Abstract

The development of multiple agent systems faces many challenges, including agent coordination and subdivision of tasks. Minsky’s “The Society of Mind” [1] provides a framework for addressing these multi-agent system problems. Following this framework, a society of agents is developed and applied to the domain of single player and multiplayer games. The advantage of the society approach is the efficient re-utilization of agents across multiple independent game domain problems.

1. Introduction

Distributed semi-autonomous intelligent problem solving is the function of multi-agent systems. Ontologisms or formalisms (frameworks) for development of multi-agent systems have been recognized as a critical element [2]. The ontology of using the perspective of a society for modeling agents, especially agent “social” interactions has been used for quite some time [3,4], where agents are viewed as belonging to a social collective and cooperating for the overall good of the society. These social perspective ontologisms even extend themselves to promote the utilization of software agents for modeling real-world social populations and corresponding interactions [5]. The social perspective also defines multiple types of agent interaction relationships in multi-agent systems dependent on context [6]

A different semi-social perspective ontology may be based on Minsky’s [1] The Society of Mind (SoM), which proposes that human intelligence and problem solving ability is derived from a large collection of independent affective, autonomous, cognitive, and reactive actors (Minsky actually uses the term agent, but it is used in a different way than traditional software-based agents). These actors handle atomic tasks within the biologic or software system and interact with each other to handle more complex tasks. As such, these actors may be emulated utilizing current software agent technology. The goal of the research reported in this article is to develop this unique ontological view and apply the society of agents (SoA) multi-agent system ontology in a complex problem solving domain.

More recent research has utilized a social perspective for multi-agent systems that emulate insect community behaviors to govern how agents interact [7], including applications in: construction [8], unmanned air vehicles (robots) [9], supply chain management [10], manufacturing scheduling [11], and datamining [12], to name a few. These insect society models typically address communication issues between multiple agents and also subdivision of complex planning tasks. The new SoA ontology proposed in this article also addresses the efficient division of tasks, but in a more problem solving oriented environment and hence we will distinguish this by calling it problem subgoaling. The other distinct advantage of the SoA ontology is the efficient and necessary re-use of atomic agents across multiple problem solving tasks.

2. The SoA ontology model

A very brief review of SoM [1] is given to enable the justification of the SoA ontology and its subsequent application. As mentioned, the SoM actors are atomic, meaning that each actor is responsible for a single and indivisible component of the overall systems. Some of these atomic actors will be responsible for performing actions, while others detect or maintain state. The individual actors are not meant to be viewed as intelligent, but rather any perceived intelligence of the system is an effect from the various actors interacting to accomplish the needs or tasks of the system. Thus the whole is definitely more than the sum of the parts.

The various actors in SoM are responsible for perception, reasoning, memory, action, and a few other anthropomorphic properties such as emotions and sense of self. Therefore, any multi-agent ontology derived from SoM must require that agents perform atomic behaviors and that these behaviors include perception, reasoning, memory, and action.

The SoA ontology for multi-agent systems requires that:

- agents operate at the atomic level, where each agent is responsible only for a single
individual non-divisible task, specifically agents must be defined that will:
  o read input from the domain for the current task (text, graphics, XML, voice, other,...),
  o reason (problem solve) utilizing various problem solving strategies, one per agent,
  o if needed, display information or requests for information (alternately this type of agent may also affect actions to occur within the domain such as robot movement),
  o serve as a local short-term memory for other agents;
• agents may be, and in fact should be, reused in multiple domain contexts as long as the atomic task of the agent is appropriate within the specified context;
• agents act independently and thus have a widely distributed locus of control, however;
• agents must interact with other agents in order to accomplish any task more complex than the atomic tasks for which the agents have been specified;
• all agents within the society may be active, however it is possible that only a subset of the society may be required to interact in order to solve the current domain problem or task.

These are the fundamental properties of the SoA ontology. Additional specificity for SoA agents may evolve as the research progresses.

As indicated in the list above, four different types of agents are specified for any society: input agents, output or action agents, reasoning agents, and memory agents. This natural division of labor is based on the SoM model of human cognition. If this division of tasks is followed then no single agent should be able to solve a standard domain problem solving task individually. Agents must work collectively to accomplish tasks. Additionally, one agent must be responsible for identifying which of the agents within the society are needed to accomplish a specified task.

Each agent understands a subset of information needed for accomplishing domain tasks. Typical agent interaction will be in the form of agents requesting missing information from other agents or suggesting modifications to information currently held by other agents. This dynamic information processing and resolution is responsible for the intelligent problem solving capabilities of the society. Solutions to domain problems may be identified in multiple ways. It is possible that an agent is specified that has the atomic action of recognizing when a solution has occurred or alternately solutions may be identified when each active agent has refined its current information so that a unique value or solution is obtained.

While the specifics of the SoA ontology are specified above, the necessity and implementation of these requirements may be better understood through an example. The next section will examine developing a multi-agent society for playing games that follows the SoA ontology.

3. SoA society for playing games

The domain of application for the initial SoA society is games. Games are chosen because they are well understood and games have served as a traditional testing ground for artificial intelligence research. The types of games to be addressed by this society is wide ranging and only requires that some type of internal representation of the game is possible for the agents to encapsulate and interact with and the game must be winnable so that the society of agents may determine when a solution is reached. This includes one-person, two-person, and eventually multi-person games that are both bounded and unbounded (indicating the lack of limits for the game board or playing area). The current implementation of the SoA Game Society is capable of playing: sudoku, kakuro, tic-tac-toe (TTT), three dimensional tic-tac-toe (3D-TTT), and work is progressing on caro (a Vietnamese game similar to TTT, but played on an unbounded/limitless board and requires 5 similar marks in a row for victory). Future research will extend the Game Society to also play checkers, chess, shogi, go, and backgammon among others.

The current set of games solvable by the SoA Game Society is chosen to incrementally increase the complexity of the games. Sudoku is a single player number puzzle with a consistent bounded board (9X9), while kakuro is also a single player number puzzle, but introduces additional solution requirements for the agents and also uses a bounded yet inconsistent board. Both sudoku and kakuro have a unique solution if the puzzle is properly configured. TTT introduces two person games and therefore introduces strategy into a consistent bounded board. 3D-TTT incorporates and additional dimension that the reasoning, input, and output agents must address.

Evaluation of the agents is done by determining if the society is solving a specific game problem at an advanced human player level. For sudoku and kakuro this implies the ability to solve arbitrarily difficult puzzles, while for TTT this implies never losing (since TTT will always result in a draw
between two knowledgeable players) and winning against a less skilled player.

As described in the previous section, several different types of agents are required to form the “society”. The various agents in the Game Society are listed in Table 1 along with their purpose in the society. All of the agents exist within the society and will be described to try and give a sense of how the society functions.

First, the Input and Output agents are responsible for all communication with users and also other systems outside of the society. The Recognition agent is similar to a centralized agent controller that is implemented in many multi-agent systems [13], with the exception that the Recognition agent passes control to a smaller collection of agents to play the game and thus creates a decentralized problem solving architecture. The Recognition agent identifies the current type of game being played. Currently the game is specified to the agent, but ongoing developments are looking at automatic recognition of games from identification of board shape and pieces involved in the starting position. Once the Recognition agent determines the game being played, it then requests game information from the corresponding Game Rule agent.

Each game that the society can play has an associated Game Rule agent defined, which specifies the type and number of pieces, size and shape of the game board, along with any rules for the play of the game (e.g., for sudoku and kakuro this would include a rule that only one of any digit may occur in a Segment agent’s collection of Cell agents). The Game Rule agents act as a long term memory in the society, recording pertinent information applicable to a specific game. Information about the games is kept within the Game Rule agents in XML format, which is a common communication protocol for multi-agent systems [14].

The Board agent serves as a short-term memory for the Game Society and holds the internal representation and current status of the game (which pieces are located where). For most two or more player games, the board has a pre-specified starting position and this information is communicated from the Game Rule agent to the Output and Board agents via the Recognition agent. The two one player numeric puzzle games currently solved by the SoA Game Society must have the starting position specified either by a user to an Input agent or may be read in from a file.

Once the Recognition agent understands the games requirements, it then creates a smaller working “community” from the larger society, which allows the society to dynamically adapt its configuration to fit the requirements of the current problem solving task (which has been noted as an important feature for multi-agent systems [15]). Cell agents are created to represent each of the playable locations in the game (e.g., 81 for sudoku and 64 for chess) and Segment agents are created to organize the Cell agents into meaningful groups for the specific game (e.g. 27 for sudoku: 9 column segments, 9 row segments, and 9 area segments). Segments are determined based on the movement type of pieces involved in the game (if any) and winning patterns as defined in the Game Rule agent. Each Cell and Segment agent is loaded with appropriate Game Rule knowledge (e.g., Cell agents know the types of values they can hold and if multiple values are permitted).

The Recognition agent also loads the community with any Strategy agents that might apply. The architecture of the Game Society is displayed in Figure 1 (Note: not all of the lines indicating instantiation of Cell and Segment agents are shown in the figure to keep the figure more readable).

The working community within the society is composed of instantiated cell and segment agents from their Java class definitions within the society as well as the actual strategy agents from the society identified by the Recognition agent as possibly contributing towards a solution. Since the number of strategy agents is currently small all of them (with the exclusion of the mathematical reasoning agents for...
the non-numeric piece games) are included in the working community of agents for solving/winning any specific game. Future work is planned to enable the Recognition agent to analyze the game description, game pieces, and game rules contained within the Game Rule agent to determine appropriate strategies. A planned heuristic will automatically increase the available strategies to the community if the community fails to find a solution.

Future work is planned to enable the Recognition agent to analyze the game description, game pieces, and game rules contained within the Game Rule agent to determine appropriate strategies. A planned heuristic will automatically increase the available strategies to the community if the community fails to find a solution.

Figure 1. SoA Game Society architecture.

The Segment and Strategy agents supply any problem solving intelligence for the society. The Segment agents utilize game rules to organize the Cells and to assist Cells in identifying unique or desirable piece values to hold. Strategy agents apply standard problem solving strategies including:

* subgoaling or simplification, which breaks the problem into smaller size chunks;
* pattern recognition for important patterns specified by the Game Rules agent;
* hill climbing, which tries to reduce the difference between the current game state held in the Board agent and the desired outcome;
* analogy to similar problem situations;
* working backwards, which attempts to move from the goal state backwards to the current state;
* heuristics, utilizing heuristic problem solving strategies for specific domains (such as controlling the corners in TTT); and
* additional agents that implement basic knowledge such as standard mathematics to assist in discovering numeric patterns and allowing pattern rotations.

New strategy agents may be added at any time to the society to increase the society’s reasoning capabilities.

Inter-agent communication is performed in two distinct ways as identified in Figure 2. Agents may communicate directly by either asking another agent for a value or telling another agent a piece of information. As an example, a Cell agent is told which types of pieces it may hold for the game by the Recognition agent, which acquires this information by asking the corresponding Game Rule agent. Additionally, the Board agent serves as a working memory agent that all Cell, Segment, and Strategy agents utilize for solving the current game problem (e.g., solving the puzzle or selecting a move with a high probability of leading to a win). Whenever an agent (Cell, Segment, or Strategy) updates the Board agent through the direct communication mechanism just described, all other agents are immediately aware of the change to the Board agent and may then utilize this new information to complete their tasks of solving or winning the game.

4. Results

The Game Society described in the previous section implements the SoA ontology for a multi-agent system in the domain of games (in general). It should be kept in mind that this is ongoing research and the Game Society will grow to include many additional games and the development of new agents within the society.
First we will examine how well the Game Society adheres to the SoA ontology. The Game Society effects input and output agents, which serve as the user interface for the society of agents. Future work will examine interaction of the society with other game systems which will also be handled via input and output agents.

Reasoning/intelligence is achieved via the Segment and Strategy agents. These agents interact with each other and the Cell agents to solve game domain problems. Recall that both Segment and Cell inherit required domain specific information for a particular game from the corresponding Game Rule agent. Simpler games such as Sudoku may be solved using just the Cell and Segment agents by having the Segment agents recognize intersections between the various Segment agents and guide the Cell agents in producing a solution by helping them eliminate possible values form consideration (a.k.a. problem space reduction via constraint propagation). Kakuro works similarly, but does utilize mathematical strategy agents that are used to identify patterns of numbers that may fit within a specified segment of cells within the Game Society, allowing the SoA to understand and create non-specified constraints.

The TTT and 3D-TTT games could play with just the Segment and Cell agents loaded with Game Rule knowledge, however this play would not be considered intelligent. These games introduce the requirement for Strategy agents to exist in more complex game problem solving tasks. Defensive and offensive heuristics concerning \( N \) in a row patterns played on a matrix-like board are commonly known and have been coded as pattern heuristic Strategy agents. The heuristics used in TTT are also re-used in 3D-TTT with one additional heuristic Strategy agent. Early work with applying the Game Society to caro has also re-utilized the various TTT heuristic agents and similar re-use is expected in similar pattern-oriented games such as: Othello/reversi and go.

Clearly, the list of strategies currently implemented for the Game Society given above will require further additions to utilize expert strategies form specific games, such as utilizing multiple pieces to defend or attack a Cell or Segment for games like chess and shogi.

Short-term memory (the current problem state) is provided by the Board agent. The Game Rule agents serve as a longer-term memory, but for narrowly defined domain subsets of specific domain information.

The atomicity requirement for the agents is achieved for all of the specified agents, by making sure that their task and knowledge is specific to a single well defined indivisible task, with the exception of the Recognition agent. Currently the Recognition agent performs two tasks: identification of the domain with transfer of appropriate Game Rule information to other agents and also the creation of the problem solving community. The decision to create the community with the same agent was made for simplicity in programming. Future work on this research project will investigate making the entire Game Society operate at the atomic level as specified by the SoA ontology, with the Recognition agent performing the task of recognizing the game and holding all game relevant information from the corresponding Game Rule agent and behaving similar to the Board agent through enabling the other agents to examine the values held within the Recognition agent to determine game specific requirements and a new Task agent that will be responsible for assembling the internal community.

Since the Game Society currently solves four different games, with work progressing on a fifth game, each with a unique set of properties, the Game Society clearly enables agents to be re-used across multiple tasks within the domain. Specifically, the Recognition agent, Board agent, Input and Output agents, Cell agents, and Segment agents will be used for every game. This is possible with the use of the Game Rule agents, which are not used across multiple domain problems (games), which allow the dynamic reconfiguration of the Board and Segment agents and automatic learning of appropriate Cell content and Output characteristics. Strategy agents are re-used across all games that share specific
properties, such as using the math Strategy agents in all games that contain numeric pieces or values within the Cell agents. Various pattern Strategy agents are used in all of the games currently implemented. The heuristic strategies are added in response to a performance need within a specific domain game problem, however these may find re-use in other game problems that share similar heuristics (e.g., TTT has a defensive heuristic to control the center of the board and a similar strategy is viable in chess).

The final three requirements of the SoA ontology concern locus of control and agent communication. As previously mentioned, the current Recognition agent may be viewed as having centralized control authority, however the locus of control is passed to a subset of agents which form the active community within the society. The different forms of agent communication have been listed and in fact, the atomic nature of the agents requires them to interact and communicate with each other to solve the various domain tasks. This interaction is especially important between the Board agent and the Segment and Cell agents. The Segment and Cell agents each now very isolated and specific domain task (game) information which when interactively combined enables the resolution of the defined agent task.

Now that the implementation of the SoA ontology in a multi-agent society for games has been defined, the Game Society is evaluated for the functionality and performance of these agents for the specified domain. Implementation of the Game Society is done in Java. The runtime creation of the dynamic Cell and Segment agents for the problem solving community within the society is accomplished using the Java code in Figure 3.

url = new java.net.URL[]{new java.net.URL("file:"+ClassName+".java")};
java.net.URLClassLoader loader = new java.net.URLClassLoader(url);
Class cls = loader.loadClass(ClassName);
r = (java.lang.Object)cls.getMethod(MethodName,x).invoke(cls.newInstance(),o);

Figure 3. Java code for dynamic agents.

The Game Society currently works on the four games previously mentioned. The Game Society has perfect performance for the domain problem of sudoku, solving 100 percent of the sudoku problems given to the society. For the kakuro problem the Game Society currently solves almost all of the problems given to the society (98 percent) and work is progressing to resolve the agent interactions and use of Strategy agents to enable 100 percent performance for this domain problem as well. TTT if played correctly should always result in a draw. The Game Society has never lost a TTT game, with most games resulting in a draw, but occasionally winning if a human player plays at a suboptimal level. The initial testing of the TTT domain problem was accomplished by having a Game Society system play against another Game Society system, thus playing against another agent with similar knowledge. The 3D-TTT game is winnable and the Game Society wins approximately 60 percent of its games, which depends on two factors: if it is the starting player (the Game Society wins 100 percent of the games in which it starts) and the level of play of the opponent. Again, initial testing is accomplished by letting the society play against a copy of itself and then later against an experienced human player, who intentionally played at various skill levels. The initial starting board and the final board for sudoku and kakuro and the final games for the Game Society playing as the first and second player in TTT and 3D-TTT are displayed in the Appendix for the interested reader.

The response times of this distributed agent community approach to playing games are very fast, with no perceived (real-time) delay due to computation and interaction amongst the agents for each of the four solved games. Initial work with Caro has enabled the Game Society to play at a very nominal level and work is continuing to improve the Game Society’s performance in Caro.

5. Conclusion and future research

A new ontology for development of multi-agent systems called the Society of Agents based on Minsky’s [1] Society of Mind has been presented. This ontology is then utilized to develop a Society of Games multi-agent system for solving a wide variety of one and two person games with very high reliability and performance.

Phase one of the research has been completed to demonstrate the application of the SoA ontology in a Society of Games. Additional research will progress in two distinct phases. The first will be to increment the quantity and type of games playable by the Society of Games. As indicated work is currently progressing on solving the game Caro and future work will examine checkers, chess, shogi, and go. Additional games that rely on chance, such as
backgammon, monopoly, and others, but which have recognized strategies and heuristics will also be developed. These additional games may require the definition of additional atomic agents into the society and will increase the type and quantity of Strategy agents within the society.

The next foreseeable phase for the research to progress in is the development of a learning capability within the Game Society. Stone [16], in his address to the 2007 IJCAI conference, indicated that multi-agent systems need to incorporate learning if they are truly to be qualified as intelligent. Currently the Game Society does not learn from past games and relies on its own inherent knowledge from the Segment agents and various Strategy agents. The issue of short-term memory has been identified as a necessity for the SoA ontology. If the SoA ontology is to be used for development of “intelligent” multi-agent systems then a long-term memory component will need to be added and agents defined that will be able to utilize this long-term memory to analyze the efficacy of various Strategy agents within specific domain games and perhaps assist in the development of new Strategy agents dynamically. A longer-term memory will also greatly enhance the analogy Strategy agent’s applicability.

The SoA ontology is not limited to just game domains. The atomic localized control focus of multi-agent systems would be applicable across a wide variety of problem solving tasks including: supply chain management, production operations, logistics planning, etc.

6. References


7. Appendix

Output examples of the SoA Game Society

<table>
<thead>
<tr>
<th>7</th>
<th>9</th>
<th>9</th>
<th>8</th>
<th>4</th>
<th>7</th>
<th>5</th>
<th>1</th>
<th>9</th>
<th>3</th>
<th>6</th>
<th>8</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td>8</td>
<td>9</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

initial board SUDOKU solution board