A Condensed Roadmap of Agents-Modelling-Agents Research

Olayide Olorunleke, Gordon McCalla
ARIES Lab, Computer Science Department
University of Saskatchewan, Canada
{oto033, mccalla}@cs.usask.ca

Abstract

In this paper we present a survey that classifies the types of information contained in agent models and describes various techniques that have been used in the past to maintain up-to-date agent models. From this discussion we present a model that captures the agent modelling ideas. The presented model is then used to describe the major issues in agent modelling research, and to suggest some future opportunities and directions for agents-modelling-agents research.

1 Introduction

An agent is often implemented as a goal-driven active component that is embedded in an environment from which it receives sensory data and based on the sensed data (or state of the environment) makes decisions about which action to carry out in the environment in order to achieve its goal(s). In the simplest case, the environment contains only one such agent and this embedded agent is always able to sense the necessary data from the environment that it requires to make a decision on which action to carry out to ensure that its goals are met. An example of such a simple case is a thermostat agent which is given the goal of maintaining a set temperature level (*desired_temp*) for a room. In this case, the agent's sensor senses the current temperature (*room_temp*) of the room and based on this value the agent determines which of its possible actions to carry out in order to maintain the desired room temperature. These actions, for example, could include elements from a set of actions such as {*do_nothing, turn_heater_on, ..., etc*}. The agent selects the action from this set that it expects to help achieve its goal sometime in the future. This action selection assumes that the agent possesses knowledge about the effect of its actions on the state (i.e. *room_temp*) of the environment. This knowledge can be encoded for the agent by the designer as a set of *condition-action* rules, such as,

{
if *room_temp* < *desired_temp* **then** *turn_heater_on,*
if *room_temp* = *desired_temp* **then** *do_nothing*,
.
.
.
Etc.
}

It is clear, therefore, that the selected action is driven by what the agent believes to be true about the current state of the environment. This belief is captured by the *room_temp* variable, and is thus often called the *environment model* or *world model*.

Consider a more complex case in which the environment contains a set T of *n* thermostat agents, labeled $t_1, t_2, .., t_n \in T$, and in which each t_i's ($1<=i<=n$) goal is to maintain the same *desired_temp* for the room (possibly for redundancy). The condition-action rules for each t_i in this case can be expressed as

{
If *room_temp* < *desired_temp* **and** *t.action*=*do_nothing* $\forall t \in T - \{t_i\}$ **then** *turn_heater_on,*
if *room_temp* = *desired_temp* **then** *do_nothing*,
.
.
.
Etc.
}

where *t.action* $\forall t \in T - \{t_i\}$ refers to the action that each of the other agents minus the agent making the decision has selected. The belief about the state of the world (i.e. world model) is now captured by variables *room_temp* and *t.action* $\forall t \in T - \{t_i\}$. That is, the world model now contains information about the environment and information about the other agents in the environment. The information stored about the other agents in an agent represents what that agent believes about the other agents and are called *agent models.*

In this paper, we present a survey of the processes involved in the building and maintenance of these agent models. We present these processes in a unified model that allows for communicating the ideas involved in the modelling of other agents. The goal of the model presented is to allow for communication (within the research community), discussion about the issues and solutions proposed in general for agent modelling, and to propose possible new opportunities for agent modelling research.

2 Agent Modelling

Agent modelling encompasses the processes, techniques, or algorithms that can be used for the building and maintenance of the knowledge stored about other agents (i.e. agent models). Before presenting various examples of these processes it is necessary to discuss and classify the types of information that are usually stored in an agent model. Such a classification will allow us to describe and understand the differences between the inputs and outputs of the processes that are involved in agent modelling.

2.1 Contents of Agent Models

In the previous example with *n* thermostat agents, the agent models only contained one attribute that is assigned to each agent, that is, the action that each agent has chosen to carry out (*t.action*). In general, the agent model *t* for any agent usually contains a combination of the following types of information:

1. Observable features

   This type of information includes any information about the agent being modeled (i.e. the *modelee*) that can be detected by the sensors of the modeler such as *t.orientation*, *t.location*, etc.

2. Internally held beliefs

   Internally held beliefs refer to the beliefs held by the modelee. For example *t.room_temp* refers to agent *t*'s belief about the *room_temp*. These types of information are generally not observable via the agents' sensors. This could also contain beliefs such as what the modeler believes the modelee believes the modeler believes (e.g. $t_{modelee}.t_{modeler}.room\_temp$), and so on. This type of modelling is referred to as *recursive modelling* [Gmytrasiewicz *et al.*, 1991].

3. Predictions relating to the agent being modeled

   Predictions refer to the expectations of the modeler about the modelee that are expected to be true at some time in the future.

4. Other properties that can be attributed to each agent.

   This type of information includes properties than can be attributed to other agents such as capabilities, reputation, sensor range, plans, goals, policies, etc.

At any point in time, the contents of an agent model, which are made up of a combination of the information types given above, can be classified into two classes; *class A* and *class B*.

2.1.1 Class A - Observations

This class contains the information that can be sensed directly from the agent's sensors at that point in time. In accessible environments – that is, environments in which the sensors are always able to sense the entire state of the world – it is clear that all the observable features in type-1 will fall into this class since their values can be sensed directly from the environment. In inaccessible environments, however, the agent's sensor's position or orientation may affect the range of the environment's states that can actually be sensed at a point in time. For example, a camera can only record objects within its view range; therefore, other objects that are not within its view range will not be considered as observable and will not belong to class A, even though they can be picked up by cameras in general. These objects that are not currently being picked up by the sensors are classified to be in class B.

2.1.2 Class B - Inferences

Those things that cannot be currently sensed are classified into this class. The inability to sense the values assigned to these features implies that the modeler has to *guess*, *infer,* or *compute* what the current values should be; based on the currently sensed information and prior knowledge of the relationships that exist between the sensed features and the non-sensed ones, the history of values that have been assigned in the past to the non-sensed features, and the dynamics of the non-sensed features. Information-types 2, 3, and 4 usually fall into this class since they are generally not sense-able by the agent's sensors. In inaccessible environments it is possible that some type-1 information can fall under this class even though they are sense-able, but are not currently being sensed.

2.2 Agent Modelling Processes, Algorithms, and Techniques

Every agent modelling process, algorithm, or technique attempts to maintain accurate and up-to-date values in the agent models being stored. This involves the updating of both the class A and class B information in the agent models. Updating the class A information in the models is a pretty straight-forward task since it only involves getting the observations from the sensors. This also often serves as the first step in most modelling techniques. Once this update of class A information is completed, the modelling process then continues to use the class A information to infer values for the unobservable class B information. For example, the REal-time Situated Least-commitments (RESL) algorithm [Kaminka *et al.*, 1998] shown in Figure 1 depicts this process.

As shown in the figure, the RESL algorithm begins by getting the current observations – i.e. the class A information – about the agent being modeled. The remainder of the algorithm then attempts to infer the unobservable class B information (plans and beliefs in this case) about that same agent. To achieve this, RESL relies on the use of a plan recognition network that provides a link between the observations made about the agent and the internal plans within the agent that could produce those observations. Once the plans have been flagged, RESL infers the agent's internal beliefs by ascribing the preconditions that must be true for those inferred plans to be true and the termination conditions that must be true for those plans that are inferred to have just been terminated.

In general, updating the class B information from the class A information alone is not possible. To carry out this task, agent modelling algorithms usually use additional knowledge that provides a means of connecting the observations to the unobservable. In the RESL example, this additional knowledge is provided by the plan recognition network which is assumed to be a reactive plan [Firby, 1987] hierarchy that controls each agent. Huber and Simpson [2004] also use a plan recognition network in maintaining values for the unobservable. Other means that have been used to encode this additional knowledge include use of plan libraries [Ferguson, 1992; Geib and Harp, 2004], dynamic Bayesian networks [Hamid *et al*., 2003; Rybski and Veloso, 2004; Kaminka et al., 2002], hidden markov models [Bui *et al*., 2002; Feldman and Balch, 2004], stochastic processes [Lerman and Galstyan, 2004], behavior graphs [Kaminka and Avrahami, 2004], and probabilistic state-dependent grammars [Pynadath and Wellman, 2000].

```
RESL ( plan recognition network, modeled agent )
{
get observations about agent
// primitive matching
for each operator that has a set of expected observations:
    1. attempt to match observations to expectations
    2. If succeed, flag operator as matching successfully
    3. If fail, flag operator as failing to match
// propagate matching
for each operator that is flagged as matching
successfully,
    • flag its parents as matching successfully
for each operator whose children (all of them) are flagged
as failing to match,
    • flag it as failing to match
// preconditions
for each operator that is flagged as matching
successfully,
    • flag its associated set of preconditions as possibly
    true (the agent possibly believes this set of
    preconditions)
// termination conditions
for each operator that has just stopped matching
successfully,
    • flag its associated set of termination conditions as
    possibly true
}
```

Figure 1: The RESL algorithm

The common thread in the above examples is that they represent a paradigm that assigns the responsibility of inferring the class B data to the modeler. This paradigm is referred to by Grosz and Kraus [1999] as *active monitoring*, and an example of such active monitoring is via *plan recognition* [Kautz and Allen, 1986; Carberry, 1990; Huber and Durfee, 1995; Intille and Bobick, 1999]. A common assumption in plan recognition is that the actions carried out by an agent are controlled by a mental state that drives the synthesis of plans that the agent intends to follow in order to achieve its goals. It is also assumed that the modeler is aware of this mechanism (or has learned this mechanism through a process of reinforcement learning for example) for decision making which the modelee uses, and hence the ability to observe the selected actions allows the modeler to infer the mental state from which they result.

Obviously, the types of information we have classified into class B go beyond plans and beliefs, and therefore plan recognition alone cannot be used to maintain values for these. Another example of an active monitoring approach that tries to maintain up-to-date values for the type-1 information that is a subset of the class B information is the *predictive memory* approach [Bowling *et al*., 1996]. The predictive memory approach maintains the state information – such as the position, velocity, direction, etc – of other agents (within the simulated RoboCup domain [Kitano *et al*., 1997]). Being that the type of information that the predictive approach addresses are usually sense-able (i.e. can be picked up by sensors), the need for a process to maintain values for them only arises in inaccessible environments; hence, their being classified into class B. This predictive memory approach works by storing an additional value – for each type-1 information – that describes the confidence in the accuracy of the current value assigned to it. This is achieved by storing the confidence as a probability that represents how confident the agent is in the accuracy of the value currently assigned to that feature. At every point of memory update, the predictive memory approach updates the values of those features of agents that are directly observable from the sensory information obtained from the environment (class A) – a step similar to that shown in RESL. Additionally, the probabilities attached to these values are set to 1.0. For those values that cannot be obtained from sensory input (class B), the predictive memory approach uses two phases to achieve the update.

The first phase – called the *internal phase* – considers those changes that should occur based on the agent's own most-recent actions. For example, if the agent's last action was a turn by angle *a*, then it is possible to update the positions of the other agents by correcting for this turn (since they are stored as relative positions).

The second phase – which is called the *external phase* – in the update of the data stored in memory about the states features applies particularly to mobile objects (the ball and other agents in RoboCup). The assumption is that mobile objects tend to continue in their direction of motion and thus – even when out of view – there is a short-lived certainty that the objects will continue moving in that direction. Thus the unseen mobile object's positions are updated using their last-observed velocities and positions. To account for this guess, the probability values attached to these unseen data items are reduced – by multiplying by a decay factor (e.g. 0.9) – to reflect a reduction in the certainty in the accuracy of the maintained data. Additionally, the last observed velocity that was used to update the object's position is also decayed, to reflect the possibility that moving objects eventually slow down. A threshold can be used to determine at which point the update from memory is to be considered inaccurate. Bowling *et al.* used a threshold value of 0.5.

Thus, if a confidence decay of 0.9 per update cycle and a confidence threshold of 0.5 are used, then an object can remain unseen for 6 cycles – as shown in Table 1 – with relatively accurate values (or at least relatively confident values) still maintained about it. At the 7th cycle the object's confidence goes below the set threshold.

This approach, in a similar way to RESL and the other plan recognition approaches, also assumes some other knowledge which allows the connection from past values of the unobservable information to be used to estimate what the current values should be. In the case of RoboCup, the rules of the game and the physics driving the system allows this connection to be made. In general, applying the predictive memory approach is dependent on the domain, since it relies on the knowledge of the rules governing the dynamics of the system.

Table 1. This table shows how the confidence in the values associated with unseen objects is decayed. After the 7th cycle of not being seen an object goes below the threshold of confidence.

Time	object observed?	old confidence	decay factor	new confidence
t	yes	Any value	0.9	1
t+1	no	1	0.9	0.9
t+2	no	0.9	0.9	0.81
t+3	no	0.81	0.9	0.729
t+4	no	0.729	0.9	0.6561
t+5	no	0.6561	0.9	0.59049
t+6	no	0.59049	0.9	0.531441
t+7	no	0.531441	0.9	0.4782969

Another active monitoring approach adopted – called *ideal-model-based behavior outcome prediction* (IMBBOP) – by Stone *et al.* [2000] for predicting (the type-3 information) the actions that are expected of other agents within the RoboCup domain. IMBBOP works by predicting other agents' future actions in relation to the behavior considered optimal for them in any given situation. The prediction is carried out based on information that is readily available in the world and thus, the prediction that is made is not dependent on the agent being modeled, but on the world dynamics and the actions considered to be optimal by the modeler in that situation. In this regard of dependence on world situation and dynamics, IMBBOP can be classified as being a *focal point* approach that is used for coordinating the action predicted and the action which the modelee actually chooses to carry out. Essentially, *focal points* are defined as prominent solutions that stand out from a set of possible solutions [Schelling, 1963; Kraus and Rosenschein, 1992]. The optimal nature (i.e. prominent solution) of the predicted action is what makes it a focal point.

A second paradigm for maintaining values for the class B information in the agent models relies on communication between the agents. These communication-based approaches are referred to as *passive monitoring* approaches [Grosz and Kraus, 1999] or *report-based* approaches [Kaminka *et al.*, 2002]. This approach is particularly ideal for use in teams of agents that are assumed to share a common language and ontology, and can be assumed to be committed to each other. Through communication, therefore, the modeler receives the class B information from the other agents that know such information rather than using any of the active monitoring approaches described earlier. The information sent to the modeler could be in response to a query from the modeler or a proactive inference on the part of the other agents that the modeler needs such information. Finally, a third paradigm combines both the active and passive approaches for maintaining the class B information [Parker, 1993].

3 Example Applications of Agent Models and Agent Modelling Techniques

Agent models have been used for various purposes in multi-agent systems. In this section, we highlight a few of these.

Recognition and Critiquing

Agent models have been used in the recognition and provision of appropriate feedback to human agents during the problem-solving processes of the human agents. Examples include [Mengshoel and Wilkins, 1996].

Natural Language Discourse

Agent models have been used both in selecting appropriate response from a stream of observational inputs [e.g. Green and Lehman, 1996], and making decisions about what should be said, by whom, and when it should be said [e.g. Donaldson and Cohen, 1996].

Failure Detection and Recovery

Kaminka and Tambe [1997] used the contents of agent models for failure detection and recovery. This approach, called SOCFAD[1] (SOcial Comparison for FAilure Detection), uses the other agents as sources of information to which the modeler compares its own beliefs, plans, goals, etc with those of the other modeled agents, and then reasons about the differences observed in order to draw conclusions about the correctness of their own behavior.

Finding Helpers for Helpees

The I-Help project [Vassileva *et al.*, 2002] attempts, in an e-learning context, to match helpees with possible helpers based on the contents of agent models of both helpers and helpees.

3 A Model of Agent Modelling Research

In this section we present a model that summarizes the analysis of agents-modelling-agents research that we have outlined in section 2. The aim behind such a model is to serve as a tool in service of (1) correctly classifying and understanding how new research ideas fit into or complement other pre-existing modelling ideas, techniques, or

[1] Later expanded and called Socially Attentive Monitoring (SAM) in Kaminka and Tambe [1998].

processes and (2) understanding where new opportunities for agent modelling research lie. We do not claim that this model is the only such model that can summarize the ideas in agent modelling, but since to our knowledge no other such model exists, we believe that the presented model adds value to agent modelling research.

As shown in Figure 2, **Model$_{xy}$** refers to the model that agent x (shown in the bottom right corner of the figure) keeps about another agent y (not shown in the figure). The figure shows that the contents of the model are partitioned into the previously discussed class A and B partitions, with the class A partition consisting only of type-1 information, and the class B partition consisting of type-2,3, and 4 partitions with the possibility of also containing type-1 information in inaccessible environments.

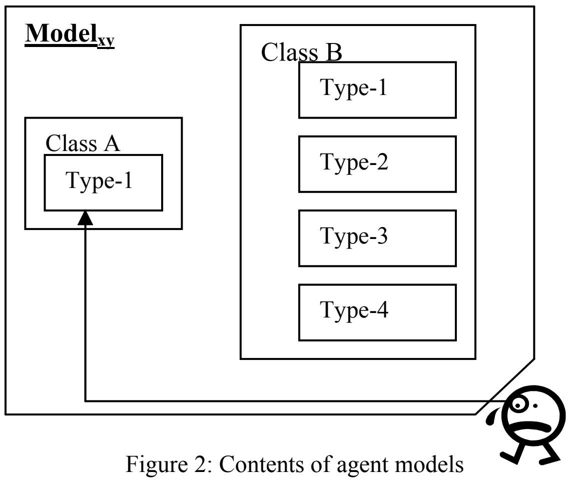

Figure 2: Contents of agent models

Maintaining up-to-date values for the class A data is easily achieved by wiring the agent's sensors (the circled eye in the diagram) to the class A information in the model.

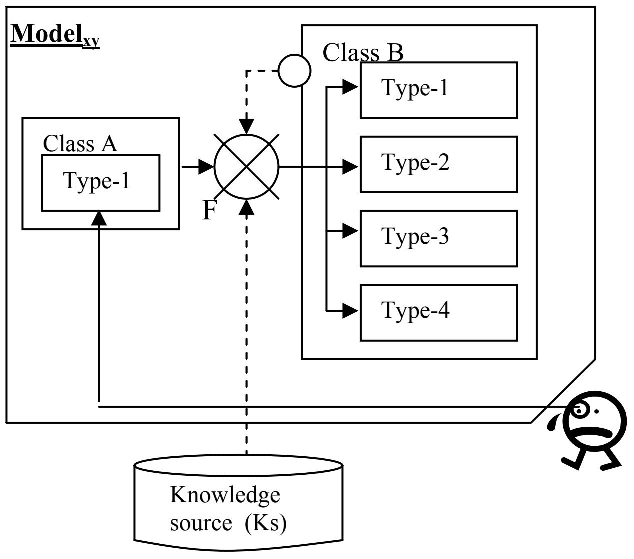

Figure 3: The active monitoring paradigm

For the class B data, which is unobservable, the active monitoring paradigm suggests, as shown in Figure 3, that a set of functions F should exist that take elements of class A, with the possibility of some external knowledge sources and elements of class B as input, and compute an element of class B. That is, if $f \in F$ then

$$f\text{: A} \times \text{Class A} \times [\text{Ks}] \times [\text{Class B}] \rightarrow \text{Class B}$$

where A is the agent being modeled and the input parameters in square brackets (i.e. Ks and Class B) are optional. The optional nature of these parameters is depicted as dotted arrows in Figure 3. The optional feedback connection (i.e.[Class B]) signifies the possibility of doing recursive modelling or making predictions arbitrarily far into the future. Any function that satisfies the above definition can be called an *active monitoring agent modelling technique*. Examples of such functions that we have already outlined earlier-on include: RESL, the predictive memory approach, and IMBBOP.

The passive monitoring paradigm, on the other hand (as shown in Figure 4), suggests that the class B data be updated using communication ideas such as asking other agents for information about the contents of class B, subscribing to such information, or proactively communicating such information.

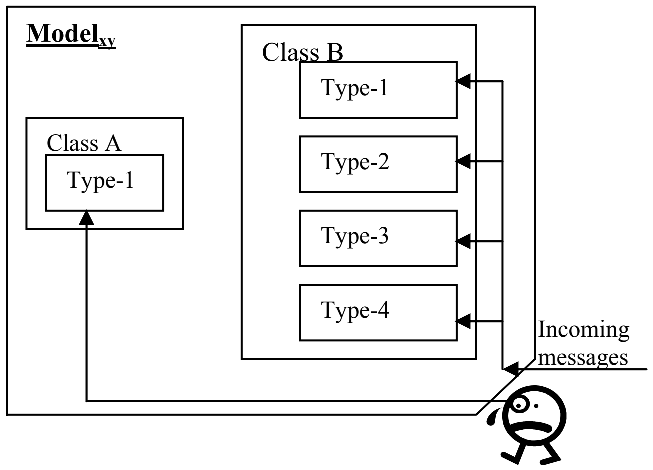

Figure 4: The passive monitoring paradigm

The model representing the third paradigm can be generated by combining figures 3 and 4 (This is trivial so the figure is not shown to save space). This paradigm allows for actively computing some elements of class B while some others are updated via communication links, or using the elements of class B received via communication to confirm the locally computed values assigned to them, and so on.

The external knowledge that is stored in the knowledge source (Ks) that is used in modelling other agents can be classified into two categories. The first category encompasses the knowledge/information that represents what is

known about the modelee on account of a priori knowledge of how the modelee selects its actions. For example, within the RoboCup domain, Stone [1998] refers to this as the *locker-room agreement*. Examples of such include the formation used by a team in which each agent's position on the pitch depends on the location of the ball. Hence, knowing the location of the ball is enough to infer where the other agents are (or are trying) to be.
The second category in a very similar way also encompasses the knowledge or information that is known a priori about the modelee that accounts for its action selection. The distinguishing factor between the two categories lies in the source of the knowledge. For the first category, the source of the knowledge is assumed to be the designer of the modelee and therefore applies usually to modelling teammates. On the other hand, the knowledge that falls into the second category is usually obtained by observing the behavior of the modelee and encoding the relationship between the observations and the actions selected by the modelee. Essentially the difference is that the knowledge is given in the first category (by the designer of the modelee) while the knowledge is learnt in the second category through a reinforcement learning process for example.

5 The Issues Involved

Since the contents of the agent models are used for action-selection, it is very important that the agent models contain accurate information. Inaccurate information in the agent models will result in the selection of actions that are inappropriate for the current context. The presence of any such inaccuracies in the models is referred to as (or results in) *delusion* [Olorunleke and McCalla, 2003; 2004a] or *model-entity discrepancy* [Ferguson, 1992]. A desirable property of any agent modelling technique, therefore, is that it minimizes the presence of delusion in the agent models. This means that sensors should return accurate data, active monitoring approaches should make accurate inferences, the passive approaches should communicate accurate data to each other, and the external knowledge sources should contain accurate information. Olorunleke [2002] demonstrates how reinforcement learning can be used to remove delusion in the information about agent capabilities (which fall into the type-4 information category). The use of reinforcement learning in this case is possible because of the assumption that agent capabilities do not change very often; therefore, there is an opportunity to learn the capability of each agent and use this information in the assignment of tasks. Avoiding the spread of delusions from agent to agent is particularly important when passive modelling approaches are used. The reason for this is that an error in one agent's model can be easily passed to other agents via communication, and if such information is always believed then delusion will spread easily through the system. Three strategies are discussed in [Olorunleke and McCalla, 2004a] to solve this problem, with general guidelines for maintaining delusion-free models also given.

To illustrate the second important requirement of agent modelling techniques consider the situation in Figure 5 in which the modeler now maintains models for *n* other agents.

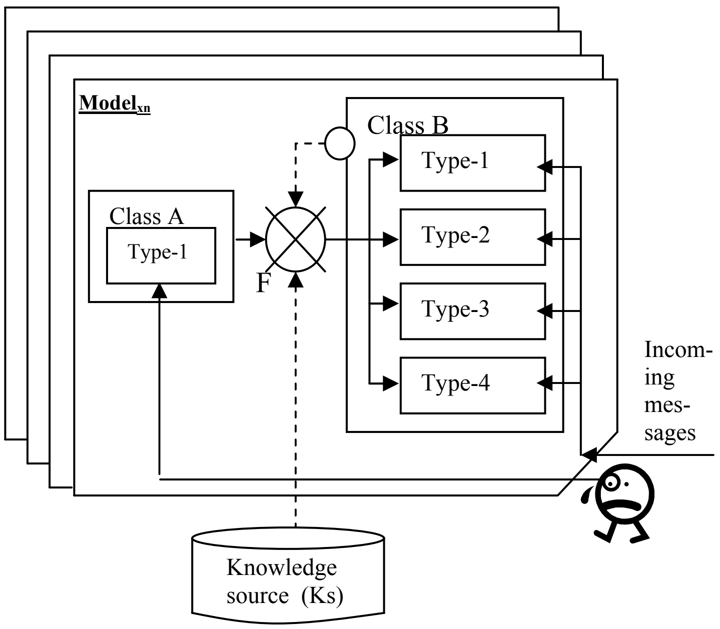

Figure 5: Multiple (*n*) agent models

As shown in the figure, agent x must now use the agent modelling techniques that it has available to it for each of its models at every update opportunity. This could be achieved using an algorithm such as:

```
Update_Models()
{
  for i = 1 to n
  {
    curAgent = Model$_{xi}$

    $\forall f, f$(curAgent,Class A, Ks, Class B) | $f \in F$
  }
}
```

Before action-selection, therefore, the *Update_Models* function is called to ensure that action-selection is based on the most recent information about the state of the world. This means that the time spent updating the agent models is time taken away from selecting an action and carrying out the actual actions that will enable the agent in achieving its goals. In real-time domains (such as simulated RoboCup, for example, in which the agent only has 100ms to *sense, Update_Models, action_select*, and *act*) it is particularly desirable that the *Update_Models* function executes in the shortest possible time to guarantee that the selected action is still a valid action when it is actually carried out in the environment. Possible solutions include implementing functions in F such that the quality of the resulting model is traded off against faster computation time (using ideas such as *anytime algorithms* [Dean and Boddy, 1988], *flexible computations* [Horvitz, 1987], *imprecise computations* [Liu *et al.*, 1991], *limited rationality* [Russell and Wefald, 1989], and *approximate processing* [Lesser *et al.*, 1988]), or actively monitoring only a subset of the agent models available. This latter approach to the problem is referred to as the *monitoring selectivity* problem [Kaminka, 2000].

Approaches that have been used to tackle the selectivity problem include limiting perceptual intake [Ferguson, 1992], using hierarchical layouts [Jennings, 1995], and using social relationships (such as formations, role-similarity, mutual exclusion, and teamwork [Kaminka, 2000]; *paradigmatic agents* [Tambe, 1995]) that exist between the agents being modeled.

6 Future Directions for Agents-Modelling-Agents Research and Conclusions

When we consider the solution of trading-off the quality of the resulting model for faster model update, we are led to ask questions such as how do we implement anytime versions of algorithms such as RESL, IMBBOP, and the predictive approach? In general, how do we implement the active monitoring agent modelling techniques in F? This question has not received attention from the agent modelling research community.

A common assumption made by active monitoring approaches that infer values for type-2, type-3 and type-4 information is that the modeler possesses complete and accurate information in its knowledge source. For instance, the plan recognition network used in RESL is assumed to contain complete and accurate information about the reactive plan-library used by the modelee for selecting actions, and thus, observing the actions always allows for an explanation of why the modelee has chosen the observed action. This assumption limits the use of such techniques to agents belonging to the modeler's team, since it is usually the case that the same designer designs all the agents in a team. What is needed, therefore, is active monitoring approaches that can still maintain high quality models even when the knowledge source is not accurate or cannot completely explain observed behavior. A step in this direction is the use of reinforcement learning in building the knowledge source. Such learning is possible offline. It is not yet clear, however, how the knowledge source can be updated online, while being used by the modelling functions in F at the same time.

One of the design guidelines suggested by Olorunleke and McCalla [2004a] is that agents should be designed such that they can detect when their sensors have failed. How can this be achieved? It is clear that this suggests, at least, that the agent should have a self-model that contains information about the operational state of its sensors. What kinds of techniques can be brought to bear in maintaining such information accurately? Are there other strategies that can be used to avoid the spread of delusion from agent to agent other than those suggested by Olorunleke and McCalla [2004a]? Are there other unexplored solutions to the selectivity problem?

In conclusion, we have presented a survey of agent modelling ideas and presented a diagrammatic model in a way that allows new researchers to easily join the community and find ways of contributing to the problems being studied. Our classification of model contents into various types will be useful in comparing various techniques against each other. For example if we have to compare 2 modelling techniques, the starting point will be to determine what type of information is being computed by each. Clearly, it makes no sense to compare RESL with the predictive memory approach since (from the model presented) they compute different types of information (type-2 and type-4 vs. type-1) and use different knowledge sources (plan-libraries vs. past values). Thus, we can make decisions about which modelling techniques to compare (or replace) another approach with by determining what type of information is being computed, and where the inputs to the modelling functions are drawn.

Finally, it is clear that compiling an exhaustive list of all past research related to agents-modelling-agents will require a larger volume than this, but we believe that the model presented is a good starting point, for a model in which other works that have been left out can be plugged in.

References

[Bowling *et al.*, 1996] Bowling, M., Stone, P., and Veloso, M. (1996) Predictive Memory for an Inaccessible Environment. In *Proceedings of the IROS-96 Workshop on RoboCup*, pp. 28-34, 1996.

[Bui *et al.*, 2002] Bui, H.H., Venkatesh, S., and West, G.(2002) Policy Recognition in the Abstract Hidden Markov Model. In *Journal of Artificial Intelligence Research*, Vol.17, pp. 451-499, 2002.

[Carberry, 1990] Carberry, S. (1990) Plan Recognition on Natural Language Dialogue. The MIT Press, 1990.

[Dean and Boddy, 1988] Dean, T., and Boddy, M. (1988) An Analysis of Time-Dependent Planning. In Proceedings of the Seventh National Conference on Artificial Intelligence, pp.49–54, Menlo Park, California, 1988.

[Donaldson and Cohen, 1996] Donaldson, T. and Cohen, R. (1996) Turn-Taking in Discourse and Its Application To The Design of Intelligent Agents, *Agent Modelling 1996*, pp. 17-23, 1996.

[Feldman and Balch, 2004] Feldman, A. and Balch, T. (2004) Modeling Honey Bee Behaviour for Recognition Using Human Trainable Models. In *Proceedings of MOO 2004 Workshop*, pp. 17-24, 2004.

[Ferguson, 1992] Ferguson, I.A. (1992) TouringMachines: An Architecture for Adaptive, Rational, Mobile Agents. PhD Thesis – Technical Report 273, Computer Laboratory, University of Cambridge, UK, 1992.

[Firby, 1987] Firby, J. (1987) An investigation into reactive planning in complex domains. In Proceedings AAAI-87, 1987.

[Geib and Harp, 2004] Geib, C.W. and Harp, S.A. (2004) Empirical Analysis of a Probabilistic Task Tracking Algorithm. In *Proceedings of MOO 2004 Workshop*, pp. 65-71, 2004.

[Gmytrasiewicz *et al*., 1991] Gmytrasiewicz, P.J., Durfee, E.H., and Wehe, D.K. (1991) A Decision Theoretic Approach to Coordinating Multiagent Interactions. In *Proceedings of IJCAI-91*, pp. 62-68, 1991.

[Green and Lehman, 1996] Green, N. and Lehman, J. (1996). Comparing Agent Modeling for Language and Action. *Agent Modeling 1996*, 1996.

[Grosz and Kraus, 1999] Grosz, B. and Kraus, S. (1999) The Evolution of SharedPlans. In *Foundations and Theories of Rational Agencies*, A. Rao and M. Wooldridge, eds. pp. 227-262, 1999.

[Hamid et al., 2003] Hamid, R., Huang, Y., and Essa, I. (2003) ARGMode – Activity Recognition Using Graphical Models. In *Proceedings of Conference on Computer Vision and Pattern Recognition Workshop*, Vol.4, pp. 38-44, 2003.

[Horvitz, 1987] Horvitz, E. J. (1987) Reasoning about Beliefs and Actions under Computational Resource Constraints. In *Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence*, Seattle, July 1987.

[Huber and Durfee, 1995] Huber, M.J. and Durfee, E.H. (1995) On Acting Together: Without Communication. In *American Association for Artificial Intelligence*, Spring Symposium Working Notes on Representing Mental States and Mechanisms, Stanford, California, pp. 60-71, 1995.

[Huber and Simpson, 2004] Huber, M.J. and Simpson, R. (2004) Recognizing the Plans of Screen Reader Users. In *Proceedings of MOO 2004 Workshop*, pp. 1-8, 2004.

[Intille and Bobick, 1999] Intille, S.S. and Bobick, A.F. (1999) A Framework for Recognizing Multiagent Action from Visual Evidence. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (AAAI-99), pp. 518-525, 1999.

[Jennings, 1995] Jennings, N.R. (1995) Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems using Joint Intentions, *Artificial Intelligence*, Vol. 75(2), pp. 195-240, 1995.

[Kaminka, 2000] Kaminka, G. (2000) Execution Monitoring in Multi-Agent Environments. Ph.D. Dissertation, University of Southern California, Computer Science Department, 2000.

[Kaminka and Tambe, 1997] Kaminka, G., and Tambe, M. (1997) Towards Social Comparison for Failure Detection: Extended Abstract. In *Proceedings of the "Socially Intelligent Agents" 1997 AAAI Fall Symposium*, 1997.

[Kaminka and Tambe, 1998] Kaminka, G., and Tambe, M. (1998) What's Wrong With Us? Improving Robustness through Social Diagnosis. In *Proceedings of AAAI-98*, 1998.

[Kaminka *et al*., 1998] Kaminka, G., Tambe, M., and Hopper, C. (1998) The Role of Agent-Modeling in Agent Robustness. In *AI Meets the Real-World: Lessons Learned* (AIMTRW-98), 1998.

[Kaminka *et al.,* 2002] Kaminka, G.A., Pynadath, D.V., and Tambe, M. (2002) Monitoring Teams by Overhearing: A Multiagent Plan Recognition Approach. In *Journal of Artificial Intelligence Research,* Vol. 17, pp. 83-135, 2002.

[Kaminka and Avrahami, 2004] Kaminka, G.A. and Avrahami, D. (2004) Symbolic Behaviour Recognition. In *Proceedings of MOO 2004 Workshop*, pp. 73-79, 2004.

[Kautz and Allen, 1986] Kautz, A. and Allen, J.F. (1986) Generalized Plan Recognition. In *Proceedings of AAAI-87*, pp.32-37, Menlo Park, AAAI Press, 1986.

[Kitano *et al*., 1997] Kitano, H., Kuniyoshi, Y., Noda, I., Asada, M., Matsubara, H., & Osawa, E. (1997). RoboCup: A challenge problem for AI. *AI Magazine*, vol. 18(1), pp. 73–85, 1997.

[Kraus and Rosenschein, 1992] Kraus, S. and Rosenschein, J.S. (1992) The Role of Representation in Interaction: Discovering Focal Points Among Alternative Solutions. In *Decentralized AI*, Vol. 3, Amsterdam, Elsevier Science Publishers, 1992.

[Lerman and Galstyan, 2004] Lerman, K. and Galstyan, A. (2004) Automatically Modeling Group Behaviour of Simple Agents. In *Proceedings of MOO 2004 Workshop*, pp. 49-55, 2004.

[Lesser *et al*., 1988] Lesser, V., Pavlin, J., and Durfee, E. (1988) Approximate Processing in Real-Time Problem Solving. AI Magazine, Vol. 9(1), pp. 49–61, 1988.

[Liu *et al*., 1991] Liu, J., Lin, K.J., Shih, W.K., Yu, A.C., Chung, J.Y., and Zhao, W. (1991) Algorithms for Scheduling Imprecise Computations. IEEE Computer Vol.(24), pp.58–68, 1991.

[Mengshoel and Wilkins, 1996] Mengshoel,O., and Wilkins, D. (1996) ReCognition and Critiquing of Erroneous Agent Actions. *Agent Modeling 1996*, pp. 61-68, 1996.

[Olorunleke, 2002] Olorunleke, O. (2002) Fragmented Agent Modelling. Master’s Thesis, Department of Computer Science, University of Saskatchewan, Canada, August, 2002.

[Olorunleke and McCalla, 2003]Olorunleke, O. and McCalla, G. (2003) Overcoming Agent Delusion. In *Proceedings of AAMAS-2003*, pp. 1086-1087, Melbourne, Australia, 2003.

[Olorunleke and McCalla, 2004a]Olorunleke, O. and McCalla, G. (2004) The Study of Delusion in Multiagent Systems. In *Proceedings of MOO 2004 Workshop*, pp. 33-40, 2004.

[Olorunleke and McCalla, 2004b] Olorunleke, O. and McCalla, G. (2004) Model Sharing in Multiagent Systems. In *Proceedings of AAMAS-2004*, pp. 1412-1413, New York, 2004.

[Parker, 1993] Parker, L.E. (1993) Designing Control Laws for Cooperative Agent Teams. In *Proceedings of the IEEE Robotics and Automation Conference*, pp. 582-587, 1993.

[Pynadath and Wellman, 2000] Pynadath, D.V. and Wellman, M.P. (2000) Probabilistic State Dependent Grammars for Plan Recognition. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 2000.

[Russell and Wefald, 1989] Russell, S.J., and Wefald, E.H. (1989) Principles of Metareasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 1989.

[Rybski and Veloso, 2004] Rybski, P.E. and Veloso, M.M. (2004) Using Sparse Visual Data to Model Human Activities in Meetings. In *Proceedings of MOO 2004 Workshop*, pp. 9-16, 2004.

[Stone, 1998]Stone, P. (1998) Layered Learning in Multi-Agent Systems. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, Dec. 1998.

[Stone *et al.,* 2000] Stone, P., Riley, P. and Veloso, M. (2000) Defining and Using Ideal Teammate and Opponent Models. In *Proceedings of IAAI 2000*, 2000.

[Tambe, 1995] Tambe, M. (1995) Recursive Agent and Agent-group Tracking in a Real-time, Dynamic Environment. In *Proceedings of ICMAS-95*, AAAI Press, 1995.

[Vassileva *et al.*, 2002] Vassileva, J., McCalla, G. and Greer, J. (2002) Multi-Agent Multi-User Modelling in I-Help. In *User Modelling and User Adapted Interaction*, e. Andre and A. Paiva (eds.) Special Issue on User Modelling and Intelligent Agents, 2002.