2004

When Are Behaviour Networks Well-Behaved?

Bernhard Nebel
Universitat fur Informatik

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

When Are Behaviour Networks Well-Behaved?

Bernhard Nebel[1] and **Yuliya Babovich-Lierler**[2]

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. This problem is addressed by a number of action-selection mechanisms. Behaviour networks as proposed by Maes are one such mechanism, which is quite popular. In general, it seems not possible to predict when behaviour networks are well-behaved. However, they perform quite well in the robotic soccer context. In this paper, we analyse the reason for this success by identifying conditions that make behaviour networks *goal converging*, i.e., force them to reach the goals regardless of the details of the action selection scheme. In terms of STRIPS domains one could talk of *self-solving planning domains*.

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 figure out the best suitable actions under tight time constraints. There exist a number of techniques to address this so-called *action-selection* problem. Optimal *policies* for MDPs [1], *universal plans* [16], *situated automata* [10], *dual dynamics* [2], and *behaviour networks* [12] are some of them. The latter was intended to address 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 [12].

Compared with the other approaches, the latter approach lacks theoretical rigour. However, modelling a domain is easy and straightforward, which is probably one reason for its popularity. For instance, it has been used in the implementation of an intelligent e-mail agent [20] and as the underlying mechanism for generating behaviour of autonomous characters in interactive story systems [15]. Most notably, behaviour networks have been employed in the simulated robotic soccer team *magmaFreiburg* [5] and in the real robotic soccer (F2000 league) team *CS Freiburg* [18; 19]. In both cases, the teams were highly successful. The simulation team *magmaFreiburg* was runner-up in 1999 [5] and *CS Freiburg* won the *RoboCup* world championship in 2000 and 2001 [18; 19].

Although Maes' behaviour networks and variations have been analysed from several perspectives, there are nevertheless many issues that have not been resolved. For example, Dorer [6] describes some experiments where he used behaviour networks in order to solve *blocks-world planning* problems. As it turns out, for some five-block problems, the behaviour network goes into an infinite loop and does not come up with a solution, regardless of the parameter setting. What is even worse is that there is no understanding of when a behaviour network is "well behaved," i.e., when it will necessarily achieve its goals—provided they are reachable at all.

In the soccer systems mentioned, the agents never ran into infinite loops or got stuck. Instead, given enough time they were always able to score a goal against an immobilised opponent. In other words, the soccer behaviour networks appear to be "well-behaved." This raises of course the question under which condition one could guarantee that.

In general, we are interested to find conditions that guarantee that the behaviour network will generate a successful sequence of actions provided there exists one and no exogenous events intervene. Furthermore, we want this guarantee regardless of any parameter setting and the details of how actions are selected.

Behaviour networks with this property will be called **goal converging**. In terms of STRIPS planning domains one could classify such domains as "self-solving." However, are there non-trivial restrictions on the topology of behaviour networks that would guarantee this? As it turns out, there exists such a condition, which indeed holds also of the existing robotic soccer networks.

The rest of the paper is structured as follows. In the next section we sketch the behaviour network approach. In Section 3, we identify two conditions for a behaviour network being goal converging. Based on that, we analyse 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 BEHAVIOUR NETWORKS

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

2.1 Specifying behaviour 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). **Behaviour 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 $\langle pre, eff^+, eff^-, beh\rangle$ with $pre \subseteq \mathcal{P}$ denoting the **preconditions**, $eff^+, eff^- \subseteq \mathcal{P}$ denoting the positive and negative effects, respectively, with $eff^+ \cap eff^- = \emptyset$ and *beh* being the name of an executable **behaviour**, 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 $pre(m), eff^+(m)$, etc.
- Π is a set of **global parameters** used to control the action selection process.

[1] Institut für Informatik, Fakultät für Angewandte Wissenschaft, Albert-Ludwigs-Universität Freiburg, nebel@informatik.uni-freiburg.de
[2] IMMD8, Universität Erlangen, yuliya@immd8.informatik.uni-erlangen.de

Depending on the type of behaviour networks, some variations are possible. For example, in Dorer's [4] *extended behaviour networks*, the state is an assignment of fuzzy values, 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. All these details will not be important for us, though.

2.2 Activation spreading and action selection

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 behaviour network is given in Figure 1.

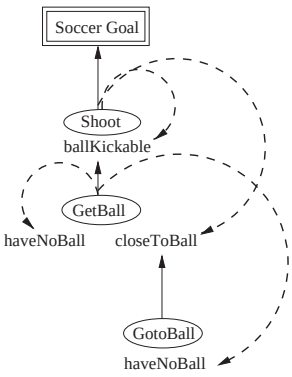

Figure 1. Example of a behaviour network: Solid arrows denote positive effect links and dashed arrows denote negative effect links.

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 behaviour 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 behaviour networks, there is also a "forward spreading" of activation energy, i.e., activation energy flows from propositions true in a situation towards competence modules that have these propositions as preconditions. However, this forward spreading of activation does not seem to increase the quality of the action selection [4; 7] and for this reason this kind of activation is not present in Dorer's [4] extended behaviour 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.

The process of sending activation energy and inhibitions is iterative and the number of iterations is controlled by the global parameters Π. After the process has ended, the utility of a module is determined by combining activation values with executability values (depending on the satisfaction of preconditions), and then the module with the highest utility value is chosen for execution (ties are broken arbitrarily).

From the description above it follows that there are only a few things one can be sure of when using a behaviour 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 [4] 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.

2.3 Ideal abstract behaviour networks

If we want to guarantee properties of a network regardless of the details of the action selection scheme and the 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 behaviour of a competence module is always successful, and that no exogenous event will intervene. Based on these assumptions, we define an abstract version of behaviour networks, which from a formal point of view are identical to STRIPS domain descriptions.

An **ideal, abstract behaviour 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 BEHAVIOUR NETWORKS

If we want to guarantee that a behaviour network is successful regardless of the details of the action selection scheme and the parameter setting,[4] 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 behaviour network **terminating** if for all states and under all possibilities to choose actions, it is impossible to generate infinite

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

[4] The only restriction is that we never consider actions such that all their positive effects are already satisfied (see Section 2.3).

action sequences—provided the goal was reachable initially.[5] Figure 2 gives a simple example of a non-terminating network.[6]

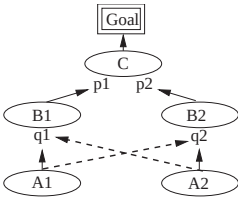

Figure 2. A *non-terminating* behaviour network

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 $A1, B1, A2, B2, C$. 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.[7]

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.

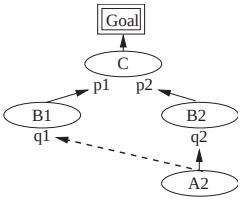

Figure 3. A behaviour network with a *dead end*

Finally, we call a behaviour network **goal converging** when it will necessarily generate a finite action sequence leading to the goal, provided the goal is reachable at all. When viewing the behaviour networks as specifications of STRIPS planning problems, we would talk of **self-solving** planning domains, because regardless of the 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 behaviour 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. ■

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

[6] It becomes terminating if one of the negative edges is removed

[7] Note again that we are not interested in initial states from which the goal is unreachable.

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 the network.

Proposition 2 *Monotone behaviour networks are goal converging.*

Monotone behaviour networks appear hardly to be interesting. However, they are equivalent to STRIPS planning problems that have only positive preconditions and effects, for which it is well known that generating a shortest plan is still an NP-hard problem [3]. Furthermore, such planning problems have become popular as the basis for computing heuristic estimates in action planning [9]. For our purposes, however, the restriction to purely positive effects is not feasible.

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**.[8] In what follows, we identify behaviour networks with their corresponding action-fact graphs to simplify matters. Furthermore, we say that an action-fact graph contains an **effect cycle** if there exists a directed cycle formed out of positive effect edges and reversed negative effect edges. In addition, we say that an action-fact graph is **strictly acyclic** if it neither contains an effect cycle nor a cycle formed by positive edges.

Theorem 3 *Action-fact graphs without effect cycles are terminating.*

Proof: In order to prove the theorem, 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 graph formed by the positive effects edges and the reversed negative effect edges is acyclic, this value assignment is well-defined.

[8] Such graphs correspond to what has been called *connectivity graph* [9] in the planning literature.

With this value assignment to atoms, each action application will strictly increase the overall value of the state (as the 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, something we address next.

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 [8]. While this condition is often true in classical planning tasks, it seems very unlikely that we can guarantee this in our case. Hoffmann [8] gives a number of other sufficient conditions for the absence of dead ends, 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 be used again after it has been falsified. For example, this condition is satisfied in Figure 1. One way to guarantee this for *strictly acyclic action-fact graphs* is to require the following **modularity** condition. For all atoms q that can be falsified by an action a, 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 strictly acyclic action-fact graphs satisfying this condition **modular action-fact graphs**.

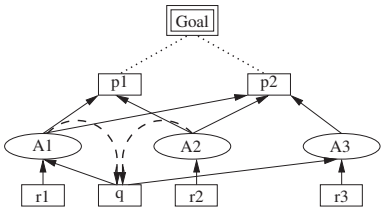

Figure 4. An action-fact graph satisfying the *modularity* condition

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. ■

While it might seem to be the case that such networks are only good for trivial (self-solving) domains, one should keep in mind that finding a shortest sequence of actions is still an NP-hard problem. This follows from the fact that precondition-free, monotone networks are a special case of modular action-fact graphs, for which the problem is already NP-hard [3].

4 ROBOCUP BEHAVIOUR NETWORKS

As mentioned in the Introduction, the analysis of behaviour networks was motivated by the observation that the behaviour networks of the *magmaFreiburg* and *CS Freiburg* robotic soccer players work so robustly. When one now analyses the networks with the tools developed in this paper, it turns out that they indeed satisfy the condition of being *modular*—modulo some qualifications. In Figure 5 the main part of the *CS Freiburg* [14] behaviour network is displayed as an action-

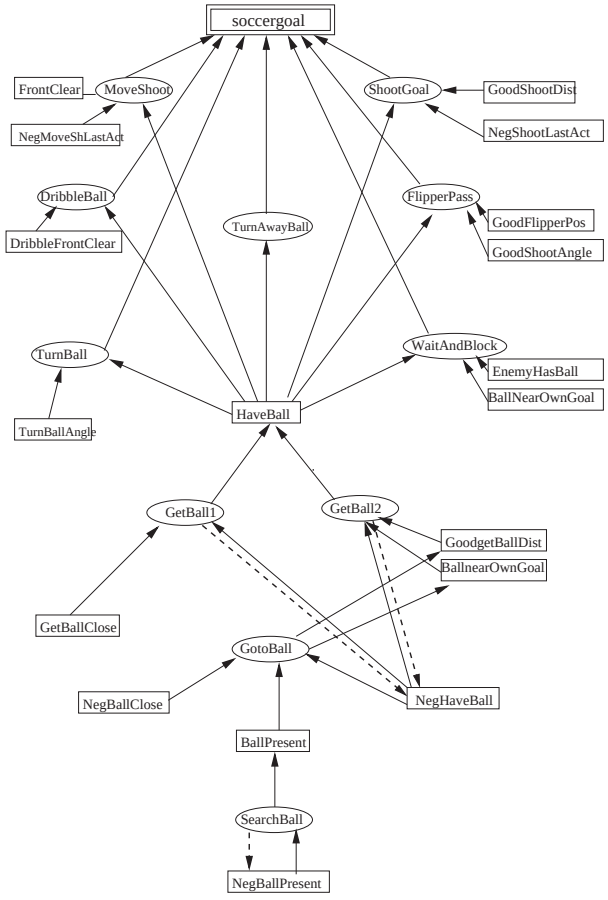

Figure 5. Part of the Action-Fact Graph of the *CS Freiburg* behaviour network [14]

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. A similar comment applies to the missing positive links back to *NegHaveBall*.

Often it is necessary to take more than one goal into account. The extended behaviour 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.

Finally, it should be noted that there are levels in the decision making that influence the behaviour networks, e.g., the role assignment and placement of players on the field [18], which are, however, not part of the network.

Summarising, 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 behaviour networks of the *CS Freiburg* [14] and the *magmaFreiburg* [6] players satisfy the modularity condition and are therefore goal converging, which goes somewhere in explaining why they are so robust. 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, finding a shortest sequence of actions is still NP-hard. Furthermore, 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 on some level of the action-selection process. 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 behaviour networks, called *modularity*, that guarantees that behaviour networks are well behaved, i.e., goal-convergent, which means that they will reach their goals in a static environment under all circumstances—if the goals are reachable at all. One should note that in this case the simplifying assumptions from Section 2.3 are not any longer significant. Any erroneous observation or behaviour or exogenous intervention is compensated for by the behaviour network. It will just start in a new state and is guaranteed to reach the goal (provided the failure probability is low enough).

Having shown that a network has this property means that we never have to fear that the network leads by itself to infinite action sequences or blocked states. In addition, it means that tuning network parameters [13] will not modify the principal property of reaching the goal, but only the efficiency. On the other hand, tuning the parameters might be necessary to approximate the NP-hard optimisation problem of finding a shortest action sequence.

Interestingly, there exists a significant application of behaviour networks where the modularity restriction is met, namely, the networks of the Freiburg simulation and real robot (F2000) soccer players [5; 18; 19]. The most interesting aspect of the entire analysis is, however, that it is not restricted to behaviour networks. It applies to all kinds of domains that can be captured using STRIPS-like formalisms. The property of goal convergence (or almost goal convergence) can be used to understand a given domain and how it shall be dealt with.

Acknowledgements

We acknowledge the comments and suggestions by Jörg Hoffmann, Wolfram Burgard, and Malte Helmert on an earlier version of this paper. The authors have been supported by DAAD as part of the International Quality Network on Spatial Cognition, by DFG (project ROBOT-KOOP) as well as by the University of Newcastle.

REFERENCES

[1] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[2] A. Bredenfeld, T. Christaller, W. Göhring, H. Günther, H. Jaeger, H.-U. Kobialka, P.-G. Plöger, P. Schöll, A. Siegberg, A. Streit, C. Verbeek, and J. Wilberg, 'Behavior engineering with 'dual dynamics' models and design tools', In Veloso et al. [17], pp. 231–242.

[3] T. Bylander, 'The computational complexity of propositional STRIPS planning', *Artificial Intelligence*, **69**(1–2), 165–204, (1994).

[4] K. Dorer, 'Behavior networks for continuous domains using situation-dependent motivations', in *Proc. IJCAI-99*, pp. 1233–1238, Stockholm, Sweden, (1999). Morgan Kaufmann.

[5] K. Dorer, 'The magmaFreiburg soccer team', In Veloso et al. [17], 600–603.

[6] K. Dorer, *Motivation, Handlungskontrolle und Zielmanagement in autonomen Agenten*, Ph.D. dissertation, Albert-Ludwigs-Universität, Freiburg, Germany, 2000. Published on FreiDok server under http://www.freidok.uni-freiburg.de/volltexte/57.

[7] P. Goetz and D. Walters, 'The dynamics of recurrent behavior networks', *Adaptive Behavior*, **6**(2), 247–283, (1997).

[8] J. Hoffmann, 'Local search topology in planning benchmarks: A theoretical analysis,', in *Proc. AIPS-02*. AAAI Press, Menlo Park, (2002).

[9] J. Hoffmann and B. Nebel, 'The FF planning system: Fast plan generation through heuristic search', *Journal of Artificial Intelligence Research*, **14**, 253–302, (2001).

[10] L. P. Kaelbling and S. J. Rosenschein, 'Action and planning in embedded agents', In Maes [11], 35–48.

[11] *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, ed., P. Maes, MIT Press, Cambridge, MA, 1990.

[12] P. Maes, 'Situated agents can have goals', In Maes [11], 49–70.

[13] P. Maes, 'Learning behavior networks from experience', in *Proc. 1st European Conf. on Artificial Life*, pp. 48–57, (1992).

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

[15] B. Rhodes, *PHISH-Nets: Planning Heuristically in Situated Hybrid Networks*, Master's thesis, MIT, Cambridge, MA, 1996.

[16] M. J. Schoppers, 'Universal plans for reactive robots in unpredictable environments', in *Proc. IJCAI-87*, pp. 1039–1046, Milan, Italy, (1987). Morgan Kaufmann.

[17] *RoboCup-99: Robot Soccer World Cup III*, eds., M. Veloso, E. Pagello, and H. Kitano, Springer-Verlag, Berlin, 2000.

[18] T. Weigel, W. Auerbach, M. Dietl, B. Dümler, J.-S. Gutmann, K. Marko, K. Müller, B. Nebel, B. Szerbakowski, and M. Thiel, 'CS Freiburg: Doing the right thing in a group', in *RoboCup-2000: Robot Soccer World Cup IV*, eds., P. Stone, T. Balch, and G. Kraetzschmar, 52–63, Springer-Verlag, Berlin, (2001).

[19] T. Weigel, A. Kleiner, F. Diesch, M. Dietl, J.-S. Gutmann, B. Nebel, P. Stiegeler, and B. Szerbakowski, 'CS Freiburg 2001', in *RoboCup-2001: Robot Soccer World Cup V*, eds., A. Birk, S. Coradeschi, and S. Tadokoro, Springer-Verlag, Berlin, (2002).

[20] Z. Zhang, S. Franklin, B. Olde, Y. Wan, and A. Graesser, 'Natural language sensing for autonomous agents', in *Proc. IEEE J. Symposia on Intelligence and Systems*, pp. 374–381, Rockville, Maryland, (1998).