

Between Teaching and Learning: Development of the Team 'Mainz Rolling Brains' for the Simulation League of RoboCup '99

Mainz-Rolling-Brains

Daniel Polani, Thomas Uthmann

DP, TU: Johannes Gutenberg-Universität Mainz

Abstract. *The development of our team for RoboCup '99 is mainly oriented towards a transparent way of transferring explicit knowledge into the agent control and its combination with learning algorithms capable of fine-tuning the acquired skills. The explicit knowledge is formulated in terms of rules, the non-explicit knowledge is to be realized as a set of parameters adapted by hierarchical reinforcement learning and by rule evolution. The teaching process for the implicit learning is not determined by a simple fixed reinforcement return, but by a — possibly complex — agent that represents a human or an automated coach.*

1 Introduction

A classical dilemma of AI is to bridge the gap between explicit and implicitly learnt knowledge. In a typical non-AI application not only the goal, but also the subgoals are explicitly formulated, as well as the relation between a goal and its subgoals. Such systems are — at least in principle — clearly interpretable and have a well-understood dynamics implemented by the developer. An AI system, however, is seldom used in a context where the conditions mentioned above are fully met. While it is still possible that the ultimate goal has an explicit formulation, this need not be the case for possible subgoals. Of course an AI program must be represented as an explicit algorithm, thus typical AI applications contain a lot of tunable parameters which have the purpose to fit the implicit knowledge about the handling of a given situation into the rigid frame of an algorithmical formulation.

Reinforcement Learning (RL) is one way of finding strategies to reach a given goal without having to specify explicitly how this goal should be achieved [3, 7]. Here the parameters are learned by interaction with the world which determines the goal. For RL algorithms like dynamic programming or Q -learning an abstract representation of state space is required adequate for the application at hand. All relevantly distinct states and actions in those states must be modeled. In larger problems, e.g. a multi-agent system like RoboCup [4], this implies a large size of the state space and a

slow learning or, either, the necessity of the development of a sophisticated function approximation representation which may, in turn, pose problems to an understanding of the mechanisms learnt. In addition, for a task like soccer it is a plausible approach not to require that strategic as well as tactical combinations for an agent's behavior should be learned from scratch. Instead, it makes sense to map existing human ("cultural") knowledge about game strategies into an interpretable structure, where learning algorithms are mainly used to *improve* and not to find specific parameter settings.

Because of the limited space we will concentrate on describing the newer aspects of our agent team. While the important modules of world model and synchronization have been completely rewritten since RoboCup 98, their concepts have not changed in an essential way. This year's development concentrates on an improved adaptation of the rule system.

2 The Rule System

First, we wish to give a short overview over the hierarchical rule mechanism providing the skeleton for the agent control. Each rule in the hierarchical rule tree consists of two parts: the first part is the *condition* for firing the rule, the second part is an action to be performed. In a *terminal rule* this action may be an *elementary action* (e.g. a basic agent skill), in a *meta-rule* it consists the action is a set of subrules (Fig. 1).

The rule tree begins with an initial meta-rule. The condition for each subrule in the meta-rule set is evaluated giving a priority for the given subrule. The subrule with the highest priority is selected from the set. If its action is elementary, it is performed, if, in turn, it is a meta-rule, the described rule activation algorithm is recursively applied to the set of sub-rules. Thus it is possible to implement conditions triggering higher level behavior and lower level reactive behavior in the same framework.

3 A Framework for Teaching Learning Agents

The knowledge made explicit by the human developer and transferred into the agents in terms of rules cannot cover all possible constellations in a complex environment. The rules have to be refined by *half-implicit* and *implicit* adaptive mechanisms. A fully implicit mechanism, for instance, would be covered by a typical episodic or continuous reinforcement learning algorithm.

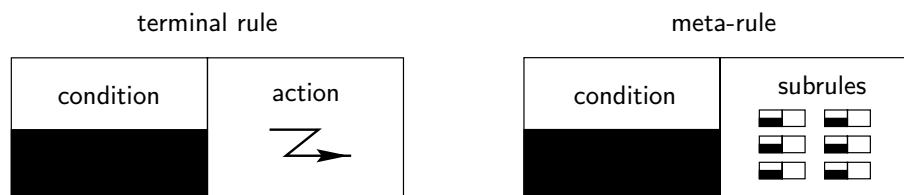


Figure 1: Rule concept

In what we call the half-implicit case, however, the behavior of the agents is corrected by a coach that may interrupt and assess an action sequence which can be afterwards repeated. This coach may be a human or an adequately designed automatic agent itself. This approach differs from the usual view of reinforcement learning problems in that the end of an episode and the timing of the reward is not a simple fixed reward model, but itself realized as an agent deciding on the level and timing of reward. The intention is to train partial behaviors in a similar way as done with human players.

One part of the task is to identify an agent’s mistake when it happens. If the external coach notes a mistake, he notifies the agent. The situation is then replayed and can be corrected until the coach is satisfied. A mistake can be caused by applying the wrong action in the right rule, by applying the wrong rule (e.g. because the priorities of other rules of the same level are set to a wrong value) or because the consistency of the agent’s world model with the simulation state in the soccer server is inadequate to trigger the right rule. The basis of the approach is to identify the reason that led to the concrete instance of failure. The first two points could be partly dealt with using learning mechanisms related to Q -learning (e.g. MAXQ [1]), the third one requires explicit introspection by the developer to check for inadequate representation of sensory data.

While in principle RoboCup implies an episodic view of the reinforcement learning task, it is not necessarily an advantage to enforce this view, since the episodes can be very long and backtracking the sequences that lead to a well-defined success or failure state (i.e. a goal) by a learning method related to Q -learning may prove inefficient. More precisely, considering a reward only at the end of an episode where one knows whether the strategy has been successful or not is not enough. For an efficient learning a more “informed” reward function is needed that does not only reward at the end of an episode but rewards as often as possible. This reward function plays the role of a coach that criticizes the moves. As we work on a hierarchical rule system, the use of a multidimensional reward function is adequate. The different components of the reward function are applied to the different levels of the hierarchy and reflect the different kind of task level that have to be solved to learn the problem. This approach allows a way to impose more structure on the learning process and by this to accelerate it.

A second approach which is investigated in parallel is the use of a rule evolution mechanism inspired by different paradigms of neuro-evolution [2, 5, 6]. In these approaches, unlike in many other neuro-evolutionary approaches, not a population of neural *networks* is evolved. Instead the evolution takes place on the level of a population of substructures, e.g. of single neurons. We transfer this idea from *neurons* to *rules*, i.e. we perform an evolution on the level of rules (and *not* on the level of complete rule trees).

Any rule, as above, consists of conditions triggering it and of actions and/or rules following it. Any condition and action is fine-tuned by certain parameters and the particular assembly of conditions and actions constitutes a given rule. Inspired by the SANE and EuSANE approach [5, 6], we co-evolve two things: one the one hand we optimize the parameters determining the condition and actions, on the other hand we evolve their arrangement into a given rule. This arrangement is also called *blueprint*, following the SANE nomenclature. The fitness is given by solving given tasks and, similarly to aforementioned approach, we attempt to use a structured fitness function, i.e. a multi-valued fitness function for optimization to give the al-

gorithm a possibility to assimilate domain knowledge. Another important feature of this approach is the inherent ability to build hierarchical representations by evolution. A rule blueprint need not have action primitives as consequences, but instead a blueprint, i.e. another rule. This opens the possibility to build up an increasingly more complex hierarchy of rules that is compatible with a hand-crafted rule system underlying our general rule architecture. Therefore it is possible to mix explicit and implicit knowledge in a transparent way. One focus of our current work is directed at obtaining systematic results for the rule-evolution approach.

Acknowledgements

We wish to mention all the developers of the team: Axel Arnold, Erich Kutschinski, Christian Meyer, Götz Schwandtner, Manuel Gauer, Birgit Schappel, Frank Schulz, Ralf Schmitt, Peter Dauscher, Tobias Jung, Tobias Hummrich, Sebastian Oehm, Peter Hubert, Achim Liese, Peter Faiß, Oliver Stein and Michael Hawlitzki. Also we wish to thank Risto Miikkulainen for fruitful discussions on neuro-evolution. In addition, we wish to express our gratitude to SerCon GmbH Germany for sponsoring our team.

References

- [1] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Submitted to Machine Learning*, 1999.
- [2] G. Edelman. *Neural Darwinism: Theory of Neuronal Group Selection*. Basic Books, New York, 1987.
- [3] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, May 1996.
- [4] Hitoshi Matsubara, Itsuki Noda, and Kazuo Hiraki. Learning of cooperative actions in multi-agent systems: a case study of pass in soccer. In *AAAI-96 Spring Symposium on Adaptation, Coevolution and Learning in Multi-Agent Systems*, pages 63–67, Mar 1996.
- [5] D. Moriarty and R. Miikkulainen. Forming neural networks through efficient and adaptive co-evolution. *Evolutionary Computation*, 5:373–399, 1997.
- [6] D. Polani and R. Miikkulainen. Fast reinforcement learning through eugenic neuro-evolution. Technical Report AI99-277, Department of Computer Sciences, The University of Texas at Austin, January 1999.
- [7] R. S. Sutton and A. G. Barto. *Reinforcement Learning*. Bradford, 1998.