Towards an Architecture for A-life Agents:II

Darryl N. DAVIS, T. CHALABI and B. BERBANK-GREEN

Department of Computer Science, University of Hull,

Kingston-upon-Hull, HU6 7RX, England.

D.N.Davis@dcs.hull.ac.uk

Abstract.

This paper presents our latest directions in developing an agent for playing the game of GO, based on Artificial Life perspectives. Our (empirical and design) investigations into complete agent architectures is being driven by the development of a theory of (artificial) mind. Rather than channel all our work into one application area, we are considering a number of different domains in order to further the theory (and its possible computational models). This work ranges from experiments with synthetic worlds, through decision support systems, image analysis and adversarial game playing. It is the latter that drives this paper, and builds on a number of concepts currently being addressed in the research areas of agents and artificial life. Our aim in this specific area is to investigate the types of architecture appropriate for developing an emergent game playing intelligence.

1. Introduction

The research of GO programs is still in its infancy, but we shall see that to bring GO programs to a level comparable with current chess programs, investigations of a totally different kind than used in computer chess are needed: John McCarthy.

The game of GO, which is known as Weiqi in China and Igo in Japan, was invented between 2500 BC and 4000 BC, as legend has it, by an Emperor to teach his son the arts of strategy and patience. The game has relatively straightforward rules that nonetheless give rise to a very complex game. GO is often compared with chess, and certainly GO enjoys at least a similar status and popularity in the East as chess does in the West. However, whilst recent efforts to write chess programs that can match the best human player have been successful, the best current GO playing programs play at a very rudimentary level [1]. In 1997 IBM's Deep Blue beat the World Champion Garry Kasparov in a convincing manner whilst Handtalk, the current computer GO champion, was squarely beaten by amateur players. GO has not have attracted the same level of attention from the computer related community as chess. Subsequently there is still a significant disparity between the level that computer programs play each game.

Heeding the results from the interesting work going on elsewhere [2],[3] we thought to tackle the challenge of designing a GO playing agent from an A-life perspective. Our initial aim was to see if the interaction of low-level (a-life) agencies enabled an emergent behaviour of benefit to the side of a game-playing agent. We hoped to demonstrate how low level organisation, capable of playing a worthy game, can emerge from the computational modelling of the properties of the game itself. We have subsequently developed these ideas and are now looking to design and build a heterogeneous agent society capable of playing the game.

2. The Game of GO

GO is a game of skill, played between two players on a board, comprising of a grid of horizontal and vertical lines. The full size board is usually 19 by 19 lines; smaller boards are used for quicker games. Here we describe the basic principles of the game. There is a vast body of theory and practice that is available for the aspiring student of the game [4].

Each of the players has a colour – white or black. Initially, the board is empty of stones. The players take turns to place a stone onto an intersection point on the grid. A stone interacts with each of its four contiguous points. When these points are not occupied they are known as the liberty points of the stone. If a stone has no liberty points i.e. if it is entirely surrounded by the opponent’s stones, then the piece is captured and removed from the board. This is the only time that a stone once placed is moved. A player cannot make a suicidal move by placing a stone on a point with no liberties.

Two stones are conceptually joined if they are of the same colour and occupy one of each other’s liberty points. This is known as a string or a unit. To capture a string all the liberty points of all the stones within the string have to be occupied i.e. the string has to be entirely surrounded. An eye is a formation of stones of special significance in the game of GO. When a stone-free point is completely surrounded by stones of one colour it is known as an eye. An opponent cannot place a stone on that point unless it is a capturing move i.e. unless placing the stone causes one or more of the stones that surround the point to be captured. If a string has two eyes then it cannot be captured, since to fill the first eye would be a suicidal move and is therefore forbidden. Strings that have or can form two eyes are termed ‘alive’. A string that is incapable of forming two eyes is termed ‘dead’ and can be captured. Experienced players always evaluate whether stones can be formed into strings, and whether the string is capable of being ‘alive’ or ‘dead’.

3. Computer GO and Computer Chess

Computer GO has so far significantly failed to reach a high-level of game playing. This is particularly apparent when compared with the success that Chess programs have achieved. GO is not susceptible to the same programming techniques that were so successful in Chess. To understand why, we need to take a closer look at the two games.

Chess programs combine a limited amount of chess knowledge with a massive capacity to search and evaluate. A chess computer examines each possible move and then sees what move the opponent could make. Each resulting position is evaluated, and the move that leads to the strongest position is selected. In chess terminology a move by each player is known as a ply. Good chess programs look seven plys ahead. On average there are about 40 legal moves available to a player. Fortunately ways have been found to significantly “prune” the resulting search tree, so that only a fraction are explored and evaluated. A look-ahead of seven plys still requires the evaluation of some 60 billion scenarios.

In contrast to Chess, the GO board starts empty and fills as the game progresses. At the start of a game there are 361 (19x19) possible moves. Toward the end of the game there will be considerably less. On average there are 200 positions on which a player could legally place a stone. This makes the branching factor of the search tree very large. IBM’s Deep Blue is capable of evaluating 200 million chess-board positions a second. Even at this speed, to handle ten thousand trillion positions, in a 7-ply look-ahead, would take a year and a half. Clearly then the size of the search space associated with GO, makes a brute force search approach implausible.

There is another problem. Evaluating chess positions is not massively difficult for a computer. At a crude level it can assess the strength of either side by comparing piece value, since normally a piece advantage leads to a victory. This can also be combined with some measure of the strength of position, and the threat to the King. Positions in GO are far less straightforward to discern. The aim of the game is to gain maximum territory. Assessing the current territory of a board frequently does not yield the true state of play, since placing a single stone may capture many stones and reverse territorial possession.

In GO, even relative beginners are able to perform deep, narrow look-ahead of say 30 moves. This frequently occurs in what are known as ‘ladder situations’, which are races between opposing groups of stones to surround one another. Beginners are able to look so far ahead because there is only one sensible point to make each move. Getting the computer to recognise this type of move is in itself a serious challenge.

Another difference between chess and GO is how the games are finished. In chess the game ends when a player resigns, or when a checkmate is achieved. Checkmate is a definite state of affairs that can be easily recognised. In GO the game finishes when both players choose to pass. They pass when they feel that placing a stone would not improve their territory in any way. Beginners frequently play beyond the optimal point at which an expert would stop. Existing GO programs display similar behaviour.

Chess is both strategic and tactical. Generally however, a tactical evaluation of the board based on piece quality correctly yields the likely winner. In GO there is not as strong a correlation between winning a tactical struggle over a group and winning the game. The GO program must measure the influence of a stone, but not solely in terms of its tactical importance. This is not a straightforward thing to do.

For all of the reasons listed above, we can safely conclude that GO cannot be programmed using the same methods that have yielded such success with chess. Chess programs use a brute-force search and evaluate algorithm, combined with heuristics to prune the search space. This works well with chess because the board size is small and chess is more tactical than strategic. GO, by comparison, has a much larger search space, and is by nature strategic.

4. Agents and Artificial Life

Agents can be defined in many ways. There is no one universally correct or acceptable definition [5]. An agent can be defined (simply) as an autonomous entity that can sense and act upon its environment. Such simple agents are analogous but not totally equivalent to orthodox automata. More sophisticated descriptions draw on concepts such as adaptation, communication, intentions, learning, social ability etc. Agent environments vary considerably, from synthetic worlds to those which robots inhabit through to more abstract worlds consisting of information and knowledge. The architecture and composition of an agent typically reflects its environment and the role(s) it plays within that environment; i.e. the challenge of an agent’s problem or niche space [6]. Our current research relies heavily on the concept of agency across a number of different domains, and categories of processes, agencies and agents. For example, we can consider our ongoing research into decision support systems [7] as that of an investigation into tightly and loosely coupled [8] agent communities making use of modern but relatively orthodox AI techniques. Again we can consider the investigation of (co-operative and competitive) emergence in our simulation work [9] as complimentary to that discussed in research related to artificial societies [2]. This current paper provides an interim report on our ongoing (and relatively recent) investigations into the use of artificial life techniques within an agent framework.

One of the threads that draws this research together is that of computational architectures that allow or are designed to support computational intelligence. We suggest that, while no one specific agent architecture may be suited to all environments, our concept (and philosophy) of agency should be sufficiently flexible and amenable to variations in needs and capabilities that it allows us to model agents across a spectra of niche and design spaces. The computational techniques used to implement these designs can be varied to suit the requirements of the environment, the agent(s)'s tasks within those environments, and the constraints imposed by the agent's computational space. For example, it is possible to design (and implement) multi-agent communities based on a range of specialised agents that perform specific functions, or on a number of self-similar agents that are distinguishable only by their environmental stance and, as a consequence, their internal and external states.

A comparison can be made with the world of social insects. For example, on the whole, individual ants are identical to one another, excepting the difference between, for example, the guard, worker, drone and queen classes. By default, each member of a specific insect class performs the same functions and reacts similarly to the same stimuli. Individual differences between instances of any specific insect class (e.g. worker) result from the interactions of that specific insect, its navigation and resulting tasks within the colony's environment, i.e. that insect's environmental stance. These relatively simple biological agents, at least at an individual level, can communicate, using chemical and other type of signals, and seem to work towards common goals. Colonies display complex social behaviours and problem solving abilities, for example the creation of air conditioning systems in termite nests.

Using this metaphor it follows that an agent may itself be an environment within which other agents are at work. We will refer to such systems as macro agents. An example analogous to the insect world is the computer simulation of cities with agents (or actors) inhabiting those cities. Each city is itself a macro-agent within a society of cities. Such societies evolve by means of a number of processes at various levels. For example the interactions of the cities, the constraints the cities place on their inhabitants, the redefinition of the parameters of the cities by the actions of agents (or actors) within those cities and the roles the actors play in mitigating resource distribution within and between cities [3]. Such modelling strategies may be appropriate in computationally expensive search spaces as a way of deferring computational trajectories through such spaces to higher and higher levels of abstraction. At that point where no further deferment can be obtained, the computational model may be displaying sufficiently intelligent behaviour that the search space problem becomes tractable or irrelevant. This is one of our initial aims in using a-life techniques. We hope that the computational cost of this is preferable to traversing the search space.

5. A-Life, Agents and GO

Instead of directly imposing human models of how to play the game of GO, we chose to use simple agencies, integer-valued Cellular Automata (CA), to represent all the available positions (freedom points) on a GO board. These simple agencies communicate to each other about their state and their local environment. By placing a stone on the board, the state of the board as a whole, at a number of levels of abstraction, is changed. This change is a result of the interaction of all the agencies that change state as a result of the placing of a stone. Part of the communication between agencies can be seen as landscaping the board space. For instance, placing a white stone will add ‘height’ to that freedom point, and subsequently raise the surrounding ‘terrain’ by a function of the stone’s ‘height’. This would create a small ‘hill’ in the model. A black stone will decrease the ‘height’, creating a local ‘basin’. Areas of zero height (i.e. neither increased nor decreased) can be considered neutral. A quite complex terrain ensues from the interaction of freedom points, stone placement and stone adjacency. This is analogous to the influence functions in more orthodox GO programs [1].

Our first implementation was perceived to produce adequate opening moves but failed to play a satisfactory game (i.e. it repeatedly fails to win). In adversarial game playing, there are a number of possible levels of analysis, and types of play, ranging from low-level (almost haphazard) moves to high-level strategic ploys. While we may perceive any sequence of moves as saving, aggressive, defensive or territorial, the emergence of such strategic concepts from these very simple artificial life agencies was not forthcoming.

We then considered a more complex agent consisting of multiple terrain models to further explore the limits of this approach. We wondered if a more effective and tactical game-playing agency would emerge through increasing the complexity of the low-level processes within the agent. We devised CA representations that modelled a number of boards to help the agent in determining its next move. One board represented default territory classifications, for example the recognised *fuseki* positions. A number of alternative liberty models plus maps that relate solely to the impact of placing a stone on the board (hill or basin forming) were investigated along with hybrid (ΔCA) maps that combine some of these models.

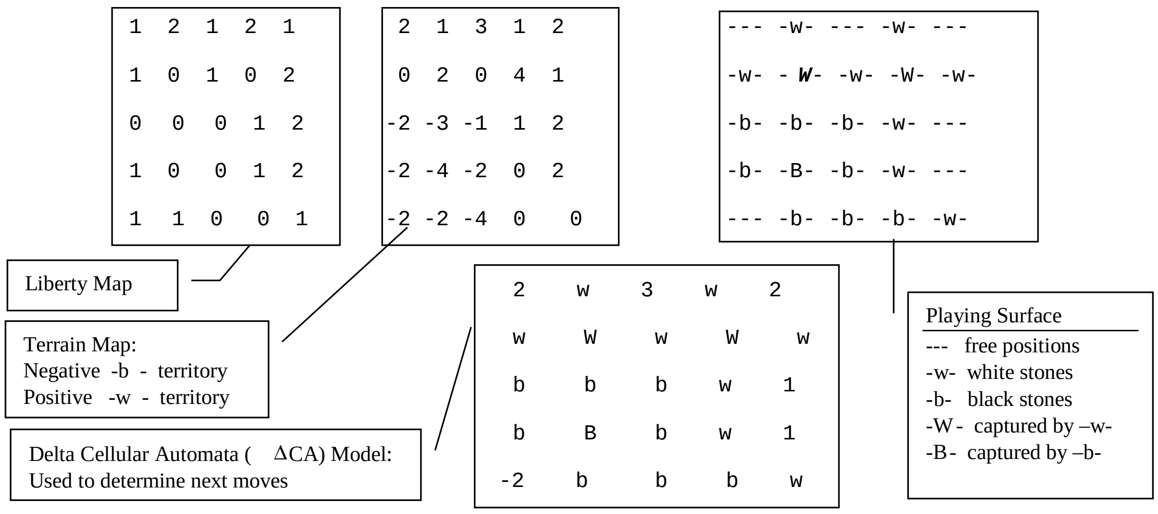

Figure 1. Multiple Terrain CA model. White (the computer) is about to win on a highly simplified board.

Figure 1 shows an example of a game that the agent system (playing white) won. In this game five separate boards were kept: the playing surface, a fuseki model, a liberty model, a terrain model, and a decision board (the ΔCA board). The agent system can play white or black (or both), and decides upon its next move using the decision board. If it is playing black it looks for the deepest terrain (i.e. most negative ΔCA position), and for the highest terrain (most positive ΔCA position) if playing white. As the decision board may have multiple best potential moves, some simple rules are used to resolve any conflict. These rules could be replaced or extended with further A-life mechanisms as discussed below. This relatively naïve approach to modelling the game displays emergent behaviour of benefit to the agent: Good opening stone placements; the building of non-empty units, i.e. geometrical arrangements of stones that contain internal freedom points; stone capture; and even string capture.

This system with its multiple CA-based strategies played the game more capably than the first system and even demonstrated some limited success in attacking isolated opposing strings. A number of further improvements could be made to improve its game playing. For example the terrain model could evolve with the game and relate to the recognised different stages of a good game of GO. However the multiple board model mostly failed to demonstrate these more sophisticated game-playing strategies. Such strategic ploys, while grounded in specific sequences of moves which vary from game to game, and within a game as the nature of the current game changes, are qualities that may require modelling as more persistent, global processes. Other work on strategies as emergent properties of tactical behaviours seems to be similarly inconclusive [10].

If we were to continue to adopt a purist a-life approach, more complex computational metaphors are required. For example, we can consider strings of stones to be a kind of multi-cellular agent, for example a synthetic and irregular coelenterata. Cellular automata representing single stones and empty spaces then become contributory elements of these macro-agents which need to grow (i.e. extend their territory), breathe (extend whilst enclosing empty board positions), eat (extend whilst capturing opposing stones) and meet (extend towards other strings of the same colour). Rather than reproduce in an orthodox sense, such macro-agents become like cities that merge or coalesce, or deposit spores in areas they aim to frequent. Our preliminary work in this specific area [11] has brought mixed results. Our analysis of the MCA work suggested that we needed to consider strings. Our analysis of the string-based CA's suggest we also need to consider a-life mechanisms capable of representing computational active regions and territories. This will require a more sophisticated macro agent architecture capable of utilising these different representational levels as required. The consequences of this are now considered.

6. Future Work

The design sketched in figure 2 shows an agent architecture of the type we are moving towards. This combines multiple communities of a-life agents, artificial neural nets and simple reasoning over propositions. In this type of macro-agent architecture, there exist many competing and co-operating a-life communities, at different levels of abstraction.

Again as in some of our other work, this builds upon the dynamic systems approach to cognition [12]. From this perspective, cognition is viewed as the changing focus of a number of interacting (sometimes extant, sometimes dormant) processes. Any particular reasoning or problem-solving capability can be tuned to a number of related, but different tasks, and that the achievement of complex tasks requires the use of many qualitatively different capabilities. The appropriateness of the actions suggested by any of the base level agents can be measured in terms of the needs of the individual and collective communities; i.e. what is good for the micro and macro-agent. The agencies responsible for determining this can be viewed as a synergistic harness of a-life mechanisms, neural networks and propositional logic coupled with constraint-based search mechanisms. Some combinations of agencies may be trained to recognise board patterns suited to attacking moves, others to defensive or space utilisation gambits (e.g. laddering). Such a decision path framework provides not only a-life driven high level processes, but in a dynamic architecture, where the active combination of agencies shifts as the macro-agent's problem space changes throughout a game, may provide a means for harnessing (potentially) positive emergent qualities and stabilising disruptive emergent behaviours.

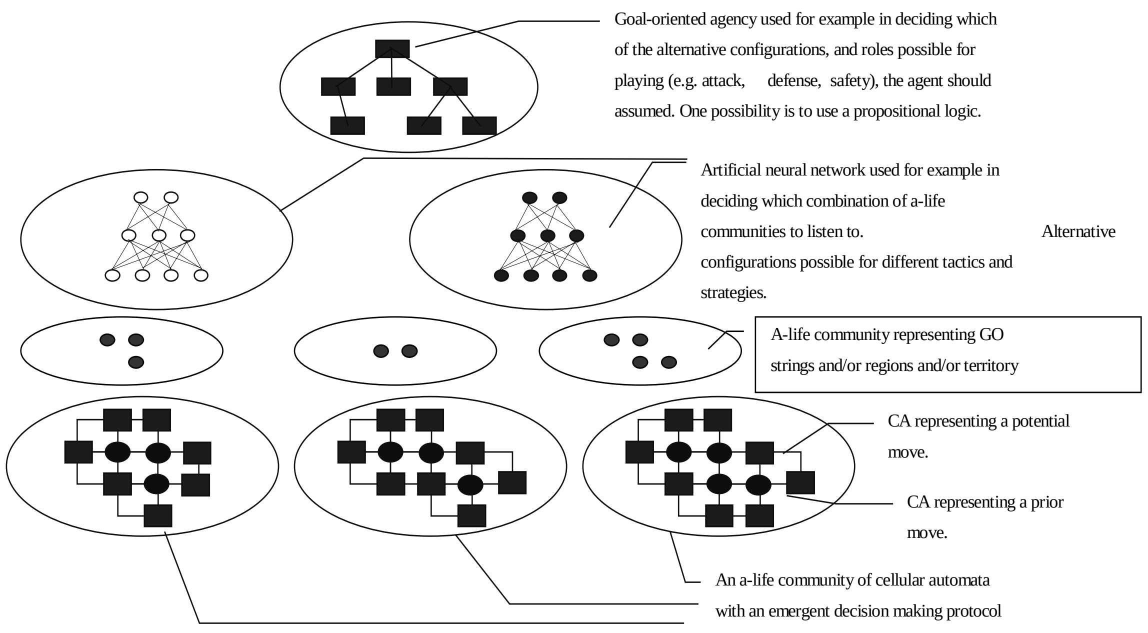
Goal-oriented agency used for example in deciding which of the alternative configurations, and roles possible for playing (e.g. attack, defense, safety), the agent should assumed. One possibility is to use a propositional logic.
Artificial neural network used for example in deciding which combination of a-life communities to listen to.
Alternative configurations possible for different tactics and strategies.
A-life community representing GO strings and/or regions and/or territory
CA representing a potential move.
CA representing a prior move.
An a-life community of cellular automata with an emergent decision making protocol

Figure 2. One (of many) possible architectures for an a-life based macro agent.

7. Discussion

The readings in [2] identify four levels or types of emergence: diachronic emergence, gestalt emergence, representational emergence, and adaptive or functional emergence (the most commonly referenced type); and moreover three categories of problems associated with emergence: sub-cognitive bias, behavioural bias, and individualistic bias. Diachronic emergence relates to evolutionary traits and is inapplicable to our current design and computational models, as we have yet to utilise evolutionary designs for this application. Gestalt emergence, however, seems to describe our aim of developing multiple a-life models that relate to the different levels of position abstraction that we can place on the game of GO. Adaptive emergence is clearly applicable to the current scenario and requires that our game playing agent, modelled using multiple agencies, contains appropriate mechanisms that enable it to take advantage of its currently most relevant processes. Of the different types of sub-cognitive bias, our assumption that collective (and co-operative) activity will emerge opportunistically is perhaps our most immediate concern. A consideration of individualistic bias suggests that adaptive emergence needs both feedback mechanisms from the micro (individual) level to the macro (societal) level and from the societal level to the individual level (a kind of focus of attention in a macro-agent). For instance an individual cell or cell matrix may suggest that a move at a specific location is beneficial. If such information is to be communicated to other more abstract computational models, it needs to be represented and communicated appropriately at those levels, and then enable feedback to other less abstract a-life processes. In such a way, a strategy may emerge from the adoption of a set of tactical moves modelled computationally as the currently attended and therefore extant a-life processes. This type of computational dynamism is analogous to situations described as representational emergence.

The architectural and design impact of these sometimes beneficial states is worth considering. As a general framework for computational and cognitive intelligence we have been developing a four-layer agent architecture that includes reflexive impulses, reactive processing, deliberative processing and meta-level (*reflective*) processes [6],[9]. Such a complete agent architecture may be able to manage perturbant patterns of processing and/or focus the processes associated with an agent (across the first three layers) to specific categories of tasks. A macro agent's need to monitor emergent patterns of behaviour, to classify them as (potentially) useful or disruptive and manage them is a clearly analogous situation. An underlying assumption here is that perturbant processes do not emerge at the reflective level, or that, if they do, they are constrained, by the rest of the agent architecture, as stalled or aborted processes. One major challenge, which future work should address, is how does such a macro agent recognise and harness these emergent behaviours, particularly where certain categories of behaviour are perceived to be useful in some but not all scenarios. Where emergence alone is sufficient, this is not a challenge, but where we want an agent to capitalise on such processes we face new challenges in agent design. The architecture sketched in figure 2 contains a-life reflexive (and reactive) processes, connectionist reactive processes and one deliberative agency. As we further develop this architecture, we will need to consider what are appropriate computational models for the reflective level. Whatever type of computational process we use, we need to proceed with the assumption that reflective agencies capable of displaying emergent patterns should not dictate but work in a synergistic manner with the other agencies within the macro-agent.

8. Summary

We have made some progress in developing an a-life-based agency capable of playing the game of GO. While our initial approaches show some promise, we have realised that to fully harness the desirable properties of A-life mechanisms we will need to investigate sophisticated agent architectures. The emergent and beneficial complexity of behaviour arising from interacting simply modelled agents can be best utilised in agents whose computational architecture allows them to recognise and utilise the more beneficial emergent states and resolve more disruptive emergent patterns. In effect, we need to design computational frameworks that enable an agent to harness functional and gestalt emergence. The suggestion is that the utilisation of the dynamic cognition approaches and the social metaphors appropriate to insect and the simulation of artificial societies may be appropriate.

GO presents us with a great computational challenge about which the agent and a-life metaphors offer a fresh perspective. We suggest that by combining such techniques and developing a methodology for creating such agents, we can make progress towards a computational intelligence of use in wider domains.

9. References

[1] J. Burmeister and J. Wiles. An Introduction to the Computer GO Field and Associated Internet Resources, *Technical Report 339, Department of Computer Science, University of Queensland 1995* http://www.psy.uq.edu.au/~jay/GO/CS-TR-339.html

[2] N. Gilbert and R. Conte, *Artificial Societies: The computer simulation of social life*, UCL Press, ISBN 1-85728-305-8, 1995.

[3] R. Hegselmann and A. Flache, Understanding complex social dynamics: A plea for cellular automata based modelling. *Journal of Artificial Societies and Social Simulation*, Vol. 1, ISSN: 1460-7425, 1998.

[4] O. Hideo 9 dan, *Opening Theory made Easy*, ISBN: 4-87187-036-7, Ishi Press, 1992.

[5] S.P. Franklin and A.G. Graesser, Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents, In: J.P. Muller, M.J. Wooldridge & N.R. Jennings (Eds.) *Intelligent Agents III*, ISBN: 3-540-62507-0, Springer-Verlag, Heidelberg, 1996.

[6] A. Sloman, Explorations in design space. In: *Proceedings 11th European Conference on AI*, Amsterdam, 1994.

[7] D.N. Davis and B. Sharp,. An Agent Framework for Decision Support in the Water Industry, *New Review of Applied Expert Systems, Vol.5*. ISSN: 1361-0244, Taylor Graham Publishing, 1999.

[8] L. R. Medsker, *Hybrid Intelligent Systems*, ISBN 0-7923-9588-3, Kluwer Academic Publishers, 1995.

[9] D.N. Davis, Synthetic Agents: Synthetic Minds?, *Frontiers in Cognitive Agents, IEEE International Conference on Systems, Man, and Cybernetics*, ISBN 0-7803-4781-1, IEEE Press, 1998.

[10] A. Dragoul, When Ants Play Chess (Or Can Strategies Emerge from Tactical Behaviours). In: Castelfranchi, C, & Muller, J.P, *From Reaction to Cognition*, ISBN: 3-540-60155-4, Verlag, 1995.

[11] T. Chalabi, An investigation into the suitability of modelling a GO-playing agent as a network of Cellular Automata, *MSc Thesis, School of Computing, University of Staffordshire.*

[12] R.F. Port and T. Van Gelder, Mind As Motion: Explorations in the Dynamics of Cognition, ISBN: 0-262-16150-8, MIT Press, 1995.