Knowledge Exchange between Agents in Real-Time Environments

Jeppe Revall Frisvad[1] Peter Falster[1] Gert L. Møller[2] Niels Jørgen Christensen[1]

[1] Informatics and Mathematical Modelling
Technical University of Denmark
{jrf, pfa, njc}@imm.dtu.dk

[2] Array Technology A/S
Nyhavn 16, 1051 Copenhagen, Denmark
glm@arraytechnology.com

Abstract

To obtain unpredictable social interaction between autonomous agents in real-time environments, we present a simple method for logic-based knowledge exchange. A method which is able to form new knowledge rather than do simple exchange of particular rules found in predetermined rule sets. The applicability of our concept is demonstrated through a simple visualization of a real-time 3D environment, where agents seek to persuade opponents to join their team. This is done through cooperation with friends and education of neutral agents.

Keywords: agents for games, knowledge exchange, logic, learning, multi-agent systems, real-time animation, synthetic characters.

1 Introduction

Most agents in dynamic real-time environments reason according to a number of observations. Their actions change either directly according to the observed states or indirectly according to a plan of actions fitting current or past observations. The agents typically draw conclusions based on the same behavioral, motivational, or cognitive model throughout each simulation.

These static models often lead to predictable behavior. To improve on this issue some systems employ machine learning techniques to tune parameters in a model. In this way the feeling of monotonous behavior can be broken. Learning techniques employed in dynamic environments, however, rarely go beyond tuning of parameters. The structure of the model capturing the knowledge of an agent usually stays the same. In a logic-based model this means that the rule set controlling an agent is the same during each simulation. In this paper we present a simple method for exchange of knowledge between agents such that their rule sets will change significantly at runtime and develop differently for each simulation.

One way to have agents communicate knowledge would be to let them communicate particular rules found in their rule sets. This would, however, not lead to new rules and the resulting behavior would be imitation rather than social interaction. What we propose, is a way to let the agents infer the relation between a few queried variables. In this way new knowledge is formed and communicated from one agent to another at each request. It is our opinion that this approach leads to interesting social interaction between autonomous agents in real-time environments.

2 Related Work

The Soar architecture has existed as an AI system for problem solving since the early eighties and in [10] it was proposed that Soar could be developed into a system capable of general intelligent behavior. At this point Soar was mainly concerned with planning and learning to achieve a goal in a closed system. Later an interface was developed between Soar and the commercial computer games Quake II and Descent 3 to let AI characters play games at the human-level [13]. This has developed into a promising Soar AI engine which can simulate a director agent directing synthetic characters as actors in a game [11]. To make this work, the actors are semi-

autonomous agents which base their choice of action on director commands. In other words the Soar engine features one-way communication from director to actors, but not interagent communication.

Another approach to high-level control of characters in games and animation was developed by Funge and presented eg. in [7]. With experience from both AI research and commercial game development Funge gives, in [8], a good overview of the kind of AI techniques which are applicable to NPCs (Non-Player Characters) in games.

To let the knowledge of agents develop during a simulation, it is an option to have an annotated environment where the agents can pick up information at different locations. This approach is described eg. in [3]. The environments they use for their experiments are, however, not real-time animated 3D environments. On the other hand many games equip their environment with affordances [8], which is a similar concept. The idea of an annotated environment could be combined with our method. For example by giving annotations the same representation as we use for knowledge.

Most projects addressing interagent communication in virtual 3D environments, aim at a system able to make agents engage in a verbal conversation that can be followed by an observing human, see eg. [2, 1]. The interagent communication presented in this paper is not meant to give an exact imitation of human-human conversation. The idea is rather to have an extended version of the exchange of internal states which NPCs in games sometimes employ [8]. Letting the agents exchange knowledge as well as internal states, enables them to teach each other new reasons to perform actions. The hope is that this leads to interesting and unpredictable behavior.

3 Theory

It is necessary for each agent to draw inference on its rule set. Basically there are two options: Either the employed inference engine is based on production rules, ie. *if-then* constructs, where the *if*-part is a firing condition such that one rule fires at the time, or it is logic-based, that is, it employs a proof procedure which is sound and complete. See eg. [9, 8].

The method we present for knowledge exchange has been derived from array-based logic [5, 12, 6] and is, therefore, logic-based. It has the prerequisite that each rule set must be reproducible as a Boolean function (defined below) and, consequently, it has the restriction that each variable of each rule set must have a discrete form. The particular choice of inference engine is, however, not crucial to the applicability of the method. Therefore we will, in this section, first give pointers to one inference engine of each kind, then describe the method using Boolean functions, and lastly describe how our approach translates to an inference engine based on production rules.

The Rete algorithm [4] is the core of many inference engines based on production rules. Recently the RC++ language [14] has been developed with emphasis on application in computer games. If translation to production rules is done, our method could be used with RC++.

Array Studio[1] and the associated Compiler and Runtime libraries embody an inference engine which is easily used with our method, since it also has its foundations in array-based logic. In a railroad interlocking system of 12,780 variables and 4,743 relations the Array Runtime has a worst case response time of 22 ms when an arbitrary query regarding the rule set is posted. Queries on a simpler rule set of 23 variable and 75 relations have worst case response times of 0.6 ms. These numbers refer to simulations run on a 1 GHz Pentium CPU. This shows that the Array tools also make a suitable inference engine for agents in real-time environments.

A simplistic example of a rule set is:

$$\begin{aligned} P &\Rightarrow Q \\ Q &\Rightarrow R\,, \end{aligned}$$

where P, Q, and R are Boolean variables. We *define* that B_n is the set of *Boolean functions* $\phi : D_1 \times \ldots \times D_n \rightarrow \{0,1\}$ mapping n values in the finite domains $D_1, \ldots, D_n$ into a single Boolean value. Clearly the simplistic rule set mentioned before could be represented by a Boolean function $f \in B_3$ given as

$$f(P,Q,R) = (P \Rightarrow Q) \wedge (Q \Rightarrow R)\,,$$

where $D_1 = D_2 = D_3 = \{0,1\}$.

[1] See http://www.arraytechnology.com/

Suppose an agent $\mathcal{A}$ has the knowledge described by the Boolean function

$$g(P, Q, R) = P \Rightarrow Q$$

and an agent $\mathcal{B}$ has the knowledge described by the function

$$h(P, Q, R) = Q \Rightarrow R\,.$$

Agent $\mathcal{A}$ may realize that the variable R representing an action, which $\mathcal{A}$ can perform, has no meaning with respect to its knowledge. Then it can send a request for knowledge about the relation between Q and R to agent $\mathcal{B}$. If agent $\mathcal{B}$ chooses to cooperate, it replies with the knowledge

$$h'(Q, R) = Q \Rightarrow R.$$

By a union of old and new knowledge in agent $\mathcal{A}$, that is, by adding the received function to $\mathcal{A}$'s rule set, it will obtain the knowledge

$$g(P, Q, R) = (P \Rightarrow Q) \wedge (Q \Rightarrow R)\,.$$

Now agent $\mathcal{A}$ will perform the action represented by R for example when P is true.

Consider the function h' which is excerpted from the knowledge of agent $\mathcal{B}$ and communicated to agent $\mathcal{A}$ upon request. If h had been a large knowledge base composed of many rules, but no rule exactly concerning the relation between Q and R, it would not immediately be clear how h' should be found. This is the problem we propose a solution for.

It is reasonable to assume that the request for knowledge from one agent to another always concerns the relation between a few variables only, ie. in the following we assume that k is small. In any case it would be unrealistic to have agents exchanging large knowledge bases at runtime in a dynamic environment. Therefore we propose the following method for excerption of a Boolean function which explains the relation between a few of the variables in a rule set.

Suppose the rule set representing the knowledge of an agent concerns n different variables $P_1, \ldots, P_n$. Let $f \in B_n$ be the Boolean function corresponding to the rule set. If $I = \{1, \ldots, n\}$ is an index set, we can let $i \in I^k$, $k < n$, with $i_1 \neq \ldots \neq i_k$, contain the indices of the variables between which a relation was requested. Let $j \in I^{n-k}$ be given as the indices which exist in the index set I, but are not accounted for in i. The method is then to find $f' \in B_k$ explaining the relation between $P_{i_1}, \ldots, P_{i_k}$ according to f as

$$f'(P_{i_1}, \ldots, P_{i_k}) = \bigvee_{P_{j_1} \in D_{j_1}, \ldots, P_{j_{n-k}} \in D_{j_{n-k}}} f(P_1, \ldots, P_n) \quad (1)$$

It will always be the case that $f \models f'$, since if $f(P_1, \ldots, P_n) = 1$, then $f'(P_{i_1}, \ldots, P_{i_k}) = 1$ why $f \Rightarrow f'$ is a tautology. Hence, (1) presents a valid form of inference. Technically (1) corresponds to an orthogonal *disjunctive projection* in the image of the Boolean function f. The same method can be used to prove modes of inference such as the hypothetical syllogism and modus ponens [5].

Performing all disjunctions in (1) leads to a combinatorial explosion. But in practice the operation is quite efficient, since, to find the function value $f'(P_{i_1}, \ldots, P_{i_k})$, we need only know if f *can* attain a true value when $P_{i_1}, \ldots, P_{i_k}$ are bound arguments. Moreover k was assumed to be small, so we can simply store the indices where f' returns true as the representation of f'. When this is done f' is send as the reply to the agent making the request for knowledge.

If inference is performed using an algorithm such as Rete, the rules should be on the form $R \Rightarrow A$, which corresponds to

if R *then* A,

where R is a logical relation between some observable variables and perhaps some variables representing internal states of the agent. A is an *action proposition*, ie. when A is true the agent performs a corresponding action. In this case the relation between a single action and a few observable variables should be requested rather than relations between several action propositions. The f' function given as reply to such a request needs to be translated back into a production rule of the form given above. To do this we need the argument index $a \in \{1, \ldots, k\}$ of the action proposition, then

if $f'(P_{i_1}, \ldots, P_{i_{a-1}}, 0, P_{i_{a+1}}, \ldots, P_{i_k}) = 0$
$\wedge\ f'(P_{i_1}, \ldots, P_{i_{a-1}}, 1, P_{i_{a+1}}, \ldots, P_{i_k}) = 1$
then A

gives the correct translation. This makes the method applicable to systems using inference engines for production rules.

If contradictory rules are unacceptable, a check of consistency between the newly obtained rule and the old rule set should be done before the new knowledge is adopted. If a contradiction is found, the new rule should be disregarded rather than adopted and the agent should find a different subject for questioning.

4 Case Study

To present a simple example illustrating the effect of our method we have cloned a human-looking agent. The case we study is a simulation of an election campaign where a few *campaigner*s from two different teams should persuade as many *neutral* agents as possible. To be successful, a *campaigner* should not only persuade the *neutral*s, but also teach them how to persuade other *neutral*s. Through cooperation the *campaigner*s will even be able to persuade *campaigner*s from the opposing team.

All the clones are given the following six actions to choose from: *walk*, *persuade*, *teach*, *query*, *reject*, and *accept*. Each action has a corresponding animation cycle. A sample frame from each action is shown in figure 1.

Beside actions each character does four observations about the surrounding environment: *friend_close*, *opponent_close*, *knowledge_received*, and *query_received*. An observation does not have a corresponding animation cycle. They are just environment states sensed in the proximity of the character three or four times every second.

For this case study we have two rule sets based on the actions and observations. The rule set for a *neutral* agent is:

$\neg friend\_close \Rightarrow walk$

$friend\_close \Rightarrow query$

$knowledge\_received \wedge friend\_close \Rightarrow accept$

$knowledge\_received \wedge \neg friend\_close \Rightarrow reject$

The rule set for a *campaigner* is:

$\neg opponent\_close \Rightarrow walk$

$friend\_close \wedge opponent\_close \Rightarrow persuade$

$query\_received \wedge friend\_close \Rightarrow teach$

$knowledge\_received \wedge \neg opponent\_close \Rightarrow accept$

$knowledge\_received \wedge opponent\_close \Rightarrow reject$

At times the agents will want to do several actions simultaneously. A *neutral* agent receiving knowledge without any friends in its proximity, will want to reject the knowledge and walk away

Figure 1: A sample frame from each of the six actions. From top left to bottom right: walk, persuade, query, teach, reject, accept.

at the same time. This is not necessarily a bad result, but since we only have an animation cycle for each action and not for combinations, one of the wanted actions should be chosen eg. stochastically.

The scenario consists of fifteen clones packed in a room. Eleven of those are *neutral*, four are *campaigner*s. The *campaigner*s are split in two different teams. A red team and a blue team. The *neutral*s are green. If an agent is persuaded by an argument of a different team, it changes color. *Neutrals* are both friends and opponents until they are persuaded. Therefore one *campaigner* can persuade a *neutral*, but it takes at least two *campaigner*s to persuade a *campaigner* from a different team. This follows from the *campaigner* rule set.

A request for knowledge points to a particular action. The Boolean function communicated

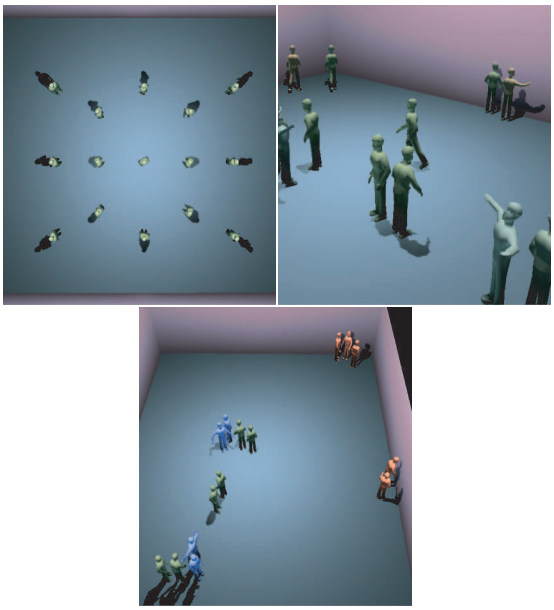

Figure 2: A simulation where no knowledge is exchanged. From top left to bottom: (a) The initial positions and directions of the agents. (b) A blue campaigner in the lower right corner is performing the teach action, but without knowledge exchange this has little effect. A red campaigner is persuading a neutral in the background. (c) The simulation quickly arrives at a deadlock were the few campaigners are constantly queried for knowledge. In this simulation the queries are not met.

as an answer to a request, specifies the relation between the requested action and all four observations.

5 Results

Figures 2, 3 and 4 present three annotated simulations. One where the agents are unable to communicate knowledge and two where they are able to. The *campaigner* rule set can be used for all the agents in the case where knowledge is not exchanged. This leads to more interesting behavior, but we still find the resulting behavior of the agents less predictable in the case where knowledge exchange is possible.

The presented case study has no problems running in real-time on a 1.7 GHz Pentium4 CPU. The case is quite simple and therefore its purpose is to document the interesting, unpredictable social interaction which the presented method can result in rather than the scalability of our approach. However, we believe that the method is scalable and for that reason it may be available in future versions of Array Runtime.

Figure 3: A simulation where knowledge is exchanged. From top left to bottom right: (a) The blue team gets a head start with persuasion and education of a neutral. (b) The red team follows up with some teaching. (c) A red agent is heading for the blue camp where an interesting event has occurred. The agent having the strongest blue color, the one who is about to walk away, has just persuaded and taught a neutral agent who is now persuading and teaching the passive neutral agent standing behind him. (d) The blue team seems to get the upper hand. (e) Here two blue agents are close to a red one and can, therefore, persuade the red agent to join the blue team. (f) Here the last red agent has just been persuaded in the lower right corner.

6 Conclusion

In this paper a method for knowledge exchange between autonomous agents has been proposed. The method is based on the assumption that a logic-based rule set, composed of variables defined on a finite domain, can be used to describe the knowledge of an agent. We have described a simple equation which finds a Boolean function representing an arbitrary subset of an agent's knowledge. Preferably a small subset. This is

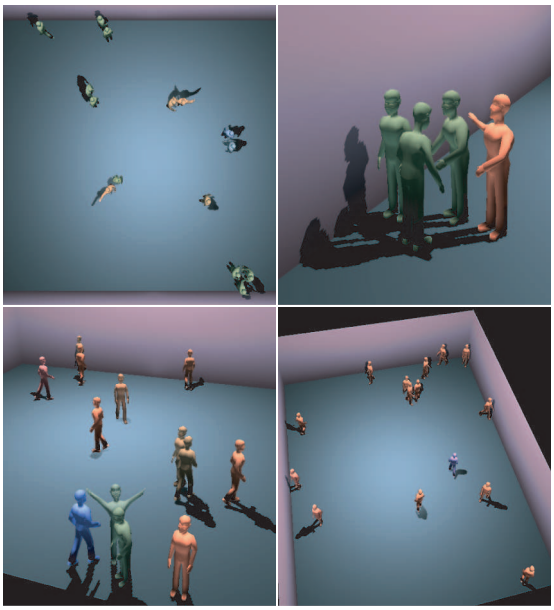

Figure 4: To show that no two simulations are the same, this figure presents another simulation where exchange of knowledge is performed. From top left to bottom right: (a) In this simulation the red team starts out with two quick persuasions. A blue campaigner has also caught a neutral. Note the group of three neutrals in the lower right corner. (b) After a short while, a red agent finds and persuades the group mentioned before. (c) Even though a blue campaigner has just persuaded a neutral and taught it how to teach, it is only a matter of time before the many red agents will get the upper hand, and (d) indeed red prevails in this simulation.

used for exchange of knowledge.

The impact of having knowledge exchange available for real-time applications has been illustrated through a simple visualization of a 3D environment where agents show cooperative and, to a certain extend, unpredictable behavior through knowledge exchange.

It is our opinion that the presented method for knowledge exchange, gives a range of possibilities for new, interesting and unpredictable behavior in games and animation using high level control of synthetic characters.

Acknowledgement

Thanks to Rasmus Revall Frisvad for creation of the animation cycles used in our case study.

References

[1] A. Caicedo, J.-S. Monzani, and D. Thalmann. Communicative autonomous agents. In N. Magnenat-Thalmann and D. Thalmann, editors, *Deformable Avatars*, pages 217–227. Kluwer Academic Publishers, 2001.

[2] J. Cassell, C. Pelachaud, N. Badler, M. Steedman, B. Achorn, T. Becket, B. Douville, S. Prevost, and M. Stone. Animated conversation: Rule-based generation of facial expressions, gestures & spoken intonation for multiple conversational agents. In *Proceedings of SIGGRAPH 1994*, pages 413–420, 1994.

[3] P. Doyle. Believability through context: Using "knowledge in the world" to create intelligent characters. In *Proc. of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2002)*, pages 342–349, July 2002.

[4] C. L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[5] O. I. Franksen. Group representation of finite polyvalent logic: A case study using APL notation. In A. Niemi, editor, *A Link between Science and Applications of Automatic Control, IFAC VII, World Congress 1978*, pages 875–887, Oxford, 1979. Pergamon Press.

[6] O. I. Franksen and P. Falster. Colligation or, the logical inference of interconnection. *Mathematics and Computers in Simulation*, 52(1):1–9, March 2000.

[7] J. D. Funge. *AI for Games and Animation: A Cognitive Modeling Approach*. A K Peters, 1999.

[8] J. D. Funge. *Artificial Intelligence for Computer Games*. A K Peters, 2004.

[9] J. Giarratano and G. Riley. *Expert Systems: Principles and Programming*. PWS Publishing Company, third edition, 1998.

[10] J. E. Laird and A. Newell. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, September 1987.

[11] B. Magerko, J. E. Laird, M. Assanie, A. Kerfoot, and D. Stones. AI characters and directors for interactive computer games. In *Proc. of the 2004 Innovative Applications of Artificial Intelligence Conference*. AAAI Press, July 2004.

[12] G. L. Møller. *On the Technology of Array-Based Logic*. PhD thesis, Electrical Power Engineering Department, Technical University of Denmark, 1995.

[13] M. van Lent, J. E. Laird, J. Buckman, J. Hartford, S. Houchard, K. Steinkraus, and R. Tedrake. Intelligent agents in computer games. In *Proc. of the National Conference on Artificial Intelligence*, pages 929–930, July 1999.

[14] I. Wright and J. Marshall. The execution kernel of RC++: RETE∗, a faster RETE with TREAT as a special case. *International Journal of Intelligent Games and Simulation*, 2(1):36–48, February 2003.