On Emergence of Scalable Tactical and Strategic Behaviour*

Mikhail Prokopenko, Marc Butler, Thomas Howard

Artificial Intelligence in e-Business Group
CSIRO Mathematical and Information Sciences
Locked Bag 17, North Ryde, NSW 1670, Australia

Abstract. The principle of behavioral programming [1] suggests to derive low-level controllers from symbolic high-level task descriptions in a predictable way. This paper presents an extension of the principle of behavioral programming — by identifying a feedback link between emergent behaviour and a scalable Deep Behaviour Projection (DBP) agent architecture. In addition, we introduce a new variant of the RoboCup Synthetic Soccer, called *Circular Soccer*. This variant simulates matches among multiple teams on a circular field, and extends the RoboCup Simulation towards strategic game-theoretic issues. Importantly, the *Circular Soccer* world provides a basis for an architecture scale-ability evaluation, and brings us closer to the idea of meta-game simulation.

1 Introduction

Behavioural approaches to artificial intelligence often feature situatedness of agents reacting to changes in environment and exhibiting emergent behaviour, instead of reliance on abstract representation and inferential reasoning [2, 5]. Tactical and strategic reasoning, however, would seem to require domain knowledge and a degree of multi-agent cooperation beyond the reach of situated behaviour-based agents. Over the last few years, it has become apparent that a unifying architecture, combining cognitive (top-down) and reactive (bottom-up) approaches, cannot be achieved by simply connecting higher and lower layers. It has been suggested in recent literature that a "middle-out" architecture [1] is required. The approach adopted in [1] follows the *behavioural programming* principle: "taking symbolic descriptions of tasks and predictably translating them into dynamic descriptions that can be composed out of lower-level controllers". The idea that reactive behaviours can be derived from (and importantly, can be proved to be correct with respect to) a higher-level theory follows an earlier approach — the situated automata framework [5] — in relating declarative agent representations and their provably correct situated behaviours.

While research and development work has progressed in both top-down and bottom-up directions, a systematic methodology for provably correct hierarchical architectures remains an important and open challenge. The view taken in this paper is that, rather than defining situated or tactical reasoning *ad hoc*, it is

* This paper subsumes the team description paper "Cyberoos2000: Experiments with Emergent Tactical Behaviour".

desirable to categorise agents according to their functionality and reactions to the environment, and identify corresponding classes of agent architectures. In other words, the principal target is a systematic description of increasing levels of agent reasoning abilities, where a given behaviour can “grow” into an expanded level, while every new level can be projected onto a deeper (more basic) behaviour. More precisely, our framework (referred to as **Deep Behaviour Projection** — DBP) supports two parallel and interdependent streams:

- a hierarchical behaviour-based agent architecture, and
- a corresponding hierarchy of logic-based action theories, each of which declaratively describes an agent type and provides a basis to prove certain agent properties.

These components complement each other in the following way:

- given a formal translation procedure, agent behaviours can be automatically derived from an action theory, and proven correct with respect to it;
- an observed emergent behaviour can be formally captured and retained by an action theory of a higher level, with subsequent translation into an embedded behaviour.

The derivability/provability link from an action theory to agent behaviours fully complies with the principle of behavioral programming. On the other hand, the feedback connection from emergent behaviour to a (meta-)action theory, and then to a (provably correct) derived behaviour on a higher level, extends this principle. Thus, a successful agent behaviour can be present in the architecture in two forms: implicit (emergent) and explicit (embedded). We believe that this duplication (or depth) provides necessary functional interchangeability, and allows the agent to “mediate” among related behaviours. The results reported in [11, 12, 13] formalise the DBP approach at the situated and basic tactical levels. This paper does not introduce new systematic models and formal correctness results, but rather presents the DBP approach, highlighting its biological motivation.

2 “Deep Behaviour Projection” Agent Architecture

While designing architectures for artificially intelligent agents, it is quite normal to draw parallels with the natural world — after all, human (and animal) intelligence is so far the only available example. For instance, a canonical problem of finding resources in unknown environment and collecting them at a specified location can be simulated as the foraging problem in ants' colonies [3] — where ants look for food and carry it towards their nest, laying down pheromones to indicate good paths to food locations. In this instance, the motivating biological example provided good intuition for experimental framework relating behavioural and genetic programming.

Motivated by biology, we intend to study increasing levels of complexity in agent behaviours and correlate these with enhanced architecture and reasoning abilities, sometimes in teamwork context. We begin at the *reactive* level — observing that certain sub-types of reactive behaviour are quite often neglected in the AI literature and clumped indiscriminately under that generic label. The previous work [11, 12] has formally identified and analysed the types of reactive behaviours that we believe correspond to very basic animal behaviors. In particular, *Clockwork*, *Tropistic* and *Hysteretic* agent classes were studied as examples of *situated* agents. Another series of agent classes has been grouped according to *tactical* abilities: *Task-Oriented* and *Process-Oriented* types. The developed hierarchy can be briefly summarised as follows:

$$\begin{array}{llll}
\langle C, S, E, & \mathit{sense} : C \to S, & & \mathit{response} : E \to C, \\
 & \mathit{timer} : C \to S, & & \mathit{command} : S \to E, \\
 & \mathit{tropistic\_behaviour} : S \to E, & & \\
I, & \mathit{hysteretic\_behaviour} : I \times S \to E, & & \mathit{update} : I \times S \to I, \\
T, & \mathit{decision} : I \times S \times T \to T, & & \mathit{combination} : T \to 2^H, \\
P, & \mathit{engage} : I \times S \times T \times P \to P, & & \mathit{tactics} : P \to 2^T \ \rangle,
\end{array}$$

where C is a communication channel type, S is a set of agent sensory states, E is a set of agent effectors, I is a set of internal agent states, T is a set of agent task states, P is a set of agent process states, and H denotes the set of $\mathit{hysteretic\_behaviour}$ instantiations $\{(i, s, e) : e = \mathit{hysteretic\_behaviour}(i, s)\}$.

The resulting architecture draws its expressive power from the situated automata and subsumption-style architectures, while retaining the rigour and clarity of logic-based representation. The Deep Behaviour Projection framework underlies this hierarchy and ensures that more advanced levels capture relevant behaviour more concisely than their deeper projections. Moreover, the depth in behaviour representation provides functional interchangeability among levels, and enables architecture scale-ability across domains. Our primary application domain is RoboCup Simulation League — an artificial multi-agent world [6], where the DBP framework provided a systematic support for design and full implementation of Cyberoos [11, 13]. Previous generations of Cyberoos developed under the DBP approach, captured certain types of situated behaviour (Cyberoos'98) and some basic classes of tactical behaviour (Cyberoos'99). Cyberoos2000 is the third "generation" designed in line with this framework. In particular, Cyberoos2000 focuses on exploring emergent tactical teamwork.

2.1 Situated Agents

A simplest perception-action feedback is implemented by the *Clockwork* agent, which is able to distinguish only between sensory states that have different time values, having no other sensors apart from a *timer*. Moreover, the *Clockwork* agent behaviour is predefined and is totally driven by time values. Like a clockwork mechanism, the *Clockwork* agent executes its fixed behaviour as a sequence of commands sent to the simulator at regular time points. Despite its almost *mechanical* simplicity, this agent type can be associated with very basic forms of cellular life driven by periodic biological cycles. In context of RoboCup, we found a pure *Clockwork* agent useful only in testing scenaria, where a player is expected to execute a given sequence of commands without synchronisation clashes (more than 1 command per simulation cycle) or stalls (no commands per simulation cycle). The activity in the *Tropistic* agent is characterised by a broader perception-action feedback represented by the *tropistic_behaviour* function. This agent reacts to its sensory inputs in a purely reflexive fashion. This kind of behaviour can be easily identified with plants, but sometimes even humans follow tropistic reflexes (eg., a hand reflexively retracts from a hot surface).

After many experiments with Cyberoos goalkeepers, we observed that a tropistic catch was the most effective behaviour in dangerous situations. However, the *Tropistic* agent may have only a partial capacity to distinguish degrees of risk (a sensory state may, for example, omit information on play mode at a given cycle — own “free_kick” or “play_on”). Therefore, a tropistic goalkeeper will try to catch a close ball all the time — unless this behaviour is subsumed by higher levels. Other important examples of tropistic behaviour exhibited by a Cyberoos agent are obstacle avoidance and ball chasing. It is worth pointing out that regardless of how instantiations of tropistic behaviour (tropistic rules) are developed — by elaborate programming and fine-tuning, genetic evolution, or reinforcement learning — their semantics remains simple and is captured by direct mapping from sensors to effectors.

However, such a direct mapping becomes conceptually and computationally cumbersome if the number of tropistic rules grows significantly — it becomes increasingly difficult to represent and encode each behaviour instantiation, and it takes a long time to match partial sensory states in a strictly sequential computational environment.

A *Hysteretic* agent is defined as a reactive agent maintaining internal state I and using it as well as sensory states S in activating effectors E, i.e. its activity is characterised by *hysteretic_behaviour*. An *update* function maps an internal state and an observation into the next internal state. Some animals seem to be proficient almost exclusively at this situated level, and yet may exhibit interesting behaviours. For example, the flocking behaviour of birds can be simulated totally at the hysteretic level with three simple rules: (i) maintain a minimum distance from other birds or other objects; (ii) match the velocity of birds in the neighbourhood; (iii) move towards the perceived centre of mass of the nearby birds [7]. Importantly, in order to behave hysteretically, an agent must maintain an internal state (containing, in the flocking example, variables for minimum distance, average velocity, distance to neighbourhood centre of mass, etc.). Faced with an obstacle, the simulated flock splits around and reunites past it. This is a classic example of *emergent*, rather than hard-coded, behaviour[2].

A Cyberoos2000 agent exhibits quite a few interesting examples of emergent hysteretic behaviour, eg., dribbling around opponents toward a target; intercepting a fast moving ball; resultant-vector passing; shooting at goal along a non-blocked path; etc. The *hysteretic_behaviour* is implemented as a (temporal) production system (TPS). Whenever the TPS fires a hysteretic rule, an atomic commands sequence is inserted into a queue for timely execution, inherited from the *Clockwork* level. In addition, the TPS monitors currently progressing actions, thus providing an explicit account of temporal continuity for actions with

[2] In fact, such flocking behaviour has been proven successful to some degree in the RoboCup Simulation domain — by YowAI team from Japan in 1999.

duration [14, 13], and allowing us to embed actions ramifications and interactions [12, 13]. For example, a dribbling action is suppressed while shooting or passing.

It is worth noting that this architecture allows us to easily express desired subsumption dependencies [2] between the *Hysteretic* and *Tropistic* levels, by an inhibition of the lower level behaviour if necessary. For instance, tropistic chase is suppressed if a teammate has possession of the ball.

2.2 Tactical Agents

The *behaviour* functions of these situated agents are not constrained. Sometimes however, it is desirable to disable all but a subset of behaviour instantiations (rules) — for example, when a tactical task requires concentration on a specific activity. The *Task-Oriented* agent type is intended to capture this feature — it incorporates a set of task states, and uses the *decision* function in selecting a subset of behaviour instantiations (a task) most appropriate at a particular internal state, given sensory inputs. A task activates only a subset of all possible rules by invoking the *combination* function. Furthermore, a *Process_Oriented* agent maintains a process state and is able to select an ordered subset of related tasks — *tactics*. Implementation of task-orientation requires some adjustments to the TPS. The TPS traces action rules whose actions may be in progress, and checks, in addition, whether a rule is valid with respect to a current task. The rules producing hysteretic behaviour mentioned in the previous section (dribbling, intercepting, etc.) are combined in corresponding tasks and can be selected by a Cyberoos2000 agent in real-time. For example, *zone playing* is implemented as a task, enabling relevant (hysteretic) rules for offside trap, making defensive blocks, cover zone, etc.

Task-orientation appears to be not only useful conceptually, but is also a practical functional element. Although it is clear that an elaborate *hysteretic behaviour* can achieve the same results as any given task-orientation, the latter captures patterns of emergent behaviour more concisely. We believe that appropriate task-orientation evolved in animals as well — to support and strengthen specialisation. One impressive example, directly related to tactical teamwork, is hunting behaviour of lions. It was found [9] that the Serengeti (Tanzania) lions most often work together when tackling difficult prey such as buffalo and zebra, but hunt alone in taking down easy prey. In Etosha (Namibia), however, lions specialise in catching the springbok — one of the fastest antelopes of all — in flat and open terrain. The research [9] has shown that “a single lion could never capture a springbok, and so the Etosha lions are persistently cooperative”. Interestingly, an analogy was drawn between Etosha lions hunting behaviour and a rugby team’s tactics, in which wings and centers move in at once to circle the ball, or prey. This “highly developed teamwork stands in sharp contrast to the disorganized hunting style of the Serengeti lions” [9].

First important lesson that can be drawn from this analogy is that **stable patterns of emergent behaviour are worth retaining** — in this instance, via suitable task-orientation. Secondly, the observed tendency in emergence of more complex tactical behaviours (eg., bird flocking to prey intercepting to prey circling) can be correlated with **appearance of new elements in agent architecture**. Put simply, when emergent behaviour struggles to fit into the existing scheme, extension of the framework is warranted! This was obvious in the introduction of internal state that allowed the hysteretic agent to situate itself better than tropistic one in relation to other agents and environment objects. The following observation exemplifies this tendency on a tactical teamwork level.

The easily recognised pattern of "kiddie soccer", where everyone on the team would chase the ball, could emerge as sub-optimal tactics on tropistic level — and could genetically evolve in the RoboCup Simulation domain as well [8]. We observed that *Hysteretic* Cyberoos agents exhibited another sub-optimal behaviour — solo dribbling towards the enemy goal. This kind of behaviour could emerge if there are no visible teammates and internal agent state does not keep track of them. Solo dribbling can be observed in "teenage soccer", and arguably is more efficient than the "kiddie" one, while being more demanding and challenging in terms of the individual skills required. It may also lead to a devastating consumption of the player stamina.

A simple tactic complementing the "solo dribbling" is the so-called "backing up" — the following closely on the teammate with the ball "to assist him, if required, or to take on the ball in case of him being attacked, or otherwise prevented from continuing his onward course" [4]. It is not surprising that this simple behaviour was considered a tactical triumph in the 1870s, when the football sport was often called "the dribbling game" and forward passes were disallowed. This behaviour can be programmed on the hysteretic level as well, if desired — by appropriately inhibiting tropistic chase and elaborating (disqualifying) other competing hysteretic rules like *cover zone*. However, with teamwork tactics progressing, it becomes inconvenient and computationally expensive to keep elaborating hysteretic rules. The lessons of emergent behaviour suggest again that a new concept is required (provided in this case by the agent task state).

One particular instance of tactical behaviour, emerging at the hysteretic level, is making "defensive blocks" against an opponent dribbling towards the team goal — one defender chases the ball and tries to kick it away, while another runs towards some point on the line between the opponent and the goal. Arguably, making defensive blocks is similar to basic hunting tactics of the Serengeti lions. This tactical behaviour emerges at the hysteretic level when the second defender (lion) recognises that there is no need to directly attack the opponent (prey) — more precisely, when agent's internal state (containing the fact that a teammate presses the opponent with the ball) resolves to subsume the tropistic chase. Again, a more efficient implementation of this tactics can be achieved with appropriate task-orientation, enabling only certain hysteretic rules.

In short, task-orientation makes tactical teamwork more explicit — when agents' task states are complimentary, collaboration becomes more coherent. Importantly, a task does not fix agent behaviour, but rather constrains it within a set of relevant behaviour instantiations (rules). The task-orientation of the Serengeti lions would make them less successful hunters of the springbok: although their running and catching skills are probably as good as Etosha lions' ones, the latter kind packaged the skills tactically differently.

2.3 Towards Emergence of Domain Models

The *Task-Oriented* agent is capable of performing certain tactical elements in real-time by activating only a subset of its behaviour instantiations, and thus concentrating only on a specified task, possibly with some assistance from other agent(s). Upon making a new *decision*, the agent switches to another task. In general, there is no dependency or continuity between consecutive tasks. This is quite suitable in dynamic situations requiring a swift reaction. However, in some cases it is desirable to exhibit a coherent behaviour during longer intervals.

We are currently pursuing several complementary directions potentially leading to emergence of such a deliberative behaviour (when an agent is engaged in an activity requiring several tasks).

One direction centers on the notion of process — a set of possible tasks without a precise sequential or tree-like ordering. In other words, process-orientation is not restricted to be a result of pure deliberation. An appropriate tactical scheme comprising a few tactical elements may simply suggest for an agent a possible subset of decisions, leaving some of them optional. For example, a penetration through centre of an opponent penalty area may require from agent(s) to employ a certain tactics — a certain set of elementary tactical tasks (dummy-run, wall-pass, short-range dribbling) — and disregard *for a while* another set of tasks. It is worth noting that whereas a team's tactical formation is typically a static view of responsibilities and relationships, process-orientation is a dynamic view of how this formation delivers tactical solutions. A *Process_Oriented* agent maintains a process state and is able to select an ordered subset of tasks — *tactics* — given a particular internal state, sensory inputs, current task and *engaged* process. Ideally, a *Process-Oriented* agent should be capable of consolidating related tasks into coherent processes.

Another promising direction towards deliberation introduces a domain model into the architecture. The idea of having a "world model" directly represented in the architecture is intuitively very appealing. However, we believe that "world model" should grow incrementally instead of being inserted and glued to other elements. In other words, our preference is to observe and analyse examples of emergent behaviours which potentially make use of the domain model. At the moment, Cyberoos2000 agents do not use world models and inter-agent communication, relying entirely on deep reactive behaviour and emergent tactics.

3 Evaluating Architecture Scale-ability

In order to comprehensively evaluate an intelligent agent architecture, it is desirable to compare produced behaviours under different circumstances, and in various domains. While RoboCup Simulation creates quite a challenging synthetic soccer world, current research may still be subject to a potential methodological bias. It is conceivable that designers introduce results of their own understanding of the domain *directly* into the agent architecture. Consequently, it may become rather unclear how a given architecture would scale to a reasonably different domain.

To alleviate this situation, we introduce here an extension of the RoboCup Synthetic Soccer, called a *Circular Soccer*. This variant is very similar in terms of rules to the one currently in use in the RoboCup Simulation League, with some exceptions: the number of teams competing in a single match may be greater than 2, and the stadium field is circular rather than rectangular (Figure 1 depicts the case of three teams). Other differences include a modified implementation of the off-side rules, and corner kicks. Another sub-variant, a *Closed Circular Soccer*, simulates the field boundary as a solid circular wall, based on elastic wall-ball collisions.

In our view, the *Circular Soccer* may provide a domain, where scale-ability of agent behaviours and tactical teamwork can be verified. Ideally, a team of intelligent agents should be capable of adapting to the *Circular Soccer* world, with minimal design modifications. We would argue that allowed alterations on situated level may include, for example, visual information parsing routines and triangulation algorithm(s). Tactical scale-ability can be really tested by an amount of changes required to make a team operational, and ultimately successful. Arguably, if no tactical behaviour emerges after the deployment in the new world, the architecture fails to deliver the required flexibility. Of course, we do not intend to restrict any modifications. However, the main question translates into how easy it is to re-combine tactical behaviour instantiations in order to achieve coherent tactics.

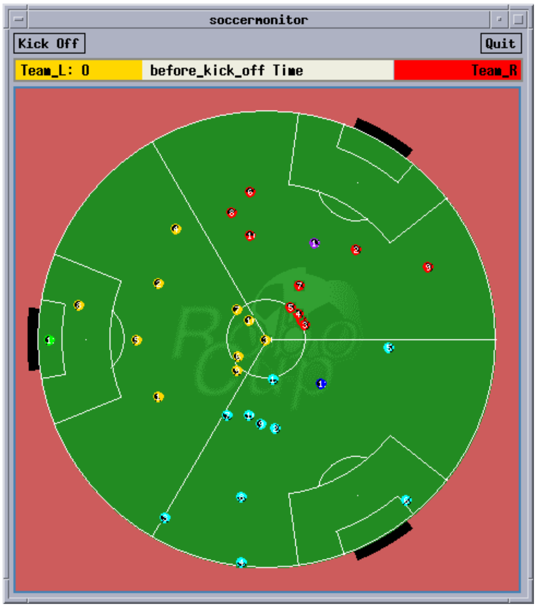

Fig. 1. *Circular Soccer*: a three teams case

Moreover, the *Circular Soccer* world provides a very interesting platform for testing *strategic* behaviour with a game-theoretic flavour. Even three competing teams, for example, bring cooperation and competition to a higher level. Let us assume the following simple zero-sum score assignment mechanism for the teams

A, B and C. If the team A scores against B, then the score of A is incremented by 1 and the score of B is decremented by 1, while the score of C is unchanged. At the end, teams are ranked by a single-number score, and the winner is the team with the most positive score (a team-to-team score is important only as a tie-breaker). For example, after a game where A scored once against B and once against C, and C scored four times against B, the overall score would be $A : +2$, $B : -5$ and $C : +3$. The team C is the winner, despite losing to A in a team-to-team contest.

Strategic behaviour becomes evident when players of one team, let us say team A, reason about cooperating *with* other team *against* the third. For example, in the beginning of the example match it made no difference if team A cooperated in attack with team C against B. However, at the end, team A should have cooperated in defense with team B, trying to prevent C from scoring the winning fourth goal.

In short, the *Circular Soccer* world extends the RoboCup Simulation towards strategic game-theoretic issues, and provides a basis for an architecture scale-ability evaluation. Moreover, it brings us closer to the idea of meta-game simulation in RoboCup domain. Meta-game programming is, in general, a task of writing a program that plays a game of a certain domain class, provided only with the rules of a game [10]. Rather than designing programs to play an existing game known in advance, the meta-game approach suggests to design programs to play new games, from a well-defined class, taking as input only the rules of the games. As only the class of games is known in advance, a degree of designers bias is eliminated, and meta-game playing programs are required to perform game-specific optimisations without designers assistance [10].

The concept of meta-game simulation sounds very appealing to us, as it makes the architecture scale-ability evaluation almost explicit. Put simply, a single-game behaviour will perform more strongly if it is well-tuned to the game, while a meta-game behaviour will be stronger if it is based on a more scalable architecture.

4 Conclusions

In this paper we attempted to illustrate emergence of new behavioural patterns as a good indication for inclusion of new elements in our agent architecture. This tendency has been observed while moving from *Clockwork* to *Tropistic* to *Hysteretic* to *Task-Oriented* to *Process-Oriented* agents.

We maintain that a complexity of an agent architecture is relative: for any elaborate agent type, it is possible to define more concisely another agent type with the same external behaviour. Hence, an agent has an embedded choice as to which one of related hierarchical levels should assume control to better suit external environment. If successful, such interchangeability among levels offers useful (and potentially vital) duplication and deep functional flexibility. In summary, the “middle-out” layer is required between any two levels of an intelligent agent architecture — and animals (including humans) seem to maintain an expertise and abilities providing just that. Given a formal framework (such as

DBP), we can attempt to logically prove that two agent types produce identical emergent behaviour — i.e., the *interchangeability* can be proven correct in terms of external dynamics. Therefore, rather than searching for a mysterious hub connecting "reactive behaviour" and "cognitive skills", we should identify and study dialectic relations between emergent behaviour and elements of agent architectures. This might allow us to link architecture design and behavioural programming in a more systematic, concise and predictable way.

References

1. Michael S. Branicky. Behavioural Programming. In AAAI Technical Report SS-99-05, the AAAI-99 Spring Symposium on Hybrid Systems and AI, 29–34, Stanford, 1999.
2. Rodney A. Brooks. Intelligence Without Reason. In Proceedings of the 12th Int'l Joint Conference on Artificial Intelligence, 569–595 Morgan Kaufmann, 1991.
3. Stephane Calderoni and Pierre Marcenac. Genetic Programming for Automatic Design of Self-Adaptive Robots. Genetic Programming, Springer-Verlag Lecture Notes in Computer Science, Vol. 1391.
4. Paul Gardner. The Simplest Game. The Intelligent Fan's Guide to the World of Soccer. MacMillian USA, 178–179, 1996.
5. Leslie P. Kaelbling and Stanley J. Rosenschein. Action and planning in embedded agents. In Maes, P. (ed) Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back, 35–48, MIT/Elsevier, 1990.
6. Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela M. Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda and Minoru Asada. The RoboCup Synthetic Agent Challenge. In Proceedings of the 15th International Joint Conference on Artificial Intelligence, 1997.
7. Stephen Levy. Artificial Life. The Quest for a New Creation. New York, 76–80, 1992.
8. Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson and James Hendler. Co-evolving Soccer Softbot Team Coordination with Genetic Programming In RoboCup-97: Robot Soccer World Cup I (Lecture Notes in Artificial Intelligence No. 1395), H. Kitano, ed. Berlin: Springer-Verlag, 398–411, 1998.
9. Craig Packer and Anne E. Pusey. Divided We Fall: Cooperation among Lions. Scientific American magazine, May, 1997.
10. Barney Pell. Metagame: A New Challenge for Games and Learning. In H.J. van den Herik and L.V. Allis, (ed), Heuristic Programming in Artificial Intelligence 3, 1992.
11. Mikhail Prokopenko, Ryszard Kowalczyk, Maria Lee and Wai-Yat Wong. Designing and Modelling Situated Agents Systematically: Cyberoos'98. In Proceedings of the PRICAI-98 Workshop on RoboCup, 75–89, Singapore, 1998.
12. Mikhail Prokopenko. Situated Reasoning in Multi-Agent Systems. In AAAI Tech. Report SS-99-05, the AAAI-99 Spring Symp. on Hybrid Systems and AI, 158–163, Stanford, 1999.
13. Mikhail Prokopenko and Marc Butler. Tactical Reasoning in Synthetic Multi-Agent Systems: a Case Study. In Proceedings of the IJCAI-99 Workshop on Nonmonotonic Reasoning, Action and Change, 57–64, Stockholm, 1999.
14. Erik Sandewall. Towards the Validation of High-level Action Descriptions from their Low-level Definitions. Linköping electronic articles in Computer and Information science, (1):4 1996.