Concurrent Behavior Selection in Extended Behavior Networks

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. This paper describes a distributed mechanism for concurrent behavior selection in agents using extended behavior networks. Resource nodes are introduced into the networks to coordinate conflicting behaviors and manage limited resources of the agent. Concurrent behavior selection improves the agent's reactivity and allows pursuing multiple goals at once. Empirical data on concurrent behavior selection in the RoboCup domain shows significantly better results compared to serial behavior selection.

1 Extended Behavior Networks

Extended behavior networks [1] are a means to carry out reactive and goal-directed behavior selection. They extend original behavior networks [3] through the explicit representation of goals with dynamic, i.e. situation-dependent, utility function and through the introduction of continuous state-propositions to exploit additional information in continuous domains. A shortcomming of present behavior networks is that behavior selection results in a single behavior to be executed at any time. Concurrent execution of behaviors that is not supported. This paper describes the extensions neccessary to overcome this limitation.

In order to decide if behaviors interfer, resources are introduced into the model. Two behaviors do not interfer if they do not use any resource in common or if the resources they both use are sufficiently available. Two issues can be directly concluded from this. First, resource usage between the competence modules that represent the agent's behaviors must be coordinated. Therefore resource nodes are introduced.

Definition 1. A *resource node* is a tupel (res, g, θ_{Res}) with

- res the resource represented by this node,
- g the amount of resources bound by a currently active competence module,
- θ_{Res} the resource-specific *activation threshold*.

Second, a competence module needs to know which and how many resources it will use if it is executed. The amount of resources needed may be situation-dependent. The stamina needed, for example, by the behavior 'runToBall' depends on the distance to the ball. This information is added to a competence module.

Definition 2. A *competence module* is a tupel (*Pre*, *b*, *Post*, *Res*, *a*) with

- *Pre* the *preconditions* of the module,
- *b* the *behavior* that is executed once the module is selected for execution,
- *Post* the *effects* of the behavior with corresponding effect probabilities,
- *Res* the *resources* used by *b*, with $\tau_U(k, res, s)$ the situation-dependent amount of resources expected to be used,
- *a* the *activation* of the module.

An extended behavior network able to deal with resources is then defined by:

Definition 3. An *extended behavior network* EBN is a tupel $(\mathcal{G}, \mathcal{M}, \mathcal{U}, \Pi)$, with $\mathcal{G}$ the goals of the agent, $\mathcal{M}$ a set of competence modules, $\mathcal{U}$ a set of resource nodes and Π the *parameters* used to control activation spreading and behavior selection.

For a more detailed description on goals and parameters see [1]. To be able to coordinate concurrent behavior selection and to avoid resource conflicts, a competence module is connected to the resource nodes of resources its behavior will use. These connections are used to exchange information on available resources and on resource usage by a behavior. The process of concurrent behavior selection that is based on this information is described in the next section.

2 Concurrent Behavior Selection

Concurrent behavior selection is done in a cycle containing the following steps:

1. Calculate the execution-value h of each module, which is a monotonically increasing function of the activation and executability of a module (see [1] for details).
2. For each resource *res* used by competence module k, starting with the previously unavailable resource
   (a) Check if h supercedes the activation threshold θ_{Res_i} of the corresponding resource node.
   (b) Check if enough resource units are available, i.e. check if $\tau_U \leq \tau_R(res, s)$. If so, bind the resource-units, i.e. increase the number of used resource-units of the resource node by the number of expected units the behavior will use.
3. If all tests in 2 succeeded
   (a) Execute the corresponding behavior.
   (b) Reset the activation thresholds of all resources used.
4. Release all bound resource-units, i.e. reduce the number of bound resource units of the resource node by the number of previously bound units.
5. Repeat from 1.

The activation threshold θ_{Res_i} ensures that the competence module with highest activation value that is executable will be executed. θ_{Res_i} linearly decreases over time so that eventually a module supercedes the threshold and may be executed. If the execution of the module with highest activation value is prevented by a missing resource, another module with less activation not using the missing resource may be executed. Modules that have disjunct resources may be executed concurrently.

Besides allowing concurrent behavior selection, this algorithm overcomes another limitation of behavior networks. Behavior selection has previously been done by selecting the most active executable competence module for execution. Unfortunately, this information can not be calculated locally in a competence module. Therefore, the process of action selection could not be calculated distributedly in each competence module and has not been parallelizable. By introducing resource nodes, a competence module is now able to perform action selection locally.

	serial	parallel	p ($n = 30$)
Mean no of goals	2.4	4.3	< 0.001

Table 1. Comparison of serial und parallel behavior selection in the RoboCup domain.

3 Empirical Results

Since version 5 of the RoboCup-soccerserver, commands can be executed concurrently, if they do not use the same resources. A `say`-command, for example, can be executed concurrently with a `kick`-command or a `turn_neck`-command. The concurrent execution of such actions should improve the speed and reactivity of an agent.

This has been examined in a series of 30 soccer-games. Two identical teams played against each other. The only difference was that one team used concurrent behavior selection, while the other team used serial action selection, i.e., only the first action of a cycle was executed. The concurrent team's agents were able to execute communication, head turning and running and kicking actions concurrently. Since the number of cycles an agent can communicate is restricted to 4% of all cycles and because separate turning of the head relative to the body was only executed in about 8% of all cycles, concurrent behavior selection effectively only took place in 2% of the cycles. Despite this, the team using concurrent behavior selection scored significantly[1] more goals than the team using serial behavior selection (see table 1).

[1] two samples t-test with $\alpha = 0.01$.

4 Future Work

Besides the improvements of concurrent behavior selection we are working on several other improvements for RoboCup2000 in Melbourne.

We are currently working on an improved memory model for extended behavior networks. Besides the memorization of perceptions in a global map that is already done by agents of magmaFreiburg99 [2], extended behavior networks should be able to have access to previous decisions and other internal information. Information that is difficult to perceive, e.g., being in offside, should be accessible to the agent for some time after the agent came to the conclusion that it was offside. We hope that a mechanism like this will produce a more continuous behavior in specific situations. Other mechanisms to produce a more continuous behavior, like the inertia of activation, which is already part of our model, are not specific enough to improve the overall success of the agents.

Other (domain dependent) issues we are working on are the marking of opponent players, which was not implemented in our last team, improvement of ball capabilities to reduce the number of ball losses and the ability to communicate team strategies. We are also looking into agent debugging, which is becoming increasingly important once the basics of an agent's capabilities are working. An interesting approach has been demonstrated by [4].

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. Participation in RoboCup2000 in Melbourne is sponsored by Microsoft Research.

References

1. Dorer, K. (1999). Behavior Networks for Continous Domains using Situation-Dependent Motivations. In: *Proceedings of the* 16^{th} *International Joint Conference on Artificial Intelligence* , pages 1233-1238, Morgan Kaufmann, San Francisco.
2. Dorer, K. (1999). Extended behavior networks for the MagmaFreiburg team. In S. Coradeschi, T. Balch, G. Kraetzschmar, und P. Stone (Eds.), *Team Descriptions Simulation League*, Seiten 79–83. Linköping University Electronic Press, Stockholm, 1999.
3. Maes, P. (1989). The Dynamics of Action Selection. In *Proceedings of the International Joint Conference on Artificial Intelligence-'89* , Morgan Kaufmann, Detroit.
4. Stone, P., Riley, P., and Veloso, M. (to appear). Layered Disclosure: Why is the agent doing what it's doing?. Submitted to *Seventeenth National Conference on Artificial Intelligence (AAAI-2000).*