Extended Behavior Networks for the magmaFreiburg Soccer Team

Klaus Dorer

Centre for Cognitive Science
Institute for Computer Science and Social Research
Albert-Ludwigs-University Freiburg, Germany
klaus@cognition.iig.uni-freiburg.de

Abstract. In this paper we descibe the process of action control used by the agents of the magmaFreiburg team. It is based on extended behavior networks, which add situation-dependent motivational influence on the agent and extend original behavior networks to exploit information from continuous domains. Advantages of the original networks, such as reactivity, planning capabilities, robustness, accountance for multiple goals and its cheap and distributed calculations are maintained.

1 Extended Behavior Networks

Maes [4–6] suggested a mechanism for action selection in dynamic and unpredictable domains based on so-called behavior networks.

Although Maes' networks do work in continuous domains, they do not exploit the additional information provided by continuous states. Similarly, though there are mechanisms to distinguish different types of goals in MASM, there are no means to support goals with a continuous truth state (like 'have stamina') to become increasingly demanding the less they are satisfied.

We propose an extended version of [5] and [3] that takes the step from discrete to continuous domains by introducing real-valued propositions. It also allows for advanced motivational control by situation-dependent goals. In [2] we have shown that the extensions proposed showed significantly higher success in the RoboCup domain [7].

1.1 Behavior Network Description

Behavior networks consist of the goals of the agent, its capabilities represented by so-called competence modules and its perceptions [4].

A *goal* of an extended behavior network is represented by the (static) *importance* value of the goal, a *goal condition* which describes the situation in which the goal is satisfied, and a *relevance condition* whose truth value represents the (dynamic) relevance of the goal.

Relevance conditions are introduced to model different types of goals. Maintenance goals, i.e. the motivation to preserve a certain state (e.g. have stamina)

can be achieved by adding a relevance condition ('stamina low') that increases the relevance of a goal once stamina decreases. The more the current state diverges from the goal state, the more urgent the goal becomes. Achievement goals, i.e. the motivation to reach a certain state, on the other hand, are more relevant the closer the agent is to the goal. This can be realized by adding a relevance condition (e.g. 'close to ball' to the goal 'have ball') whose truth value increases when getting closer to the goal. Static importance and dynamic relevance are combined by multiplication to calculate the utility of the goal.

A *competence module* consists of the *behavior*, which is executed once the module is selected for execution, a list of *preconditions* to determine the module's executability, a list of *effects* and their expectation values expected to occur after execution of this module and an *activity* value indicating the module's contribution to reach goals.

Perceptions are real-valued propositions representing the truth value of different states of the environment and the agent. The truth value of a proposition 'have stamina', for example, ranges from 0 (no stamina) to 1 (full stamina). Perceptions are used to calculate the executability of a competence module, the relevance of a goal and to control activation spreading (see following section).

1.2 Activation Spreading

To achieve goal-directed behavior, competence modules are connected in a network to receive activation from goals and other modules [4]. High activation improves a module's probability of being executed.

A competence module receives activation from a goal if it has an effect that satisfies the goal. Activation by goals is used to achieve goal-directed behavior by preferring modules that satisfy one or more goals. A competence module can also be inhibited by a goal if the module has an effect preventing the goal from being satisfied. Inhibition is used to suppress behaviors with unwanted effects.

If a module is not executable, it spreads activation to those modules that can satisfy the false precondition. The unsatisfied precondition becomes a subgoal of the network. The less the precondition is satisfied, the more activation is spread to other modules. A module can also be inhibited by other modules if they undo an already satisfied precondition of it. For a more detailed description see [2].

1.3 Behavior Selection

Behavior selection is done in a cycle containing the following steps [5]:

1. Calculate the activation of each module as described above.
2. Calculate the executability of each module as fuzzy-and of the preconditions
3. Multiply activation and executability. If the highest product lies above the behavior selection threshold, execute the corresponding behavior, reset the threshold to its original value and go to 1.
4. Otherwise reduce the threshold by a small amount and go to 1.

The multiplication of activation and executability results in a tradeoff between the utility of a behavior and the probability of successful execution. Behaviors with a high utility may be executed even if their executability, i.e. the likelihood of being successful, is small. In contrast to the expectation values of behavior's effects, which are a measurement for the competence of the agent, executability indicates the opportunity within the situation to perform a behavior.

2 Agents of magmaFreiburg

In this section we describe how extended behavior networks are integrated into our soccer agents.

2.1 Perception

Each time an agent receives a perception from the server, the information is entered into a *local map* containing the distances and directions of visible objects. After self-localization, the global position and direction of the agent and all visible objects are inserted into a *global map*. Moveable objects like other players and the ball are removed from the map after three seconds or if they are not seen at the expected position. From the information in the global map, functional objects are calculated using indexical-functional aspects of the situation [1]. This reduces the number of competence modules needed. E.g. there is only one rule 'attack opponent at ball' instead of 'attack opponent1', 'attack opponent2', ... needed when addressing objects absolutely.

Finally, the agent calculates the truth values for the behavior network's perception-propositions (e.g. 'near goal' is true 5 m in front of the goal and false 30 m away with linear interpolation). Information on visible objects is taken from the local map, whose data is likely to be more accurate than that of the global map. Invisible objects are accounted for by the global map.

2.2 Action

Action selection is done in three steps: First, the agent checks for any reflex to be executed. Reflexes are actions that are always executed once their stimulus is present. For example, moving an agent to its home position after a goal has been scored is modeled as a reflex. Second, behaviors of a competence module may consist of several (up to three) primitive actions. If such an action chain has not been executed to completion, the next action in the chain is sent to the server. Action chains reduce the cost of calculation, but decrease reactivity. Third, if no reflex and no other action has been executed, action selection is performed by the behavior network (see Fig. 1) as described in section 1.3.

Technically, behavior networks run in a dynamic link library, separated from the domain dependent C++ code. They are specified by ASCII-files containing the goals and behavior rules for the goalie, defensive and offensive agents. Perception and action methods are implemented as callback methods passed from the agent class to the behavior network.

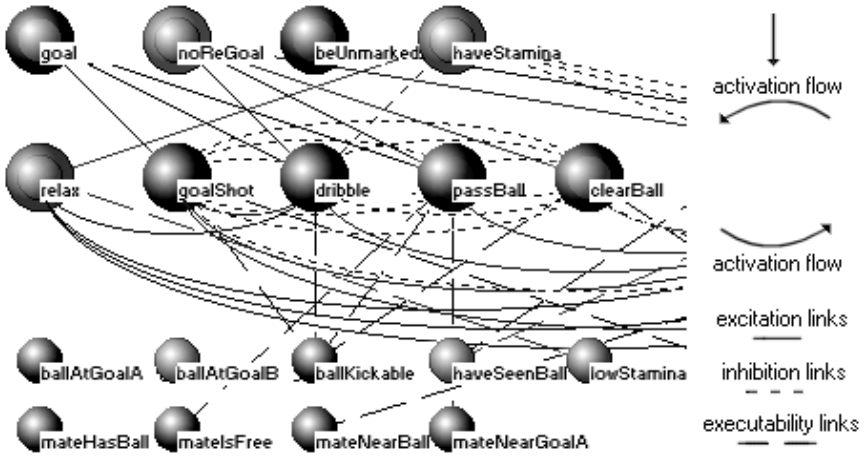

Fig. 1. Part of a behavior network used by a soccer-agent. The goals of the agent are at the top level, in the middle the competence modules and at the bottom level the situation propositions (perceptions). (The complete network contains nine competences).

Acknowledgements

The work reported here has been funded by the German Research Association (DFG, Graduiertenkolleg Menschliche und maschinelle Intelligenz). I would like to thank Gerhard Strube and Bernhard Nebel for important comments and suggestions during the preparation of this research.

References

1. Agre, Ph., and Chapman, D. (1987). Pengi: An Implementation of a Theory of Activity. In: *Proceedings of the Sixth National Conference on Artificial Intelligence, AAAI-87*, Morgan Kaufmann, Los Altos.
2. Dorer, K. (1999). Behavior Networks for Continous Domains using Situation-Dependent Motivations. To appear in: *Proceedings of the* 16^{th} *International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, Stockholm.
3. Goetz, Ph. (1997). *Attractors in Recurrent Behavior Networks.* PhD thesis, State University of New York, Buffalo.
4. Maes, P. (1989). The Dynamics of Action Selection. In *Proceedings of the International Joint Conference on Artificial Intelligence-'89* , Morgan Kaufmann, Detroit.
5. Maes, P. (1990). Situated Agents Can Have Goals. In *Journal for Robotics and Autonomous Systems*, Vol. 6, No 1, pages 49-70, North-Holland.
6. Maes, P. (1992). Learning Behavior Networks from Experience. In Varela, F. and Bourgine, P. *Proceedings of the First European Conference on Artificial Life*, pages 48-57, MIT-Press, Paris.
7. Noda, I. (1995). Soccer server: a simulator of robocup. In *Proceedings of AI symposium '95*, pages 29-34, Japanese Society for Artificial Intelligence.