2003

Goal-Converging Behavior Networks and Self-Solving Planning Domains

Bernhard Nebel
Universitat fur Informatik

Yuliya Lierler
University of Nebraska at Omaha, ylierler@unomaha.edu

Goal-Converging Behavior Networks and Self-Solving Planning Domains

Bernhard Nebel
Institut für Informatik
Albert-Ludwigs-Universität Freiburg
nebel@informatik.uni-freiburg.de

Yuliya Babovich
University of Texas
yuliya@cs.utexas.edu

Abstract

Agents operating in the real world have to deal with a constantly changing and only partially predictable environment and are nevertheless expected to choose reasonable actions quickly. One way to address this problem is to use behavior networks as proposed by Maes, which support real-time decision making. Robotic soccer appears to be one domain where behavior networks have been proven to be particularly successful. In this paper, we analyze the reason for the success by identifying conditions that make behavior networks *goal converging*, i.e., allow them to reach the goals regardless of which particular action selection scheme is used. In terms of STRIPS domains one could talk of *self-solving planning domains*. We finally show that the behavior networks used for different robotic soccer teams have this property.

1 Introduction

Agents operating in the real world have to deal with a constantly changing and only partially predictable environment; and the expectation is that the agents can figure out the best suitable actions in real-time. The *behavior network* approach [Maes, 1990] addresses this problem through activation spreading inside a network of competence modules. This approach is intended to address, as Maes [1990] states, the problems of "brittleness, inflexibility, and slow response" of classical planning approaches on one hand, and the problem of "the lack of explicit goals" in reactive approaches on the other hand. It proved to be useful and became popular during the last decade. For instance, it has been used in the implementation of an intelligent e-mail agent [Zhang *et al.*, 1998] and as the underlying mechanism for generating behavior of autonomous characters in interactive story systems [Rhodes, 1996]. Most notably, the approach has been employed in the simulated robotic soccer team *magmaFreiburg* [Dorer, 2000a] and in the real robotic soccer (F2000 league) team *CS Freiburg* [Weigel *et al.*, 2001; 2002]. In both cases, the teams were highly successful. The simulation team *magmaFreiburg* was runner-up in 1999 and *CS Freiburg* won the *RoboCup* world championship in 2000 and 2001. Although there is a wide range of components, both hard- and software, that contribute to such a success, the behavior networks, as reported by Weigel *et al.* [2001; 2002] and Dorer [2000a], played a significant role.

It must be said, however, that the particular action selection mechanism employed in the robotic soccer teams differs significantly from Maes' [1990] original proposal. The so-called *extended behavior networks* [Dorer, 1999], which are used in the robotic soccer domain, can deal with continuous propositions, use a technique called *goal-tracking* in order to address some of Tyrrell's [1994] criticisms concerning Maes' [1990] proposal, and employ a goal management mechanism that allows for changing goals. In fact, with all the extensions, the behavior networks have now the flavor of decision-theoretic planning, without implementing this framework, though. Furthermore, as shown in a number of experiments [Dorer, 1999], these changes lead to a significantly higher number of scored goals.

Although Maes' behavior networks and variations have been analyzed from several perspectives, there are nevertheless many issues that have not been resolved. For example, it is not clear under which conditions we can be sure that a behavior network converges to its goal, i.e., generates an action sequence that eventually satisfies the goal. Dorer [2000b] describes some experiments where he used the original behavior networks by Maes [1990] in order to solve *blocks-world planning* problems. As it turns out, for some five-block problems, the behavior network goes into an infinite loop and does not come up with a solution, regardless of the parameter setting. Clearly, such a performance would be unacceptable in a soccer context. Just imagine a soccer player who dribbles the ball in an endless circle. However, this does not happen in this domain. One could explain this difference by the fact that the blocks world is an artificial domain with a puzzle-like character while soccer has a real-world character much more suited for behavior networks. However, it would be, of course, more interesting to find some formal conditions that explain why behavior networks work so well for robotic soccer.

More generally, we are interested to find a condition that guarantees that the behavior network will generate a successful sequence of actions provided there exists one and no exogenous events intervene. Furthermore, we want this guarantee regardless of which particular action selection scheme and

parameter setting is employed. Behavior networks with this property will be called **goal converging**. If we view the behavior network as a STRIPS planning domain specification, then the corresponding domain specification could be termed *self-solving*, since all sequences of executable actions lead to the goal.[1]

If a behavior network is goal converging, then we know that it will always act goal-oriented and parameter tuning is only necessary to generate better, shorter action sequences. Of course, it is also clear that goal convergence will require severe restrictions on the structure of behavior networks. However, as we show in this paper, there exists a non-trivial restriction on the topology of the behavior network that guarantees that the network is goal converging. In addition, all the networks that have been designed for robotic soccer are of this type (or are very close to this form), which explains to some degree why the approach works so well for robotic soccer.

The rest of the paper is structured as follows. In the next section we sketch the behavior network approach. In Section 3, we identify two conditions for a behavior network being goal converging. Based on that, we analyze in Section 4 the networks that have been used in the Freiburg RoboCup teams and show that they satisfy one of the conditions identified. Finally, in Section 5, we conclude and give an outlook.

2 Behavior Networks

In the following, we describe the behavior network formalism. Since we do not need the full details for our purposes, the description will be sketchy at some points.

2.1 Specifying Behavior Networks

Let $\mathcal{P}$ be a set of propositional atoms. A **state** is a truth assignment to all atoms in $\mathcal{P}$ (often also represented as the set of true atoms). For extended behavior networks [Dorer, 1999], the state is an assignment of fuzzy values. **Behavior networks** are tuples $(\mathcal{P}, \mathcal{G}, \mathcal{M}, \Pi)$, where

- $\mathcal{G} \subseteq \mathcal{P}$ is the **goal specification**;
- $\mathcal{M}$ is a finite set of **competence modules** or **actions**, where $m \in \mathcal{M}$ is a tuple $\langle \mathit{pre}, \mathit{eff}^+, \mathit{eff}^-, \mathit{beh} \rangle$ with $\mathit{pre} \subseteq \mathcal{P}$ denoting the **preconditions**,[2] $\mathit{eff}^+, \mathit{eff}^- \subseteq \mathcal{P}$ denoting the positive and negative effects, respectively, with $\mathit{eff}^+ \cap \mathit{eff}^- = \emptyset$ and *beh* being the name of an executable **behavior**, which is started once the module is selected for execution. If we want to refer to one of the components of a competence module m we use the notation $\mathit{pre}(m), \mathit{eff}^+(m)$, etc.
- Π is a set of **global parameters** used to control the action selection process, among them the threshold for the activation θ. There are more parameters, but we do not need them for our purposes and ignore them for this reason.

Depending on the type of behavior networks, some variations are possible. For example, in Dorer's [1999] extended behavior networks, the goals can have an importance measure and an additional relevance condition. Further, effects have an expectation value describing how likely it is that the effect proposition becomes true after executing the competence module. These details will not be important for us, though.

2.2 Activation Spreading

Competence modules are connected in a network so that they can send and receive activation energy. A *positive effect link* connects a positive effect p of a competence module to the precondition p of another competence module. A *negative effect link* connects a negative effect p of one competence module to the precondition p of another competence module.[3] An example of a small behavior network is given in Figure 1.

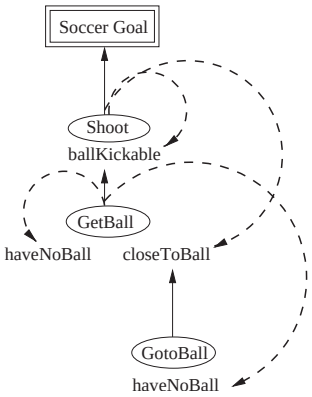

Figure 1: Example of a behavior network: Solid arrows denote positive effect links and dashed arrows denote negative effect links.

In this example, the competence module *GotoBall* has the precondition *haveNoBall* and the effect *closeToBall* enabling the competence module *GetBall*. This, in turn, has the negative effect of deleting *haveNoBall* and the positive effect of making *ballKickable* true. The latter enables the *Shoot* module, which then (hopefully) leads to scoring a goal, the ultimate goal of this behavior network.

Unsatisfied goals send some activation energy to competence modules that could make the goals true and, in turn, each activated module sends some of its activation through its unsatisfied preconditions to modules which can make the precondition true. In the original version of behavior networks, there is also a "forward spreading" of activation energy. This means that activation energy flows from propositions true in a

[1] This condition corresponds to what is called the *all-policies-proper* condition in the MDP community. However, in this context one usually assumes the condition and does not try to identify criteria which guarantee the condition.

[2] Note that we allow only for positive goals and preconditions. This, however, does not restrict the expressivity since (for STRIPS-like planning) this is equivalent to formalisms with negative preconditions and goals under various formal notions of expressive equivalence [Bäckström, 1995; Nebel, 2000].

[3] Although negative self-links are usually not considered, we will draw them in depictions of behavior networks in order to describe the actions completely.

situation towards competence modules that have these propositions as preconditions, and from executable competence modules to competence modules which have unsatisfied preconditions identical to the effects of the executable modules. However, this forward spreading of activation does not seem to increase the quality of the action selection [Dorer, 1999; Goetz and Walters, 1997] and for this reason this kind of activation is not present in Dorer's [1999] extended behavior networks. While positive effect links are used for spreading activation, negative links are used to inhibit the activation of other modules. Modules that have the negative effect $p \in \mathit{eff}^-$ are inhibited by modules that have p as a satisfied precondition.

2.3 Action Selection

Action selection is done in a cycle containing four steps [Maes, 1990; Dorer, 1999]:

1. The current activation of each module is calculated using the methods described above, i.e., each modules receives some activation and inhibition from modules connected to it.
2. Activation and executability of a module are combined by a non-decreasing function into the utility of a module, whereby non-executable competence modules always get the value zero.
3. The module with the highest utility value is chosen,[4] provided it passes a certain threshold θ (one of the global parameters). The action associated with the competence module is then executed.
4. If none of the modules reached the activation threshold, the threshold is reduced by a certain percentage (another global parameter) and the cycle is started again with the currently computed activation values for each module.

Since we usually want an agent to execute a sequence of actions leading to the goal, the above cycle will be called infinitely or until the agent has reached the goal.

From the description above it follows that there are only a few things one can be sure of when using a behavior network for action selection. First of all, only executable actions are chosen. Second, if an action selection scheme is employed that does not use forward activation spreading, for instance Dorer's [1999] scheme, then it follows that if an action is chosen, it "contributes" to one of the goals, since the competence module can receive activation only from the goal through a chain of unsatisfied preconditions.

2.4 Ideal Abstract Behavior Networks

If we want to guarantee properties of a network under different action selection schemes and parameter settings, we have to make a number of simplifying assumptions. We will assume that the state is always correctly observable (with Boolean state variables), that the competence modules describe all relevant effects correctly, that the execution of the behavior of a competence module is always successful, and that no exogenous event will intervene. Based on these assumptions, we define an abstract version of behavior networks, which from a formal point of view are identical to STRIPS domain descriptions.

An **ideal, abstract behavior network** is a tuple $\mathbf{B} = (\mathcal{P}, \mathcal{G}, \mathcal{M})$, where $\mathcal{P}, \mathcal{G}$ and $\mathcal{M}$ are defined as in Section 2.1. In the state $S \subseteq \mathcal{P}$, the network can choose any competence module m for execution such that the preconditions $\mathit{pre}(m)$ are satisfied in S, i.e., $\mathit{pre}(m) \subseteq S$, and not all positive effects are satisfied, i.e., $\mathit{eff}^+(m) - S \neq \emptyset$. When m is executed in state S, the resulting state $\mathit{Result}(S, m)$ is given by

$$\mathit{Result}(S, m) = S - \mathit{eff}^-(m) \cup \mathit{eff}^+(m).$$

We say that the network $\mathbf{B}$ can **generate** a (finite or infinite) sequence of actions $m_1, m_2, \ldots, m_i, \ldots$ in a state S_1 if

$$S_{i+1} = \mathit{Result}(S_i, m_i).$$

We say $\mathbf{B}$ can **reach the goals** $\mathcal{G}$ from a state S if it can generate a finite sequence of actions in S such that the last state S_n satisfies the goals, i.e., $S_n \supseteq \mathcal{G}$.

3 Goal-Converging Behavior Networks

If we want to guarantee that a behavior network is successful regardless of the action selection scheme and parameter setting,[5] we have to consider all action sequences the network can generate. Although this appears to be a fairly strong requirement, there are indeed realistic networks for which we can show that they are always successful—if the goal is reachable at all.

3.1 Terminating and Dead-End Free Networks

We call a behavior network **terminating** if for all states and under all possibilities to choose actions, it is impossible to generate infinite action sequences—provided the goal was reachable initially.[6] Figure 2 gives a simple example of a non-terminating network.

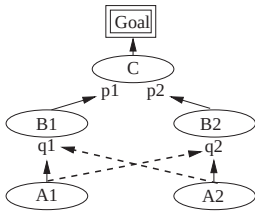

Figure 2: A *non-terminating* behavior network

Provided that $p1, p2, q1, q2$ and the *Goal* are false initially, then it is possible that the sequence $A1, A2, A1, A2, \ldots$ is chosen. Hence, the network is not terminating. Note that there is a successful sequence consisting of

[4] Ties are broken randomly.

[5] The only restriction is that we never consider actions such that all their positive effects are already satisfied (see Section 2.4).

[6] If the goal is unreachable, we do not care about the behavior of the network.

$A1, B1, A2, B2, C$. However, the action selection mechanism might not necessarily find it. An example for a terminating network is the one in Figure 1, as is easy to verify.

We say that a network is in a **blocked state** when no action is executable and the goal is not satisfied. Such a blocked state may occur because there was no way to reach the goal in the first place. However, it may be possible that the goal was reachable in the beginning. We call a network **dead-end free** if it never leads to a blocked state when it is possible to reach the goal. Consider, for example, the network in Figure 3. This network contains a dead end. Provided one starts with $p1, p2, q2$ and *Goal* as false and $q1$ as true, the execution of $A2, B2$ leads to a blocked state. However, obviously, the sequence $B1, A2, B2, C$ would have led to the goal. In other words, this network is not dead-end free. An example of a dead-end free network is again the one in Figure 1. Although in this network one can make propositions false, this can only happen in the course of satisfying the goal and it will never prohibit reaching the goal.

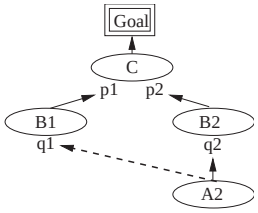

Figure 3: A behavior network with a *dead end*

Finally, we call a behavior network **goal converging** when it will generate a finite action sequence leading to the goal regardless of the action selection scheme and parameter setting, provided the goal is reachable at all. When viewing the behavior networks as specifications of STRIPS planning problems, we would talk of **self-solving** planning domains, because regardless of which order we would choose for the executable actions, one would always reach the goal—provided the goal was initially reachable at all.

Proposition 1 *A behavior network is* goal converging *if and only if it is* dead-end free *and* terminating.

Proof: The "only if" direction is obvious since networks with dead ends and networks which are non-terminating cannot be goal converging. There are possible states and action selections such that either a loop or a dead end, respectively, are chosen although there is the possibility of reaching the goal. For the "if" direction observe that a non-goal-converging network must either produce an infinite sequence or end up in a dead end although there is a action sequence leading to a goal state. ■

3.2 Monotone Networks

One particularly simple type of goal-converging networks are networks with only positive effects, which we will call **monotone networks**. Since a propositional atom can never be made false in a monotone network, one can reach any desired goal after any initial sequence of actions, provided the goal was initially reachable. This implies that it is impossible to run into a dead end. Since each action can be executed at most once, there is additionally an upper bound to the length of any action sequence generated by a monotone behavior network, implying that the network is also terminating.

Proposition 2 *Monotone behavior networks are goal converging.*

Monotone behavior networks are hardly interesting, because they almost never appear in practice.For our purposes, they are equivalent to STRIPS planning problems that have only positive preconditions and effects. While such planning problems appear to be trivial, it is well known that generating a shortest plan is still an NP-hard problem [Bylander, 1994]. Furthermore, such planning problems have become popular as the basis for computing heuristic estimates in action planning [Hoffmann and Nebel, 2001; Bonet and Geffner, 2001]. For our purposes, however, the restriction to purely positive effects is not possible.For instance, in our example network in Figure 1, the action *GetBall* destroys the *haveNoBall* condition.

3.3 Acyclic Networks with Restricted Negative Links

In order to specify a more interesting class of goal-convergent networks, let us view these networks from a slightly different angle. Let us consider directed graphs with two kinds of nodes, **action nodes** and **fact nodes** and two kinds of directed edges, **positive** and **negative** ones, such that

- there is a positive (precondition) edge from fact node p to action node a if p is a precondition of action a;
- there is a positive (effect) edge from action node a to fact node p if p is a *positive* effect of a;
- there is a negative (effect) edge from action node a to fact node p if p is a *negative* effect of a.

The resulting graph is called **action-fact graph**.[7] The **normalized action-fact graph** is the directed graph where the direction of the negative edges has been reversed. The interesting point is that acyclicity of the normalized action-fact graph implies that the behavior network is terminating.

Theorem 3 *A behavior network which corresponds to an acyclic normalized action-fact graph is terminating.*

Proof: In order to show that a behavior network satisfying the condition of the theorem is terminating, we assign as a first step values to the atoms in the action-fact graph. For each atom p the value of p should be 1 plus the sum of values of the fact nodes that are incident via a negative edge to an action having p as a positive effect. Since the normalized action-fact graph is acyclic, this value assignment is well-defined.[8]

With this value assignment to atoms, each action application will strictly increase the overall value of the state (as the

[7] Such graphs correspond to what has been called *bi-level planning graph* [Long and Fox, 1999] or *connectivity graph* [Hoffmann and Nebel, 2001] in the planning literature.

[8] In fact, as is obvious from this argument, it suffices when the sub-graph consisting of effect edges only is acyclic.

sum over the values of all true propositions), because an action is only executed when one of its positive effects is not true. This implies, however, that it is impossible to generate infinite action sequences. ■

While it was easy to find a condition for termination, it appears to be much more difficult to find a criterion that guarantees that the network is dead-end free. Let us consider even further restricted action-fact graphs. If the sub-graph formed from the positive links is acyclic and if for all negative edges from action a to fact p there exists a positive path from p to a, then we call the graph **acyclic, negative-feedback action-fact graphs**. This is obviously a special-case of an *acyclic normalized action-fact graph*. However, it is still not a criterion for guaranteeing the absence of dead ends. In fact, planning is still non-trivial as the plan existence problem is still NP-hard.

3.4 Modular Action-Fact Graphs

One way to guarantee that there are no dead ends is to make sure that it is always possible to make falsifiable propositions true without affecting other propositions, which has to be guaranteed independently of the initial state[Hoffmann, 2002]. While this condition is often true in classical planning tasks, it seems very unlikely that we can guarantee this in our case. Hoffmann [2002] gives a number of other sufficient conditions, but none appears to be applicable here. For this reason, we will look into an alternative condition. We will try to make sure that any proposition that can be falsified needs never to be used again after it has been falsified. For example, this condition is satisfied in Figure 1. One way to guarantee this is to require the following *modularity* condition. For all atoms q that can be falsified by an action a in an acyclic, negative-feedback action-fact graph, each positive path from q to a goal atom must go through an action a' such that $\mathit{eff}^+(a) \supseteq \mathit{eff}^+(a') \neq \emptyset$. This condition is, for example, satisfied by the action-fact graph in Figure 4 and the action-fact graph derivable from the network in Figure 1. We call acyclic, negative-feedback action-fact graphs satisfying this condition **modular action-fact graphs**.

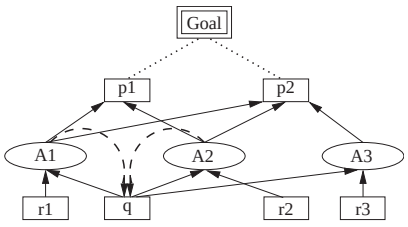

Figure 4: An action-fact graph satisfying the *modularity* condition

Theorem 4 *Modular action-fact graphs are goal converging.*

Proof: Termination follows from Theorem 3. The proof that the action-fact graphs are dead-end free is by induction on the number of negative links. For $k = 0$ negative links, the claim follows from Proposition 2. Assume now that the claim is true for modular action-fact graphs with k or fewer negative links. Consider a graph with $k+1$ negative links. Now choose one action node a that is the source of a negative link and which has no positive path to any other action node with such a property. Because of the acyclicity of the graph formed from positive links, such a node must exist. Assume that q is amongst the negative effects of a and that the positive effects are $p_1, \ldots, p_k$. If we remove the negative link from a to q, we can apply the induction hypothesis for k negative links and know that the graph is dead-end free.

Assume now for contradiction that the original network is not dead-end free. This must be connected with the possibility of falsifying q by a. However, once all the positive effects of a have been made true by executing a, the truth value of q is not of any concern since all positive paths from q to a goal go through a and actions with a subset of $\mathit{eff}^+(a)$ as their positive effects. Hence, the negative link from a to q cannot create a dead end, which completes the induction step. ■

4 RoboCup Behavior Network

As mentioned in the Introduction, the analysis of behavior networks was motivated by the observation that the behavior networks of the *magmaFreiburg* and *CS Freiburg* robotic soccer players work so robustly. When one now analyzes the networks with the tools developed in this paper, it turns out that they indeed satisfy the condition of being *modular*—modulo some qualifications. Before we talk about qualifications, we should, however, have a look at some real behavior networks. In Figure 5 the main part of the *CS Freiburg* [Müller, 2000] behavior network is displayed as an action-fact graph. Obviously, the few negative links satisfy the *modularity* condition. However, one may wonder, why there are no negative links from the actions having *HaveBall* as a precondition to *HaveBall*? Although these negative links should have been there in order to describe the action effects correctly, their absence is not problematic, since we assumed that all actions are successful—and the positive effect of all the actions is the ultimate goal. In any case, when adding the negative effects, we still would have a *modular* action-fact graph.[9]

A similar comment applies to the missing positive links back to *NegHaveBall*. Again, it is not interesting because we achieve the goal anyway. Furthermore, we can ignore these positive links without losing anything, i.e., they never help us to achieve the goal.

Often it is necessary to take more than one goal into account. The extended behavior network may contain multiple goals which can be selected based on the current situation. So, for example, a *CS Freiburg* player either tries to score a goal (if it fills the role of an *active player*) or it has the overall goal to *cooperate*. In the latter case, we would have to consider a different network, which also satisfies the structural condition of being *modular*, though. In the case of the *magmaFreiburg* players, things are even more complicated because it is possible to pursue more than one goal at once. If we break the networks down to one goal at a time, however, the resulting networks are again modular.

[9] Indeed, the *magmaFreiburg* networks contain these negative effects.

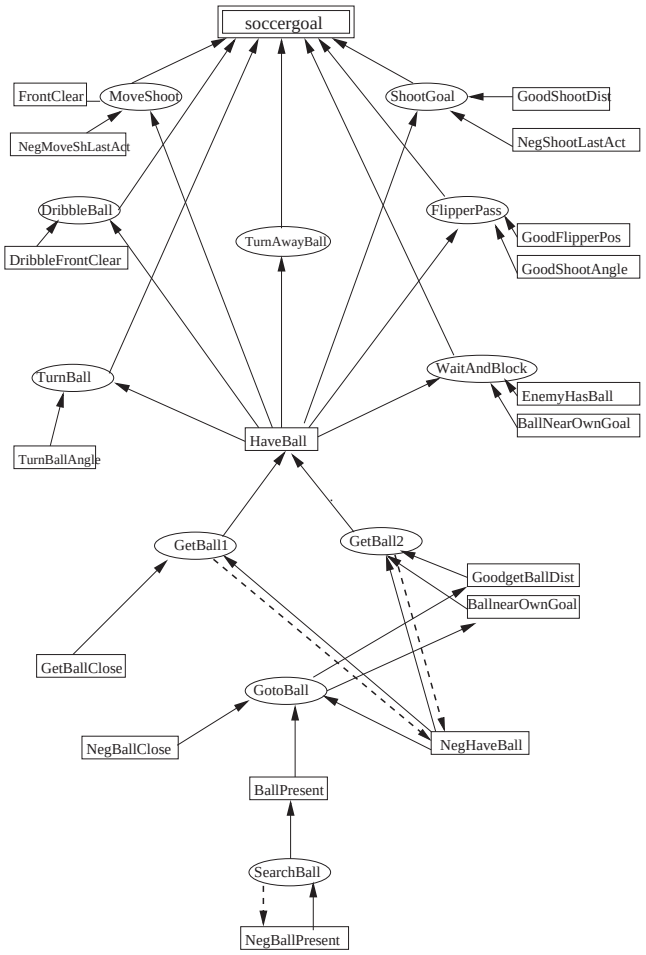

Figure 5: Part of the Action-Fact Graph of the *CS Freiburg* behavior network [Müller, 2000]

Finally, it should be noted that there are levels in the decision making that influence the behavior networks, e.g., the role assignment and placement of players on the field [Weigel *et al.*, 2001], which are, however, not taken into account when analyzing the network.

Summarizing, if we assume that no exogenous actions intervene and if there occurs no change in the goals (in particular there is no influence from the strategic component), then all the behavior networks of the *CS Freiburg* [Müller, 2000] and the *magmaFreiburg* [Dorer, 2000b] players satisfy the modularity condition and are therefore goal converging, which goes somewhere in explaining why they have been successful. At least, when players are alone on the field, they will eventually score. Although this is a rather weak guarantee, it is much better than the statement that the player might score a goal only when the parameters of the network are well adjusted.

Of course, all this seems to imply that the domain as modelled in the described RoboCup teams has a quite simple structure. However, thinking a while about the problem, one will come to the conlusion that even in the face of more complex modelling and decision making by, e.g., integrating opponent modelling and adversary planning, we nevertheless would like to guarantee the conditions mentioned above. However, it may be the case that it is not possible to verify the conditions using simple syntactic tests any longer.

5 Conclusions and Outlook

We have identified a structural property of behavior networks, called *modularity*, that guarantees that the networks will reach their goals in a static environment under all circumstances—if the goals are reachable at all. Interestingly, there exists a significant application of behavior networks where this restriction is met, namely, the networks of the Freiburg simulation and real robot (F2000) soccer players.

Having shown that a network has this property means that we never have to fear that the network leads to infinite action sequences or blocked states. In addition, it means that tuning network parameters [Maes, 1992] will not modify the principal property of reaching the goal, but only the efficiency.

In the future, we will pursue three directions of research. First of all, there is the question whether there exist other relevant restrictions on network structures that lead to goal convergence. Second, in most cases, it is enough if the network is goal converging for a subset of all possible states. Now, the interesting question is in how far this would result in a more liberal condition for goal convergence. Third, we will analyze the feasibility of testing the property of goal convergence on a semantic level. In this context, it will probably be helpful to take the syntactic restrictions identified in this paper into account, because it is probably prohibitive to inspect the entire state space.

Acknowledgments

We acknowledge the comments and suggestions by Jörg Hoffmann, Wolfram Burgard, and Malte Helmert on an earlier version of this paper.

The first author has been partially supported by the University of Newcastle, NSW, Australia, during his sabbatical. The second author has been partially supported by the DAAD as part of the International Quality Network on Spatial Cognition during a visit at the University of Freiburg.

References

[Bäckström, 1995] Christer Bäckström. Expressive equivalence of planning formalisms. *Artificial Intelligence*, 76(1–2):17–34, 1995.

[Bonet and Geffner, 2001] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33, 2001.

[Bylander, 1994] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

[Dorer, 1999] Klaus Dorer. Behavior networks for continuous domains using situation-dependent motivations. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1233–1238, Stockholm, Sweden, August 1999. Morgan Kaufmann.

[Dorer, 2000a] Klaus Dorer. The magmaFreiburg soccer team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, pages 600–603. Springer-Verlag, Berlin, Heidelberg, New York, 2000.

[Dorer, 2000b] Klaus Dorer. *Motivation, Handlungskontrolle und Zielmanagement in autonomen Agenten*. PhD thesis, Albert-Ludwigs-Universität, Freiburg, Germany, 2000. Published on FreiDok server under http://www.freidok.uni-freiburg.de/volltexte/57.

[Goetz and Walters, 1997] Philip Goetz and Deborah Walters. The dynamics of recurrent behavior networks. *Adaptive Behavior*, 6(2):247–283, 1997.

[Hoffmann and Nebel, 2001] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[Hoffmann, 2002] Jörg Hoffmann. Local search topology in planning benchmarks: A theoretical analysis,. In *Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems (AIPS-02)*. AAAI Press, Menlo Park, 2002.

[Long and Fox, 1999] Derek Long and Maria Fox. Efficient implementation of the plan graph in STAN. *Journal of Artificial Intelligence Research*, 10:87–115, 1999.

[Maes, 1990] Pattie Maes. Situated agents can have goals. In Pattie Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, pages 49–70. MIT Press, Cambridge, MA, 1990.

[Maes, 1992] Pattie Maes. Learning behavior networks from experience. In *Proceedings of the First European Conference on Artificial Life*, pages 48–57, 1992.

[Müller, 2000] Klaus Müller. Roboterfußball: Multiagentensystem CS Freiburg. Diplomarbeit, Albert-Ludwigs-Universität, Freiburg, Germany, 2000.

[Nebel, 2000] Bernhard Nebel. On the compilability and expressive power of propositional planning formalisms. *Journal of Artificial Intelligence Research*, 12:271–315, 2000.

[Rhodes, 1996] Bradley Rhodes. PHISH-nets: Planning heuristically in situated hybrid networks. Master's thesis, MIT, Cambridge, MA, 1996.

[Tyrrell, 1994] Toby Tyrrell. An evaluation of Maes' bottom-up mechanism for behavior selection. *Adaptive Behavior*, 2(4):307–348, 1994.

[Weigel *et al.*, 2001] Thilo Weigel, Willi Auerbach, Markus Dietl, Burkhard Dümler, Jens-Steffen Gutmann, Kornel Marko, Klaus Müller, Bernhard Nebel, Boris Szerbakowski, and Maximilian Thiel. CS Freiburg: Doing the right thing in a group. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, pages 52–63. Springer-Verlag, Berlin, Heidelberg, New York, 2001.

[Weigel *et al.*, 2002] Thilo Weigel, Alexander Kleiner, Florian Diesch, Markus Dietl, Jens-Steffen Gutmann, Bernhard Nebel, Patrick Stiegeler, and Boris Szerbakowski. Cs freiburg 2001. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: Robot Soccer World Cup V*. Springer-Verlag, Berlin, Heidelberg, New York, 2002.

[Zhang *et al.*, 1998] Zhaohua Zhang, Stan Franklin, Brent Olde, Yun Wan, and Art Graesser. Natural language sensing for autonomous agents. In *Proceedings of the IEEE Joint Symposia on Intelligence and Systems*, pages 374–381, Rockville, Maryland, 1998.