The DD&P Robot Control Architecture: A Preliminary Report*

Frank Schönherr and Joachim Hertzberg

Fraunhofer - Institute for Autonomous intelligent Systems (AIS)
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
schoenherr|hertzberg@ais.fraunhofer.de

Abstract. The paper presents current results of our work on DD&P, a two layer control architecture for autonomous robots. DD&P comprises a deliberative and a behavior-based part as two peer modules with no hierarchy among these two layers, as sketched in [HJZM98]. Interaction between these control layers is regulated by the *structure* of the information flow, extending the "classical" sense-model-plan-act principle. The paper stresses two architectural highlights of our approach: The implementation of the "Plan-as-Advice" principle to execute plan operators by the behavior-based part, and the grounding of the symbolic planner description via chronicle recognition.

1 Background and Overview

There are several good reasons to include a behavior-based component in the control of an autonomous mobile robot. There are equally good reasons to include in addition a deliberative component. Having components of both types results in a *hybrid* control architecture, intertwining the behavior-based and the deliberative processes that go on in parallel. Together, they allow the robot to react to the dynamics and unpredictability of its environment without forgetting the high-level goals to accomplish. Arkin [Ark98, Ch. 6] presents a detailed argument and surveys hybrid control architectures; the many working autonomous robots that use hybrid architectures include the Remote Agent Project [MNPW98,Rem00] as their highest-flying example.

While hybrid, layered control architectures for autonomous robots, such as Saphira [KMSR97] or 3T [BFG+97] are state of the art, some problems remain that make it a still complicated task to build a control system for a concrete robot to work on a concrete task. To quote Arkin [Ark98, p. 207],

> the nature of the boundary between deliberation and reactive execution is not well understood at this time, leading to somewhat arbitrary architectural decisions.

This boundary can be split into two different sub-problems:

(1) The symbolic world representation *update* problem, which is an instance of the *symbol grounding problem* [Har90]. There are solutions to important parts of that

* This work is partially supported by the German Fed. Ministry for Education and Research (BMBF) in the joint project AgenTec (01AK905B), see http://www.agentec.de for details.

problem, such as methods and algorithms for sensor-based localization to reason about future navigation actions: [FBT99] presents one of the many examples for on-line robot pose determination based on laser scans. If the purpose of deliberation is supposed to be more general than navigation, such as action planning or reasoning about action, then the need arises to sense more generally the recent relevant part of the world state and update its symbolic representation based on these sensor data. We call this representation the current *situation.*

The naive version of the update problem "Tell me all that is currently true about the world!" needs not be solved, luckily, if the goal is to build a concrete robot to work on a concrete task. *Only those facts need updating that, according to the symbolic domain model used for deliberation, are relevant for the robot to work on its task.* Then, every robot has its *sensor horizon,* i.e., a border in space and time limiting its sensor range. The term sensor is understood in a broad sense: It includes technical sensors like laser scanners, ultra sound transducers, or cameras; but if, for example, the arena of a delivery robot includes access to the control of an elevator, then a status request by wireless Ethernet to determine the current location of the elevator cabin is a sensor action, and the elevator status is permanently within the sensor horizon. This information is completed by invariable world state descriptions e.g., global maps of the environment. We assume: *The persistent world state together with the variable information within the sensor horizon is sufficient to achieve satisfying robot performance.*

This said, the task of keeping the facts of a situation up-to-date remains to continually compute from recent sensor data and the previous situation a new version of the situation as far as it lies within the sensor horizon. The computation is based on plain, current sensor values as well as histories of situations and sensor readings or aggregates thereof. Practically, we cannot expect to get accurate situation updates instantly; all we can do is make the situation update as recent, comprehensive, and accurate as possible.

(2) The *execution* of symbolic plans in the physical world. Given a plan, the robot must, first, determine whether the plan is still valid and which step is next for execution, and, second, act in the world according to this current plan step. At present DD&P offers nothing original for the first part, and this paper does not go into any detail concerning it. As in many other hybrid robot control systems in the sequence of STRIPS/Shakey's triangle tables [FHN72], we assume that an execution monitor component permanently determines the progress and feasibility of the recent plan, based on the permanently ongoing symbolic world representation update just described.

For the part of acting concretely according to the current symbolic plan step, the problem is that the abstraction from worldly detail that is wanted and unavoidable for effective planning, makes the translation into, say, physical motion non-unique. On the one hand, immediate reaction to unforeseen events, such as navigating around persons while following a planned trajectory in a crowded hallway, has to dominate the to-be-executed plan: reaction overwrites plan step execution. On the other hand, the robot must stick to its plan, tolerating momentary disadvantages over greedy reactive behavior, to achieve coherent, goal-oriented performance in the long run: plan step execution overwrites reaction. So, executing a plan means permanently negotiating between the forces urging to react and the desire to stick with the plan. Every hybrid robot control architecture must cope with that.

Our view of building hybrid robot controllers involving a behavior-based robot control system (BBS)[Mat99] as reactive component is shaped by our work in progress on the DD&P robot control architecture [HJZM98,HS01,SCHC01]. The demo example we will present is formulated in a concrete BBS framework, namely, Dual Dynamics (DD, [JC97]) and we will illustrate how to blend it with classical propositional action planners.

The rest of this paper is organized as follows. In Sec. 2, we present our approach of formulating BBSs as dynamical systems and give a detailed example in Sec. 3. Sec. 4 discusses our view on a concrete planning system in general. Sec. 5 and Sec. 6 contain the technical contribution of the paper: we first sketch how we assume the deliberation component interferes the BBS control component, and then describe the technique of extracting facts from BBS activation value histories. Sec. 7 discusses the approach and relates it to the literature. Sec. 8 concludes.

2 BBSs as dynamical systems

We assume a BBS consists of two kinds of behaviors: low-level behaviors (LLBs), which are directly connected to the robot actuators, and higher-level behaviors (HLBs), which are connected to LLBs and/or HLBs. Each LLB implements two distinct functions: a target function and an activation function. The target function for the behavior b provides the reference t_b for the robot actuators (*"what to do"*) as follows:

$$t_b = f_b(s^T, s_f^T, \alpha_{LLB}^T) \tag{1}$$

where f_b is a nonlinear vector function with one component for each actuator variable, s^T is the vector of all inputs from sensors, s_f^T is the vector of the sensor-filters and α_{LLB}^T is the vector of activation values of the LLBs. By sensor-filters – sometimes called virtual sensors – we mean *markovian* and *non-markovian* functions used for processing specific information from sensors.

The LLB activation function *modulates* the output of the target function. It provides a value between 1 and 0, meaning that the behavior fully influences, does not influence or influences to some degree the robot actuators. It describes *when* to activate a behavior. For LLB b the activation value is computed from the following *differential equation*:

$$\dot{\alpha}_{b,LLB} = g_b(\alpha_{b,LLB}, OnF_b, OffF_b, OCT_b) \tag{2}$$

Eq. 2 gives the variation of the *activation value* $\alpha_{b,LLB}$ of this LLB. g_b is a nonlinear function. OCT_b allows the planner to influence the activation values, see Sec. 5. The scalar variables OnF_b and $OffF_b$ are computed as follows:

$$OnF_b = u_b(s^T, s_f^T, \alpha_{LLB}^T, \alpha_{HLB}^T) \tag{3}$$

$$OffF_b = v_b(s^T, s_f^T, \alpha_{LLB}^T, \alpha_{HLB}^T) \tag{4}$$

where u_b and v_b are nonlinear functions. The variable OnF_b sums up all conditions which recommend activating the respective behavior (on forces) and $OffF_b$ stands for

adversary conditions to the respective behavior (off forces). The definitions in Eqs. 2 - 4 are exemplified in the next section.

The HLBs implement only the activation function. They are allowed to modulate only the LLBs or other HLBs on the same or *lower* level. In our case, the change of activation values for the HLBs $\alpha_{b,HLB}$ are computed in the same manner as Eq. 2. To result in a stable control system, the levels must "run" on a different *timescale*; HLBs change activation only on a longer term, LLBs on a shorter term.

In the original Dual Dynamics paper [JC97] Eq. 1 was also implemented as a differential equation – therefore the name Dual Dynamics. The global *motor controller* is now integrated into the microcontroller of our robot, making behavior design independent from effects like battery values or floor surface. This avoids the integration of the *closed-loop* controller locally into *every* target function and allows the use of much easier to handle absolute output values. The overall result is very similar to the original approach.

The reason for updating behavior activation in the form of Eq. 2 is this. By referring to the previous activation value α_b, it incorporates a memory of the previous evolution which can be overwritten in case of sudden and relevant changes in the environment, but normally prevents activation values from exhibiting high-frequency oscillations or spikes. At the same time, this form of the activation function provides some *low-pass filtering* capabilities, damping sensor noise or oscillating sensor readings.

Independent from that, it helps to develop stable robot controllers if behavior activations have a tendency of moving towards their boundary values, i.e., 0 or 1 in our formulation. To achieve that, we have implemented g_b in Eq. 2 as a *bistable* ground form (see [BK01] Sec. 4 for details) providing some *hysteresis effect*. Without further influence, this function pushes activation values lower/higher than some threshold β (typically $\beta = 0.5$) softly to 0/1. The activation value changes as a result of *adding* the effects coming from the variables OnF, $OffF$, OCT and the bistable ground form. Exact formulations of the g_b function are then just technical and unimportant for this paper, [KJ02] gives a detailed description.

The relative smoothness of activation values achieved by using differential equations and bistability will be helpful later in the technical contribution of this paper , when it comes to derive facts from the time series of activation values of the behaviors (Sec. 6).

In our BBS formulation, behavior arbitration is achieved using the activation values. As shown in Eqs. 2 - 4, each behavior can interact with (i.e., encourage or inhibit) every other behavior on the same or lower level. The model of interaction between behaviors is defined by the variables OnF and $OffF$.

The output vector or *reference vector* r of the BBS for the robot actuators is generated by summing all LLB outputs by a *mixer*, as follows:

$$r = \kappa \sum_b \alpha_{b,LLB} t_b \tag{5}$$

where $\kappa = (1/\sum_b \alpha_{b,LLB})$ is a normalization factor.

Together with the form of the activation values, this way of blending the outputs of LLBs avoids *discontinuities* in the reference values r_i for the single robots actuators,

such as sudden changes from full speed forward to full speed backward. One could even use voting, “winner-takes-all” or priority-based behavior selection mechanisms very easily, if desired.

DD has recently been extended with *team variables* [BK01], which are exported by a behavior system, so that other behavior systems (robots) can read their current values. For every team variable only the exporting behavior systems is allowed to change its value, for all others it’s value is read-only. This mechanism adds multi-robot cooperation capabilities into the DD framework, fitting perfectly in the global context of agent technology [Age] in which we are doing this work.

DD differs from most other BBS, e.g. [Ark98,SKR95,Ros97], by using dynamical system theory for the definition and *analysis* of behaviors. Furthermore this structure supports a practical interface for the integration of a deliberative control layer, see Sec. 5, 6. The implementation, evaluation and analysis of DD behaviors for different types of robots is supported by a set of tools [Bre00,BGG+99], cf. Fig.1.

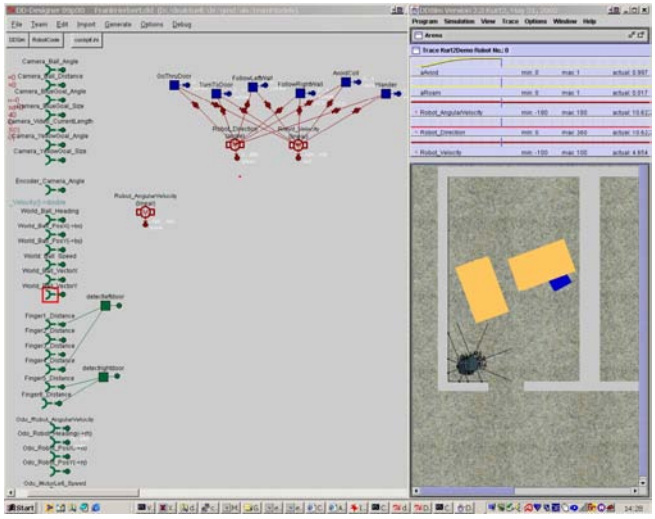

Fig. 1. Screen shots from the most recent version of the DDDesigner tools [Bre00,BGG+99]. These tools mainly comprise an graphical user interface, where DD models can be specified in an intuitive high-level language; automatic code generators for simulation, monitoring, and different kinds of robot hardware; a simulator; and a monitoring environment for wireless online tracing of information processing on board the running robot.

3 An Example

To illustrate the notation of Sec. 2 we give a demonstration problem consisting of the task of following a wall with a robot and entering only those doors that are wide enough

to allow the entrance. Figure 2 gives an overview about the main part of our arena. The depicted robot is equipped with a short distance laser-scanner, 4 infrared side-sensors, 4 front/back bumpers and some dead-reckoning capabilities.

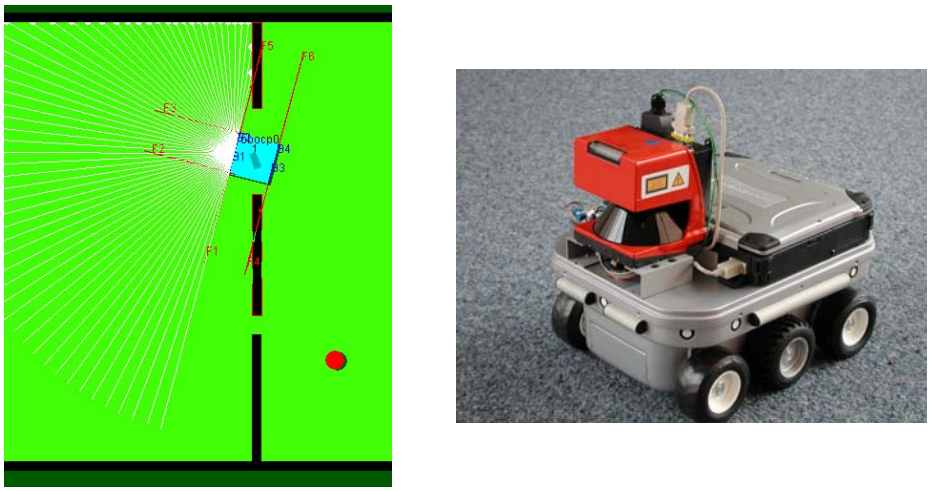

Fig. 2. The left picture shows the demo arena and the final robot pose in Example 1. The robot has started its course at the lower right corner, below to the round obstacle. The right picture shows our office navigation robot KURT2 which was simulated in the example.

We used a simulator based on the DDDesigner prototype tool [Bre00,BGG+99]. The tool allows checking isolated behaviors or the whole BBS in designated environmental situations (configurations).

The control system contains three HLBs and six LLBs, depicted in Tab. 1, with $RobotDirection$ and $RobotVelocity$ as references for the two respective actuators.

Most of the implemented behaviors are common for this kind of tasks. However, we decided to split the task of passing a door in a sequence of two LLBs. This helps structure, maintain and independently improve these two behaviors. The implementation of these behaviors used in this study are deliberately non-sophisticated in order to test the power of the chronicle recognition described below (Sec. 6) in the presence of unreliable behaviors.

To give a simplified example of BBS modeling, here are the "internals" of Wander:

$$OnF_{\text{Wander}} = k_1(1-\alpha_{\text{CloseToDoor}}) * (1-\alpha_{\text{FollowRightWall}}) * (1-\alpha_{\text{FollowLeftWall}}) * (1-\alpha_{\text{AvoidColl}}) \quad (6)$$

$$OffF_{\text{Wander}} = k_2\alpha_{\text{AvoidColl}} + k_3\alpha_{\text{CloseToDoor}} + k_4\alpha_{\text{FollowRightWall}} + k_5\alpha_{\text{FollowLeftWall}} \quad (7)$$

$$RobotDirection_{\text{Wander}} = randomDirection() \quad (8)$$

$$RobotVelocity_{\text{Wander}} = mediumSpeed \quad (9)$$

The HLBs are the following:	
CloseToDoor	is activated if there is evidence for a door;
InCorridor	is active while the robot moves inside a corridor;
TimeOut	was implemented in order to avoid getting stuck in a situation, like two robots blocking each other by trying to pass a door from the opposite sides;
The LLBs are the following:	
TurnToDoor	is depending on CloseToDoor and gets activated if the robot is situated on a level with a door;
GoThruDoor	is activated after the behavior TurnToDoor was successful;
FollowRightWall	is active when a right wall is followed;
FollowLeftWall	is active when a left wall is followed;
AvoidColl	is active when there is an obstacle in the front of the robot
Wander	is active when no other LLB is active.

Table 1. Overview of the Example 1 behaviors. Even if some of their names e.g., CloseToDoor and InCorridor, suggest predicate names, they are all *behaviors*.

$$t_{\text{Wander}} = \begin{bmatrix} RobotDirection_{\text{Wander}} \\ RobotVelocity_{\text{Wander}} \end{bmatrix} \tag{10}$$

$$\dot{\alpha}_{\text{Wander}} = g_{\text{Wander}}(\alpha_{\text{Wander}}, OnF_{\text{Wdr}}, OffF_{\text{Wdr}}, OCT_{\text{Wdr}}) \tag{11}$$

where $k_1 \ldots k_5$ are empirically chosen constants. $randomDirection()$ could be every function that generates a direction which results in a randomly chosen trajectory.

Due to its *product* form, OnF_{Wander} can only be significantly greater than zero if all included α_b are approximately zero. $OffF_{\text{Wander}}$ consists of a *sum* of terms allowing every included behavior to deactivate Wander. Both terms are simple and can be calculated extremely fast, which is a guideline for most BBSs. The OCT term will be explained in the next section. In general, DD terms have a very modular structure – sum of products – which permits thoughtful adaptability to new situations and/or behaviors.

Fig. 3 shows the *activation value histories* generated during a robot run, which will be referred to as Example 1. The robot starts at the right lower edge of the arena in Fig. 2 with Wander in control for a very short time, until a wall is perceived. This effect is explained by Eq. 6. While the robot starts to follow the wall, it detects the small round obstacle in front. In consequence, two LLBs are active simultaneously: AvoidColl and FollowLeftWall. Finally, the robot follows the wall, ignores the little gap and enters the door. In the examples for this paper, FollowRightWall is always inactive and therefore not shown in the activation value curves.

This exemplifies the purpose of a *slow* increase in behavior activation. FollowLeftWall should only have a strong influence to the overall robot behavior if a wall is perceived with *both* side-sensors for some time, so as to be more sure that the robot really has sensed a wall. The small dent in the activation of FollowLeftWall (around the time $t = 4$) is explained by perceiving free space with one side-sensor. If both side-sensors detect free space this behavior would be deactivated. The turning to the door is described by rising/falling edges of some activation values. The second rise of AvoidColl

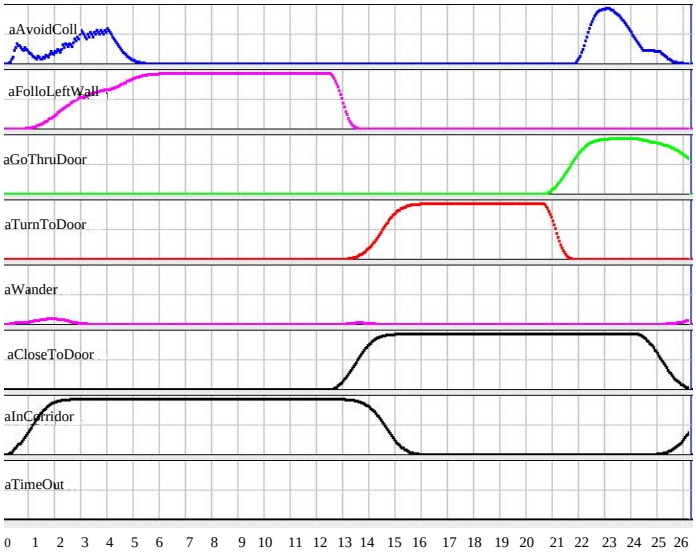

Fig. 3. The activation value histories for Example 1. The numbers are time unit for reference.

(after $t = 22$) is caused by the door frame, which pops into sight as a close obstacle at the very end of the turning maneuver. Effectively, the collision avoidance guides the robot through the door. Finally GoThruDoor gets slowly deactivated allowing other behaviors to take control of the robot.

The HLBs CloseToDoor and InCorridor describe global states, thereby modulating the interaction, activation and sequencing of the LLBs. Some other remarks on behavior arbitration in DD: Our approach is neither purely competitive nor cooperative, allowing internal or external constraints and degree of freedom to be reflected flexibly. For example TurnToDoor and FollowLeftWall are mutually exclusive, while FollowLeftWall and AvoidColl are cooperative.

4 Deliberation in DD&P

At the current work state we have identified some constraints on useful planning systems which arise from the targeted application area [Age], which comprises industrial production, delivery services or inaccessible environments. *Soundness* and *completeness* of the planning system are non-relaxable restrictions in service robotics. Otherwise we could get no guarantee that plans for solvable missions get in fact generated, which is requested by most service robotic clients. The usage of *heuristics* has to be considered very carefully under this assumption. From the conceptual point of view, DD&P is able to work with almost any kind of planner, which we will show in the next section. Please not that the successful plan generation means in now way the successful plan

execution. We just want to exclude additional sources of uncertainty from the planning component.

One should be aware that even a reliable knowledge base update process is not immune against generating *irrelevant* facts. Irrelevant means that these facts are not needed for any solution of the current planning problem. They can enlarge the planner's search space and thereby its search time dramatically. There is work ongoing about solving this problem from the planner side, like e.g. RIFO [NDK97] – which has the disadvantage of not being completely solution preserving.

We are not developing our own planner(s). We are modeling our symbolic task description in (subsets of) PDDL 2.1 [FL01] which enables us to use the full variety of "off-the-shelf planners" that can be downloaded from the Internet. Our idea is to use existing and fast *propositional* systems which generate even non trivial plans very quickly. The hypothesis is that a "throw-away-plan" approach is practically superior to the use of more sophisticated planners. A probabilistic one promises more reliable plans, yet taking much more time for generation. Furthermore assigning probabilities is far from trivial.

In terms of implementation (see Fig. 4), we are currently thinking about IPP [KNHD97] as it adds to pure propositional planning some expressivity (ADL operators) which can be easily expressed in PDDL. An HTN-Planner like SHOP [NCLMA99] is a potential alternative. It permits the use of domain knowledge by decomposing tasks into subtasks and handling domain constraints. This even includes a kind of *plan correction* capability by backtracking other decompositions of high-level (sub-)goals in case of unexecutable actions.

Note that we are not restricting the deliberation part to exactly one planner or to one single planning level. Plans may make sense for a robot in different varieties, on different, possibly unconnected levels of abstraction, and on different degrees of commitment. For example, a path plan and an action plan differ in specificity and, consequently, can efficiently be generated by different algorithms, yet they are both part of the robot's deliberation. Plans on a social level tend to be more abstract than those on individual robot mission level. To yield a satisfying and coherent overall performance, such different plans need to be coordinated in the end. However, DD&P does not achieve this harmonization by forcing the robot domain modeler to employ exactly one planner, but currently leaves open the structure of the deliberative control part.

From the propositional planner point of view execution of an action in the current plan is finished if its specified postconditions hold in the knowledge base or if a replanning process is started. The first one permits skipping of unnecessary actions like opening an already open door. These kind of symbolic actions can arise from different conditions, e.g. imperfect knowledge base entries. Replanning can occur if a *time-out condition* signals a flaw in the execution, the final state of the current plan is achieved, or a serious change in the knowledge-base has been recognized. Serious changes means a new "high-priority" goal or dramatic changes to the current plan, like an inoperative elevator which "cuts off" wide operation areas from the robot. How to define and detect such serious changes is a non-trivial task and lies out of the scope of this paper.

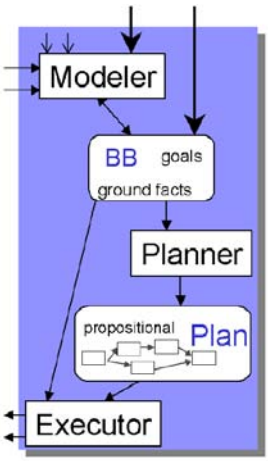

Fig. 4. A "closer-view" on the deliberation part of DD&P. As we do not assume that the system's knowledge base is accurate, it is more aptly called belief base (BB). It includes the current world state and goals. The *Modeler* is updating the world state in the BB, via chronicle recognition (see Sec. 6) or external input, e.g. form other agents. The *Executor* is biasing behaviors towards the current to be executed action, and monitoring its results. See text for details.

5 From symbols to action: Blending behaviors with operators

This section sketches the control flow from the planner to the BBS. The basic idea is this: One of the action planners discussed in Sec. 4 continually maintains a current action plan, based on the current situation and the current set of externally-provided or self-generated mission goals. Based on the current plan and the current situation, an execution component picks one of the operators in the plan as the one currently to be executed. Plan execution is done in a *plans-as-advice* [Pol92,PRK90,Suc87] fashion: Executing an operator means stimulating more or less strongly the behaviors working in favor of the operator, and muting those working against its purpose. This does not mean rigid control, rather giving some degree of freedom to the BBS to react to unforeseen situations, like obstacles, batteries going down or other events which we discuss at the end of Sec. 7, basing concrete execution in the world more on fresh sensor data than on the planner's advice. Which operator stimulates or mutes which behaviors is an information that the domain modeler has to provide along with the domain model for the deliberative component and the set of behaviors for the BBS.

Therefore we first introduce the so-called *influence-matrix* $\mathbf{I}$ with $I_{b,op} \in \mathbb{R}$, which encodes whether and how the current ground operator $op \in OP$ influences the behavior b through the term OCT_b.

Technically, the influence of the *current* operator is "injected" into the BBS in terms of the *Operator-Coupling-Terms* (OCT) in the activation functions, see Eqs. 2. The influence of the *current* ground operator op gets inside every behavior b through the

term OCT_b, as follows:

$$OCT_b = |I_{b,op}|(sign^+(I_{b,op}) - \alpha_b) \tag{12}$$

where $sign^+$ is $+1$ if $x > 0$ and 0 else. $I_{b,op} \neq 0$ iff op influences the behavior b. Its sign expresses whether the operator influence is of the stimulating or muting sort: If $I_{b,op} > 0$, then the respective behavior is stimulated, and muted if $I_{b,op} < 0$. The absolute value of $I_{b,op}$ models the strength of the operator influence on the behavior b.

To give an example, assume that the domain model for the deliberation component includes an operator variable **GO-IN-RM**(x) modeling the action of some office delivery robot to go (from wherever it is) to and enter room x. Let the behavior inventory be the one specified in Sec. 3. Here is a selection of $I_{b,\textbf{GO-IN-RM}(x)}$-column for the influence of a ground operator like **GO-IN-RM**(A) for a concrete room A:

	....	GO-IN-RM(x)	....
CloseToDoor	....	5	....
InCorridor	....	0	....
TimeOut	....	0	....
TurnToDoor	....	0	....
GoThruDoor	....	0	....
FollowRightWall	....	-5	....
FollowLeftWall	....	-5	....
AvoidColl	....	0	....
Wander	....	-2	....

Table 2. A selection of the influence-matrix I. See text for details.

Note that in Tab. 2 $I_{b,op} \in [-40, 40]$ which is in relation to the actual implementation of low-level behaviors "running" with 40 iterations per second. To discuss some values in detail: $I_{\text{CloseToDoor},\textbf{GO-IN-RM}(x)} \neq 0$ means that this behavior is influenced if **GO-IN-RM**(x) is chosen to be executed. In this case, its large *positive* value stimulates CloseToDoor strongly. With the same strength FollowRightWall and FollowLeftWall are muted while Wander is only muted slightly.

The zero influence on TurnToDoor or GoThruDoor does in no way mean that it is not needed to execute **GO-IN-RM**(x). The activation of TurnToDoor is directly depending on the sensor context – perceiving a gap – and on the activation of CloseToDoor which is influenced by this operator. Furthermore GoThruDoor can only be activated in a successful sequence with TurnToDoor, see Sec. 3.

On the other side **GO-IN-RM**(x) can only be chosen for execution if its precondition CloseTo(x) is true in the knowledge base, which also gives strong evidence to be close to the door of room A. Accordingly we are combining the already available capabilities of the BBS in a natural way with that one of the deliberative part.

6 From (re-)action to symbols: Extracting facts from activation values

We now turn to the method for extracting facts from activation value histories. It is influenced by previous work on chronicle recognition, such as [Gha96].

To start, take another look at the activation curves in Fig. 3 in Sec. 3. Some irregular activation time series occur due to the dynamics of the robot/environmen interaction, such as early in the AvoidColl and Wander behaviors. However, certain patterns reoccur for single behaviors within intervals of time, such as a value being more or less constantly high or low, and values going up from low to high or vice versa. The idea to extract symbolic facts from activation values is to consider characteristic groups or *gestalts* of such qualitative activation features occurring in *chronicles* over time.

To make this precise, we define, first, *qualitative activation values* (or briefly, qualitative activations) describing these isolated patterns. In this paper, we consider four of them, which are sufficient for defining and demonstrating the principle, namely, rising/falling edge, high and low, symbolized by predicates $\Uparrow$e, $\Downarrow$e, Hi, and Lo, respectively. In general, there may be more qualitative activations of interest, such as a value staying in a medium range over some period of time. For a behavior b and time interval $[t_1, t_2]$, they are defined as

$$\begin{aligned}
\mathsf{Hi}(b)[t_1,t_2] &\equiv \alpha_b[t] \geq h \text{ for all } t_1 \leq t \leq t_2 \\
\mathsf{Lo}(b)[t_1,t_2] &\equiv \alpha_b[t] \leq l \text{ for all } t_1 \leq t \leq t_2 \\
\Uparrow\mathsf{e}(b)[t_1,t_2] &\equiv \alpha_b[t_1] = l \text{ and } \alpha_b[t_2] = h \text{ and} \qquad (13)\\
&\quad\ \alpha_b \text{ increases } \textit{generally monotonically} \text{ over } [t_1,t_2] \\
\Downarrow\mathsf{e}(b)[t_1,t_2] &\equiv \alpha_b[t_1] = h \text{ and } \alpha_b[t_2] = l \text{ and} \qquad (14)\\
&\quad\ \alpha_b \text{ decreases } \textit{generally monotonically} \text{ over } [t_1,t_2]
\end{aligned}$$

for given threshold values $0 \ll h \leq 1$ and $0 \leq l \ll 1$, where $\alpha_b[t]$ denotes the value of α_b at time t. *General monotonicity* requires another technical definition. We are using a simple first order discrete time filter, the details of which are beyond the scope of this paper. The idea is that some degree of noise should be allowed in, e.g., an increasing edge, making the increase locally non-monotonic. In the rather benign example activation curves in this paper, regular monotonicity suffices. Similarly, it is not always reasonable to use the global constants h, l as Hi and Lo thresholds, respectively. It is possible to use different threshold constants or thresholding functions for different behaviors. We do not go into that here. Then, it makes sense to require a minimum duration for $[t_1, t_2]$ to prevent useless mini intervals of Hi and Lo types from being identified. Finally, the strict equalities in Eq.s 13 and 14 are unrealistic in real robot applications, where two real numbers must be compared, which are seldom strictly equal. Equality $\pm\epsilon$ is the solution of choice here.

The key idea to extract facts from activation histories is to consider patterns of qualitative activations of several behaviors that occur within the same interval of time. We call these patterns *activation gestalts*. We express them formally by a time-dependent predicate AG over a set Q of qualitative activations of potentially many behaviors. For a time interval $[t, t']$ the truth of $AG(Q)[t, t']$ is defined as the conjunction of conditions

on the component qualitative activations $q \in Q$ of behaviors b in the following way:

$$\begin{aligned}
&\text{case } q = \mathsf{Hi}(b) \text{ then } \mathsf{Hi}(b)[t,t'] \\
&\text{case } q = \mathsf{Lo}(b) \text{ then } \mathsf{Lo}(b)[t,t'] \\
&\text{case } q = \Uparrow\mathsf{e}(b) \text{ then } \Uparrow\mathsf{e}(b)[t_1,t_2] \text{ for some } [t_1,t_2] \subseteq [t,t'], \\
&\qquad\qquad \text{and } \mathsf{Lo}(b)[t,t_1] \text{ and } \mathsf{Hi}(b)[t_2,t'] \\
&\text{case } q = \Downarrow\mathsf{e}(b) \text{ then } \Downarrow\mathsf{e}(b)[t_1,t_2] \text{ for some } [t_1,t_2] \subseteq [t,t'], \\
&\qquad\qquad \text{and } \mathsf{Hi}(b)[t,t_1] \text{ and } \mathsf{Lo}(b)[t_2,t']
\end{aligned}$$

Note that it is not required that different rising or falling edges in Q start or end synchronously among each other or at the interval borders of $[t,t']$—they only must all occur somewhere within that interval.

For example, $AG(\{\Uparrow\mathsf{e}(\mathsf{GoThruDoor}), \Downarrow\mathsf{e}(\mathsf{TurnToDoor}), \mathsf{Hi}(\mathsf{CloseToDoor})\})$ is true over $[20,24]$ in the activation histories in Fig. 3; it is also true over $[16,23]$ (and therefore, also over their union $[16,24]$), but not over $[16,25]$, as CloseToDoor has left its Hi band by time 25, and possibly the same for GoThruDoor, depending on the concrete value of the h threshold.

A *chronicle* over some interval of time $[t_0,t]$ is a set of activation gestalts over subintervals of $[t_0,t]$ with a finite set of n linearly ordered internal *interval boundary points* $t_0 < t_1 < \cdots < t_n < t$. A ground fact is extracted from the activation history of a BBS as true (or rather, as *evident*, see the discussion below) at time t if its *defining chronicle* has been observed over some interval of time ending at t. The defining chronicle must be provided by the domain modeler, of course.

We give as an example the defining chronicle of the fact InRoom that the robot is in some room, such as the one left of the wall in Fig. 2. $\mathsf{InRoom}[t]$ is extracted if the following defining chronicle (15) is true within the interval $[t_0,t]$, where the t_i are existentially quantified. Combining a complex sequence of conditions takes into account the wilful simplicity and unreliability of our BBS (see Sec. 3). Accordingly, the resulting – *reliable* – chronicle looks a little bit longish:

$$\begin{aligned}
&AG(\{\Downarrow\mathsf{e}(\mathsf{GoThruDoor})\})[t_4,t] \\
\wedge\ &AG(\{\Uparrow\mathsf{e}(\mathsf{GoThruDoor}), \Downarrow\mathsf{e}(\mathsf{TurnToDoor}), \mathsf{Hi}(\mathsf{CloseToDoor})\})[t_3,t_4] \\
\wedge\ &AG(\{\mathsf{Hi}(\mathsf{TurnToDoor}, \mathsf{Lo}(\mathsf{InCorridor})\})[t_2,t_3] \\
\wedge\ &AG(\{\Uparrow\mathsf{e}(\mathsf{TurnToDoor}), \Uparrow\mathsf{e}(\mathsf{CloseToDoor}), \Downarrow\mathsf{e}(\mathsf{InCorridor})\})[t_1,t_2] \\
\wedge\ &AG(\{\mathsf{Hi}(\mathsf{InCorridor})\})[t_0,t_1] \\
\wedge\ &AG(\{\mathsf{Lo}(\mathsf{TimeOut})\})[t_0,t]
\end{aligned} \tag{15}$$

Assuming reasonable settings of the Hi and Lo thresholds h, l, the following substitutions of the time variables to time-points yield the mapping into the activation histories in Fig. 3: $t = 28$ (right outside the figure), $t_0 = 3, t_1 = 12, t_2 = 16, t_3 = 20, t_4 = 24$. As a result, we extract $\mathsf{InRoom}[24]$.

This substitution is not unique. For example, postponing t_0 until 5 or having t_1 earlier at 9 would also work. This point leads to the process of *chronicle recognition:* given a working BBS, permanently producing activation values, how are the given defining chronicles of facts checked against that activation value data stream to determine whether some fact starts to hold?

The obvious basis for doing this is to keep track of the qualitative activations as they emerge. That means, for every behavior, there is a process logging permanently the qualitative activations. For those of type Hi and Lo, the sufficiently long time periods of the respective behavior activation above and below the h, l thresholds, resp., have to be recorded and, if adjacent to the current time point, appropriately extended. This would lead automatically to identifying qualitative activations of types Hi and Lo with their earliest start point, such as $t_0 = 3$ for Hi(InCorridor) in the example above. Qualitative activations of types $\Uparrow$e and $\Downarrow$e are logged iff their definitions (eqs. 13 and 14, resp.) are fulfilled in the recent history of activation values.

Qualitative activation logs are then permanently analyzed whether any of the existing defining chronicles are fulfilled, which may run in parallel to the ongoing process of logging the qualitative activations. An online version of this analysis inspired by [Gha96] would attempt to match the flow of qualitative activations with all defining chronicles c by means of *matching fronts* that jump along c's internal interval boundary points t_i and try to bind the next time point t_{i+1} as current matching front such that the recent qualitative activations fit all sub-intervals of c that end in t_{i+1}. Note that more than one matching front may be active in every defining chronicle at any time. A matching front in c vanishes if it reaches the end point t (the defining chronicle is true), or else while stuck at t_i is caught up by another matching front at t_i, or else an activation gestalt over an interval ending at t_{i+1} is no longer valid in the current qualitative activation history. The complexity of this process is O(|Behaviors|*|Chronicles|*max|AGsInChronicle|), see [SCHC01] for details.

Practically, the necessary computation may be focused by specifying for each defining chronicle a *trigger condition,* i.e., one of the qualitative activations in the definition that is used to start a monitoring process of the validity of all activation gestalts. For example, in the InRoom definition above, $\Uparrow$e(GoThruDoor), as occurring in the $[t_3, t_4]$ interval, might be used. Note that the trigger condition need not be part of the earliest activation gestalts in the definition. On appearance of some trigger condition in the qualitative activation log, we try to match the activation gestalts prior to the trigger with qualitative activations in the log file, and, if successful, verify the gestalts after the trigger condition in the qualitative activations as they are being logged. [SCHC01] gives an example where the derivation of the InRoom fact fails and a trigger condition would save unnecessary matching effort.

Some more general remarks are in place here. Our intention is to provide fact extraction from activation values as a *main* source of information, not the exclusive one. In general, the obvious other elements may be used for defining them: sensor readings at some time points (be they physical sensors or sensor filters), and the validity of symbolic facts at a time point or over some time interval. That type of information can be added to the logical format of a chronicle definition as in (15). For example, if the exact time point of entering a room with the robot's front is desired as the starting point of the InRoom fact, then this might be determined by the time within the interval $[t_4, t]$ (i.e., within the decrease of the GoThruDoor activation) where some sensor senses open space to the left and right again. As another example, assume that the fact At($Door_A$) for the door to some room A may be in the fact base. Then At($Door_A$)$[t_4]$ could be added to the defining chronicle (15) above to derive not only InRoom$[t]$, but

more specifically $\mathsf{InRoom}(A)[t]$. Such a position information can be easily derived from a normal localization process like [Fox01]. This kind of knowledge base update would be purely sensor related and orthogonal to chronicle recognition. This mechanism can also be used for "instanciated actions" like **GO-IN-RM**(A). Its preconditions mentioned at the end of Sec. 5 would be generated by the planner's knowledge base rule $\mathsf{At}(x) \Rightarrow \mathsf{CloseTo}(x)$.

The fact extraction technique does not presume or guarantee anything about the consistency of the facts that get derived over time. Achieving and maintaining consistency, and determining the ramifications of newly emerged facts remain issues that go beyond fact extraction. Pragmatically, we would not recommend to blindly add a fact as true to the fact base as soon as its defining chronicle has been observed. A consequent of a recognized defining chronicle should be interpreted as evidence for the fact or as a fact *hypothesis*, which should be added to the robot's knowledge base only by a more comprehensive knowledge base update process, which may even reject the hypothesis in case of conflicting information. A possible solution would be to add some integrity constraints to the defining chronicles, like adding InRoom(A) means deleting all InRoom(y) with $y \neq A$. However, this is not within the scope of this paper.

7 Discussion

A physical agent's perception categories must to some degree be in harmony with its actuator capabilities—at least in purposively designed technical artifacts such as working autonomous robots.[1] Our approach of extracting symbolic facts from behavior activation merely exploits this harmony for intertwining control on a symbolic and a reactive level of a hybrid robot control architecture.

The technical basis for the exploitation are time series of behavior activation values. We have taken them from a special type of behavior-based robot control systems (BBSs), namely, those consisting of behaviors expressed by nonlinear dynamical functions of a particular form, as described in Sec. 2. The point of having activation values in BBSs is not new; it is also the case, e.g., for the behavior-based fuzzy control part underlying Saphira [KMSR97], where the activation values are used for context-dependent *blending* of behavior outputs, which is similar to their use in our BBS framework. Activation values also provide the degree of applicability of the corresponding motor schemas in [Ark98, p. 141].

The activation values of a dynamical system-type BBS are well-suited for fact extraction in that their formal background in dynamical systems theory provides both the motivation and the mathematical inventory to make them change smoothly over time—compare, e.g., the curves in Figure 3 with the ragged ones in [SRK99, Fig. 5.10]. This typical smoothness is handy for defining *qualitative activations*, which aggregate particular patterns in terms of edges and levels of the curves of individual behaviors, which are recorded as they emerge over time. These then serve as a stable basis for chronicle recognition over qualitative activations of several behaviors. Note, however, that this

[1] We do not speculate about biological agents in this paper, although we would conjecture that natural selection and parsimony strongly favor this principle.

smoothness is a practical rather than a theoretical issue, and other BBS approaches may serve as bases for fact extraction from activation values.

Using the system dynamics itself as information source seems to be a promising idea even in the area of cognitive psychology. Making the program sensitive to patterns in its own processing, the so called *self-watching*, is an interesting commonality between our ideas and the approach described by Marshall in [Mar99]. However, the targeted application area differs completely from robot control: Marshall computationally models the fundamental mechanisms underlying human cognition on analogy-making in ASCII-strings. In this microdomain, self-watching helps to understand how an answer was found and to detect senseless looping in the system. The latter could be a interesting extension for our approach.

We want to emphasize that the activation values serve three purposes in our case: first, their normal one to provide a reliable BBS, second, to deliver the basis for extracting persistent facts, and third, as an interface for biasing behaviors towards the current symbolic action's intension. With the second use, we save the domain modeler a significant part of the burden of designing a complicated sensor interpretation scheme only for deriving facts. The behavior activation curves, as a by-product coming for free of the behavior-based robot control, focus on the environment dynamics, be it induced by the robot itself or externally. By construction, these curves aggregate the available sensor data in a way that is particularly relevant for robot action. We have argued that this information can be used as a main source of information about the environment; other information, such as coming from raw sensor data, from dedicated sensor interpretation processes, or from available symbolic knowledge, could and should be used in addition.

As activation values are present in a BBS anyway, it is possible to "plug-in" the deliberative component to an already existing behavior system like the DD control system in [BGG+99]. Yet, if a new robot control system is about to be written for a new application area, things could be done better, within the degrees of freedom for variations in behavior and domain model design. The ideal case is that the behavior inventory and the fact set is in harmony in the sense that such facts get used in the domain model whose momentary validity engraves itself in the activation value history, and such behaviors get used that produce activation values producing evidence for facts. For example, a single WallFollow behavior working for walls on the right and on the left, may be satisfactory from the viewpoint of behavior design for a given robot application. For the deliberative part, it may be more opportune to split it into FollowLeftWall and FollowRightWall, which would be equally feasible for the behavior control, but allows more targeted facts to be deduced and behaviors to be biased directly. The fact extraction and behavior biasing scheme leaves the possibility to un-plug the deliberative part from the robot control, which we think is *essential* for robustness of the whole robot system.

Our technique is complementary to anchoring symbols to sensor data as described in [CS01]. It differs from that line of work in two main respects. First, we use sensor data as aggregated in activation value histories only, not raw sensor data. Second, we aim at extracting ground facts rather than establishing a correspondence between percepts and references to physical objects. The limit of our approach is that it is inherently robot-centered in the sense that we can only arrive at information that has to do directly

with the robot action. The advantage is that, due to its specificity, it is conceptually and algorithmically simpler than symbol anchoring in general.

Behavior(-sub)systems have been modeled as self-organizing dynamical systems by other researchers, too. Steinhage presents an approach where the control architecture is very strongly related to dynamical systems theories. Even position estimation, sensor fusion, trajectory planning, event counting, or behavior selection is entirely described in terms of differential equations, see e.g. [SS98] for details. From our point of view, using a planner instead of coding all goal directed knowledge directly into the BBS makes it much easier to define and change "high-level" mission goals and to communicate plans between agents. Most robot control architecture used a kind of situation depending *context* – called context-dependend blending [SRK99] or sensor context [SS98] – which helps regulating the behavior activation. DD&P supports this context – via activation value biasing – in a very flexible way without "touching" the BBS description. Validation of the overall architecture is even easier with a clear view on the operator that is currently being executed.

DD&P permits users not familiar with BBS and robot control to define new tasks for our robots *only* with formal logic. Following the principle of "Plan-as-Advice", a well-defined BBS and influence-matrix would even not allow the user to drive the robot into a wall or to break down with empty batteries. This kind of effect can be even used for *opportunistic* task execution. In a mail-delivery example, a person X that the robot meets by chance on the corridor, can be handed over a letter directly, instead of following the strict plan sequence which would lead – among others – to the office of X. These effects can result from causal links in the current plan – similar to STRIPS's triangle tables [FHN72] – where an operator is skipped if its relevant postconditions are already valid. *Opportunistic* mail-delivery to a person Y that is not included in the current plan can only be achieved by pre-defined conditions.

The fact extraction technique can also be used to monitor the execution of actions. With our example in Sec. 6 the execution of the symbolic action EnterNextRoom can be monitored, simply by means of post-conditions of actions, like InRoom in this case.

Our approach is not in principle limited to a particular combination of deliberation component and BBS, as long as the BBS is expressed as a dynamical system and involves a looping computation of activation values for the behaviors. The method for fact extraction from activation value histories of BBSs presented here is potentially applicable in BBS frameworks other than DD, so long as they are smooth enough to allow some instance of chronicle recognition.

8 Conclusion

We have presented a new approach for extracting information about symbolic facts from activation curves in behavior-based robot control systems. To this end, we use already available, aggregated, filtered, and fused sensor-values instead of re-calculating this data a second time in the symbol grounding component.

Updating the symbolic environment situation is a crucial issue in hybrid robot control architectures in order to bring to bear the reasoning capabilities of the deliberative control part on the physical robot action as exerted by the reactive part. Unlike standard

approaches to sensing the environment in robotics, we are using the information hidden in the temporal development of the data, rather than their momentary values. Therefore, our method promises to yield environment information that is complementary to normal sensor interpretation techniques, which can and should be used in addition.

Biasing behaviors instead of exerting hard control fits nicely into a general architectural design rule: Focusing action execution always on the *current* state of the world. Additionally it offers the chance of opportunistical reactions to unforeseen events.

We have presented these techniques in principle as well as in terms of selected demo examples in a robot simulator, which has allowed to judge the approach feasible and to design the respective algorithms.

Work is ongoing towards a physically concurrent implementation of DD&Pon physical robots, as described in [HS01]. We have recently implemented an alternative approach for fact extraction based on supervised learning [JHS02]. At the present time, we are extending and improving the tools that are available in the context of DD&P. We are confident that a complex software system, like a robot control architecture, can only get over the prototype stage if a proper programming environment is available that supports its general applicability.

Acknowledgments

Thanks to Mihaela Cistelecan, Herbert Jaeger, Hans-Ulrich Kobialka, and the Fraunhofer AiS.BE-Team for their support on this work.

References

[Age] The AgenTec project. http://www.agentec.de.

[Ark98] R. Arkin. *Behavior-Based Robotics*. MIT Press, 1998.

[BFG+97] P. Bonasso, J. Firby, E. Gat, D. Kortenkamp, D. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental & Theoretical Artificial Intelligence: JETAI*, 9:237–256, 1997.

[BGG+99] A. Bredenfeld, W. Göhring, H. Günter, 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 *Proc. 3rd Int. Workshop on RoboCup at IJCAI-99*, pages 57–62, 1999.

[BK01] A. Bredenfeld and H.-U. Kobialka. Team cooperation using dual dynamics. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, number 2103 in LNAI, pages 111–124. Springer, 2001.

[Bre00] A. Bredenfeld. Integration and evolution of model-based tool prototypes. In *11th IEEE International Workshop on Rapid System Prototyping*, pages 142–147, Paris, France, June 2000.

[CS01] S. Coradeschi and A. Saffiotti. Perceptual anchoring of symbols for action. In *Proc. of the 17th IJCAI Conf.*, pages 407–412, Seattle, WA, August 2001.

[FBT99] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *J. Artif. Intell. Research (JAIR)*, 11, 1999.

[FHN72] R.E. Fikes, P.E. Hart, and N.J. Nilsson. Learning and executing generalized robot plans. *Artif. Intell.*, 3:251–288, 1972.

[FL01] M. Fox and D. Long. PDDL2.1 : An extension to PDDL for expressing temporal planning constraints. http://www.dur.ac.uk/d.p.long/pddl2.ps.gz, 2001.

[Fox01] D. Fox. KLD-Sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[Gha96] M. Ghallab. On chronicles: Representation, on-line recognition and learning. In Aiello, Doyle, and Shapiro, editors, *Proc. Principles of Knowledge Representation and Reasoning (KR-96)*, pages 597–606. Morgan-Kauffman, November 1996.

[Har90] S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

[HJZM98] J. Hertzberg, H. Jaeger, U. Zimmer, and Ph. Morignot. A framework for plan execution in behavior-based robots. In *Proc. of the 1998 IEEE Int. Symp. on Intell. Control (ISIC-98)*, pages 8–13, Gaithersburg, MD, September 1998.

[HS01] J. Hertzberg and F. Schönherr. Concurrency in the DD&P robot control architecture. In M. F. Sebaaly, editor, *Proc. of The Int. NAISO Congr. on Information Science Innovations (ISI'2001)*, pages 1079–1085. ICSC Academic Press, march, 17–21 2001.

[JC97] H. Jaeger and Th. Christaller. Dual dynamics: Designing behavior systems for autonomous robots. In S. Fujimura and M. Sugisaka, editors, *Proc. Int. Symposium on Artificial Life and Robotics (AROB'97)*, pages 76–79, 1997.

[JHS02] H. Jaeger, J. Hertzberg, and F. Schönherr. Learning to ground fact symbols in behavior-based robots. In F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002)*, Amsterdam, In Press, 2002. IOS Press.

[KJ02] H.-U. Kobialka and H. Jaeger. Experiences using the dynamical system paradigm for programming robocup robots. unpublished, 2002.

[KMSR97] K. Konolige, K. Myers, A. Saffiotti, and E. Ruspini. The Saphira architecture: A design for autonomy. *Journal of Experimental & Theoretical Artificial Intelligence: JETAI*, 9:215–235, 1997.

[KNHD97] J. Koehler, B. Nebel, J. Hoffmann, and Y. Dimopoulos. Extending planning graphs to an ADL subset. In S. Steel and R. Alami, editors, *Recent Advances in AI Planning. ECP-97*, pages 273–285. Springer (LNAI 1348), 1997. http://www.informatik.uni-freiburg.de/koehler/ipp.html.

[Mar99] J. Marshall. *Metacat: A Self-Watching Cognitive Architecture for Analogy-Making and High-Level Perception*. PhD thesis, Indiana University, Bloomington, IN, USA, November 1999.

[Mat99] M. J. Matarić. Behavior-based robotics. In Robert A. Wilson and Frank C. Keil, editors, *MIT Encyclopedia of Cognitive Sciences*, pages 74–77. The MIT Press, April 1999.

[MNPW98] N. Muscettola, P. Nayak, B. Pell, and B.C. Williams. Remote Agent: to boldly go where no AI system has gone before. *J. Artificial Intelligence*, 103:5–47, 1998.

[NCLMA99] D. S. Nau, Y. Cao, A. Lotem, and H. Munoz-Avila. SHOP: Simple hierarchical ordered planner. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 968–975. Morgan Kaufmann Publishers, 1999.

[NDK97] B. Nebel, Y. Dimopoulos, and J. Koehler. Ignoring irrelevant facts and operators in plan generation. In *ECP-97*, pages 338–350, 1997.

[Pol92] M.E. Pollack. The uses of plans. *Artif. Intl.*, 57(1):43–68, 1992.

[PRK90] D. W. Payton, J. K. Rosenblatt, and D. M. Keirsey. Plan guided reaction. *IEEE Transactions on System, Man and Cybernetics*, 20(6):1370–1382, 1990.

[Rem00] Remote Agent Project. Remote agent experiment validation. http://rax.arc.nasa.gov/DS1-Tech-report.pdf, February 2000.

[Ros97] J. K. Rosenblatt. DAMN: A distributed architecture for mobile navigation. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2/3):339–360, 1997.

[SCHC01] F. Schönherr, M. Cistelecan, J. Hertzberg, and Th. Christaller. Extracting situation facts from activation value histories in behavior-based robots. In F. Baader, G. Brewka, and T. Eiter, editors, *KI-2001: Advances in Artificial Intelligence (Joint German/Austrian Conference on AI, Proceedings*, volume 2174 of *LNAI*, pages 305–319. Springer, 2001.

[SKR95] A. Saffiotti, K. Konolige, and E.H. Ruspini. A multivalued-logic approach to integrating planning and control. *Artificial Intelligence*, 1995.

[SRK99] A. Saffiotti, E.H. Ruspin, and K. Konolige. Using fuzzy logic for mobile robot control. In H-J. Zimmermann, editor, *Practical Applications of Fuzzy Technologies*, chapter 5, pages 185–205. Kluwer Academic, 1999. Handbook of Fuzzy Sets, vol.6.

[SS98] A. Steinhage and G. Schöner. Dynamical systems for the behavioral organization of autonomous robot navigation. In Schenker P. S. and McKee G. T., editors, *Sensor Fusion and Decentralized Control in Robotic Systems: Proceedings of SPIE*, volume 3523, pages 169–180, 1998.

[Suc87] L. A. Suchman. *Plans and Situated Action: The Problem of Human-Machine Communication*. Cambridge University Press, 1987.