(assumption) (** meta-level 2 **)
```

```
output atoms:
```

```
method of acquisition(observation) (** meta-level 2 **)  
method of acquisition(communication) (** meta-level 2 **)  
method of acquisition(assumption) (** meta-level 2 **)
```

```
knowledge base:
```

```
possible_method_of_acquisition(observation);  
possible_method_of_acquisition(communication);  
possible_method_of_acquisition(assumption);  
  
prior_to(observation, communication)  
prior_to(communication, assumption)  
  
if not performed(obs)  
then selected_method_of_acquisition(obs) ;  
  
if possible_method_of_acquisition(X)  
and possible_method_of_acquisition(Y)  
and performed(X)  
and not performed(Y)  
and prior_to(X, Y)  
then selected_method_of_acquisition(Y) ;
```

The component `determine assumptions` receives explicit information on the agent's lack of knowledge of A's observations (the truth value `false` for the input atom `known_to_me(observed(A, hat_colour(I, C:Colour)))` from the input interface of the component `agent specific task: determine hat colour` (which it, in turn, has received from the component `own process control`). In addition, `determine assumptions` receives information on A's conclusions on its own hat colour (received from the component `manage agent interaction`), and information that the assumed observations of A on the agent B's own hat colour, are contradictory. Based on this input information the component `generate assumptions`, generates both possible assumptions (which are transferred to the component `validate assumptions and maintain agent information`) and best assumptions (which are transferred to the output interface of the component `determine assumptions`, and from there to the component `evaluate process state`).

component `generate_assumptions`

input atoms:

```
communicated(A, concludes(A, hat_colour(A, C: Colour)))           (** meta-level 2 **)
known_to_me(observed(A, hat_colour(I, C:Colour)))                (** meta-level 2 **)
contradictory(observed(A, hat_colour(I, C:Colour)))              (** meta-level 2 **)
```

output atoms:

```
possible_assumption(observed(A, hat_colour(I, C:Colour)))       (** meta-level 2 **)
best_assumption(observed(A, hat_colour(I, C:Colour)))           (** meta-level 2 **)
```

knowledge base:

```
if      communicated(A, concludes(A, hat_colour(A, white)))
and    not known_to_me(observed(A, hat_colour(I, white)))
then   possible_assumption(observed(A, hat_colour(I, white))) ;

if      communicated(A, cannot_reach_a_conclusion(A))
and    not known_to_me(observed(A, hat_colour(I, black)))
then   possible_assumption(observed(A, hat_colour(I, black))) ;

if      contradictory(observed(A, hat_colour(I, black)))
then   best_assumption(observed(A, hat_colour(I, white))) ;

if      contradictory(observed(A, hat_colour(I, white)))
then   best_assumption(observed(A, hat_colour(I, black))) ;
```

Within the component `maintain agent information` a possible assumption with respect to observations on A's hat colour is transferred to the component `interpret agent info`, in which the conclusions A would draw on the basis of this assumption are derived. This information is transferred to the input interface of the component `agent specific task: determine hat colour`, which in turn, transfers this information to the input interface of the component `determine assumptions`. The component `determine assumptions` also receives information on A's communication with respect to its own conclusions with respect to its own hat colour from the component `manage agent interaction`. Both the information on the conclusions A would have drawn if A had observed specific assumed facts (the possible assumption) and the information on A's communicated

conclusions with respect to its own hat colour are transferred to the component validate assumptions. The component validate assumptions also receives information about the possible assumption directly from the component generate assumptions. The component validate assumptions determines whether these conclusions on the expected conclusions of A contradict the conclusions A actually has drawn (and communicated) on the colour of A's own hat. This information on the existence of a contradiction is transferred to the component generate assumptions.

component validate_assumptions

input atoms:

```
communicated(cannot_reach_a_conclusion(A))           (** meta-level 2 **)
communicated(A, concludes(A, hat_colour(A, C; colour))) (** meta-level 2 **)
expected(concludes(A, hat_colour(A, C: colour)))    (** meta-level 2 **)
expected(cannot_reach_a_conclusion(A))              (** meta-level 2 **)
possible_assumption(observed(A, hat_colour(I, C:Colour))) (** meta-level 2 **)
```

output atom:

```
contradictory(observed(A, hat_colour(I, C: colour))) (** meta-level 2 **)
```

knowledge base:

```
if      communicated(cannot_reach_a_conclusion(A))
and    expected(concludes(A, hat_colour(A, white)))
and    possible_assumption(observed(A, hat_colour(I, black)))
then   contradictory(observed(A, hat_colour(I, black))) ;

if      communicated(A, concludes(A, hat_colour(A, white)))
and    expected(cannot_reach_a_conclusion(A))
and    possible_assumption(observed(A, hat_colour(I, white)))
then   contradictory(observed(A, hat_colour(I, white))) ;
```

Task control of determine hat colour

Activation of determine hat colour, in combination with activation of the links which can provide the information required by determine hat colour, is specified by agent B's task control. Task control of determine hat colour determines which internal components and links to activate. Activation of evaluate process state is done in combination with activation of the incoming links. If the final evaluation criterion depicting success of determination of colour of the hat, is reached then the task of determine hat colour is fulfilled. If, however, the evaluation criterion that specifies that one or more conclusions concerning previous performance have been reached, succeeds, task control specifies that the component determine method of information acquisition is to be activated, together with the related links. Based on the success or failure of the evaluation criteria, task control determines which component and links to activate next. If, for example, the evaluation criterion observations required, is successful, then determine hat colour sends a request to manage world interaction to make observations in the external world. If, for example, the evaluation criterion assumptions required is successful, then another component of determine hat colour, namely determine assumptions, is activated. The component determine assumptions, in turn, activates one of its components, based on its own task control knowledge.

Refinement of the component: maintain agent information

The component maintain agent information has two components: update current agent information, which stores information on other agents, and interpret agent info, which interprets the available agent information. The first component only stores and updates information, it does not reason. To interpret agent information the component interpret agent info has knowledge with which it can reason about the other agent. In the wise men example the knowledge specifies how the other agent can reason; it gives an explicit representation of A's deduction system and A's knowledge. For example, part of the knowledge on A is the explicit meta-statement that if a fact X is derivable by A and A has knowledge that X implies Y, then Y is derivable by A (modus ponens). Agent B uses this knowledge of A to reason about A's reasoning, as shown in the knowledge base of B's component interpret agent info specified below. In this knowledge base the meta-statement rule(A, X, Y) denotes that A has the knowledge that X implies Y. The notation [X,Y] is interpreted as the conjunction of the statements X and Y, and derivable(A, X) denotes that A is able to derive statement X. The (meta-)fact observed(A, X) states that fact X is observed in the external world by A. Note that the I in this knowledge base refers to A, because it refers to A's own knowledge.

component interpret_agent_info

input atoms:

observed(A, hat_colour(B,C:Colour)) (** meta-level 2 **)

output atoms:

derivable(A, X) (** meta-level 2 **)

knowledge base:

rule(A, hat_colour(B,black), hat_colour(I,white)) ;

if observed(A, X)
then derivable(A, X) ;

if derivable(A, X)
and rule(A, X,Y)
then derivable(A, Y) ;

if derivable(A, X)
and derivable(A, Y)
then derivable(A, [X,Y]) ;

Refinement of the component maintain world information

The component maintain world information has two components: update current world information, which stores information on the world, and interpret world info, which interprets the available world information. Comparable to the composition described in the previous section, the first component only stores and updates information and does not reason. To interpret world information the component interpret world info has knowledge with which it can reason about the

world. This knowledge is used by B to draw conclusions from information he has obtained from observation of A's hat colour: if B observes a black hat, then his own hat is white.

component interpret world info

input atoms:

hat_colour(A, C:Colour) (** object level **)

output atoms:

hat_colour(I, C:Colour) (**object level **)

knowledge base:

```
if      hat_colour(A,black)
then   hat_colour(I,white) ;
```

Note that both A and B can observe part of the external world, but that they observe *different* parts. This difference is expressed in the specifications by the different information links defined between the external world and the agents. The difference is also mirrored in the input information types of the two agents.

5. Analysis of an example reasoning process in the model

First the agent B generates a number of possible information acquisition strategies. As the agent is assumed to only perform one of these strategies at a time this entails a conflict between the strategies. This conflict is resolved by explicit knowledge that as long as observation has not been performed, it should be selected; otherwise if observation has been performed and communication has not, then (because the strategic knowledge specifies that observation is performed prior to communication), communication is selected, and so on.

Next, the agent deliberately aims at introducing another conflict, by making an assumption that may quite well turn out to be false: it assumes that its own hat is black (component generate assumptions). The subsequent process aims at falsification of the assumption. As a first step the deductive consequences of the assumption are derived, taking into account a model of the reasoning process of the other agent: the consequences of agent B's own hat being black, are that this would be observed by agent A and that agent A would draw the conclusion that its own hat is white, and communicate this. Conflict detection occurs when B compares the deductive consequences of B's assumption to observation results; the results contradict each other. After detection of the conflict (within the component validate assumptions), and determination of the assumption from which the conflict originates, the conflict is resolved within B's component generate assumptions (by blaming the assumption for the conflict, and making the opposite assumption).

6. Discussion

This chapter has addressed meta-reasoning and reflection in the context of conflict management, in particular with respect to agent abilities and architecture. An example of a reflective agent, based on a generic agent model, has been presented within which reasoning about (1) observation and agent interaction (2) an agent's own information state and reasoning processes (3) other agents' information states and reasoning processes, and combinations of these types of reflective reasoning, are explicitly modelled. To illustrate the transparency of the structure, partial specifications of the wise men's puzzle have been presented within which components at different meta-levels of knowledge and reasoning) are distinguished. The agent is able to deliberately introduce a conflict, by making an assumption that is expected to turn out to be false. Conflict management is performed in a process aimed at falsification of the assumption. Conflict detection occurs when the the deductive consequences of the assumption are compared to observation results. After detection of the conflict the conflict is resolved by assigning a higher priority to observation results than to assumptions.

In the model presented in this chapter the dynamics of the combined pattern of reasoning, observation and communication is modelled: the specification explicitly expresses the strategy with which the problem is approached. Specification of the problem, abstracting from the dynamics, would also have been possible. However, in that case, either strategic knowledge to guide the problem solving has to be added at the implementation level, or a theorem prover or other program would need to search for the solution in the space of all possible alternatives. In the former case an implementation independent description of the dynamics of the system would be lacking. In the latter case the search process may be inefficient. Moreover, the system behaviour differs significantly from the way in which human agents most often approach problems such as this: human agents use strategic knowledge to guide the search.

The modelling approach adopted in this paper distinguishes components reasoning at different levels in all cases where semantically distinct meta-levels can be found. An advantage of this approach is that the model has a rich structure with different constructs for entities that are semantically different. As a result a more complex problem may require additional meta-levels. This may be considered to be the price that has to be paid for the richer structure. An alternative approach is that all meta-levels are encoded in the highest meta-level. The price that is paid in this case is that the finer semantical distinctions between the different meta-levels found in practice are not explicitly represented.

References

- Attardi, G., Simi, M. (1994). Proofs in Context, In: L. Fribourg and F. Turini (eds.), Logic Program Synthesis and Transformation-Meta-Programming in Logic, Proc. of the Fourth International Workshop on Meta-Programming in Logic, META'94. Springer Verlag, Lecture Notes in Computer Science, vol. 883.
- Brazier, F.M.T., Dunin Keplicz, B.M., Jennings, N.R. and Treur, J. (1995). Formal Specification of Multi-Agent Systems: a Real World Case, In: V. Lesser (Ed.), Proc. First Int. Conference on Multi-Agent Systems, ICMAS-95, MIT Press, pp. 25-32. Extended version in M. Huhns, M. Singh, (eds.), International Journal of Cooperative Information Systems, vol. 6, special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems, pp. 67-94

- Brazier, F.M.T., Dunin-Keplicz, B.M., Treur, J., Verbrugge, L.C. (1998). Modelling Internal Dynamic Behaviour of BDI agents. In: A. Cesto & P.Y. Schobbes (eds.), Proceedings of the Third International Workshop on Formal Models of Agents, MODELAGE'97, Lecture Notes in AI, Springer Verlag. In press, 1998, pp. 21.
- Brazier F.M.T., Jonker C.M., Treur J. (1996). Modelling Project Coordination in a Multi-Agent Framework. In: Proc. of the Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WET ICE'96, IEEE Computer Society Press, Los Alamitos, 1996, pp. 148-155.
- Brazier, F.M.T., Jonker, C.M., and Treur, J. (1997). Formalization of a cooperation model based on joint intentions. In: J.P. Mueller, M.J. Wooldridge, N.R. Jennings (eds.), Intelligent Agents III, Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, ATAL'96, Lecture Notes in AI, volume 1193, Springer Verlag, Berlin, 1997, pp. 141-155.
- Brazier F.M.T., Langen P.H.G. van, Treur J. (1995). Modelling conflict management in design: an explicit approach. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, (AIEDAM), Special Issue on Conflict Management in Design (I.F.C. Smith, ed.), Vol. 9, No. 4, 1995. pp. 353-366.
- Brown, D.C. (1998). Modelling Conflicts Between Agents in a Design Context. In H.J. Mueller and R. Dieng (eds.) Computational Conflicts: Conflict Modeling as a Primary Design Technique for Distributed Intelligent Systems.
- Castelfranchi, C. (1995). Self-awareness: notes for a computational theory of intrapsychic social interaction. In: G. Trautteur (ed.), Consciousness: Distinction and Reflection, Bibliopolis, pp. 55-80.
- Castelfranchi, C. (1998). Conflict Ontology. In H.J. Mueller and R. Dieng (eds.) Computational Conflicts: Conflict Modeling as a Primary Design Technique for Distributed Intelligent Systems.
- Cimatti, A., Serafini, L. (1995). Multi-agent Reasoning with Belief Contexts II: Elaboration Tolerance, In: V. Lesser (ed.), Proceedings of the First International Conference on Multi-Agent Systems, ICMAS-95, MIT Press, pp. 57-64
- Hoek, W. van der, Chr. Meyer, J.-J. and Treur, J. (1994). Formal Semantics of Temporal Epistemic Reflection. In: L. Fribourg and F. Turini (ed.), Logic Program Synthesis and Transformation-Meta-Programming in Logic, Proceedings of the Fourth International Workshop on Meta-Programming in Logic, META'94. Springer Verlag, Lecture Notes in Computer Science, vol. 883, pp. 332-352.
- Maes, P. and Nardi, D. (eds.) (1988). Meta-level architectures and reflection, Elsevier Science Publishers.
- Weyhrauch, R.W. (1980), Prolegomena to a theory of mechanized formal reasoning, Artificial Intelligence, Volume 13, pp. 133-170.
- Wooldridge, M.J., and N.R. Jennings (1995). Intelligent Agents: Theory and practice. In: Knowledge Engineering Review, 10(2), 1995, pp. 115-152.