Extracting Situation Facts from Activation Value Histories in Behavior-Based Robots[*]

Frank Schönherr, Mihaela Cistelecan[**], Joachim Hertzberg, and Thomas Christaller

GMD – German National Research Center for Information Technology
Institute for Autonomous intelligent Systems (AiS)
Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abstract. The paper presents a new technique for extracting symbolic ground facts out of the sensor data stream in autonomous robots for use under hybrid control architectures, which comprise a behavior-based and a deliberative part. The sensor data are used in the form of time series curves of behavior activation values. Recurring patterns in individual behavior activation curves are aggregated to well-defined patterns, like edges and levels, called *qualitative activations.* Sets of qualitative activations for different behaviors occurring in the same interval of time are summed to *activation gestalts.* Sequences of activation gestalts are used for defining *chronicles,* the recognition of which establishes evidence for the validity of *ground facts.* The approach in general is described, and examples for a particular behavior-based robot control framework in simulation are presented and discussed.

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 work is partially supported by the German Fed. Ministry for Education and Research (BMBF) in the joint project AgenTec (521-75091-VFG 0005B).

[**] On leave from Tech. University of Cluj-Napoca, Romania. The work was supported by a Roman-Herzog scholarship of the Alexander von Humboldt foundation

One of the problems is to keep up-to-date the symbolic world representation for the deliberative component. There are solutions to important parts of that 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. We assume: *The world state 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.

This paper contributes to this task an approach of using histories of activation values in behavior-based robot control systems (BBSs) [Mat99] as a *main* source of information for situation update. This approach is useful for three reasons:

– Activation values are calculated anyway in most BBSs to allow for arbitration or merging between behaviors; they can be used at no additional computation cost, provided that they are sufficiently fine grained.
– An activation value is grounded in sensor readings and, by definition, evaluates them in a way tailored to its respective behavior; using activation values like aggregated sensor readings yields automatically an action-centered way of "looking through the sensors".
– To have a practical hybrid robot control, the symbolic world model must be in accord with the inventory of behaviors anyway; using activation value histories in situation update only makes even more explicit the need to co-design the BBS and deliberative control components.

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. Some additional requirements apply that will be clarified in the paper.

All demo examples are formulated in a concrete BBS framework, namely, Dual Dynamics (DD, [JC97]), which has also inspired our abstract view of BBSs. Our view of building hybrid robot controllers involving a BBS as reactive component is shaped by our work in progress on the DD&P robot control architecture [HJZM98,HS01], which blends DD controllers with action plans generated by a classical propositional planner (concretely, IPP [KNHD97]) as the central deliberation component. Mind, however: The method for fact extraction from activation value histories of BBSs presented here is potentially applicable in BBS frameworks other than DD, as will be discussed at the end of the paper.

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. To provide some background concerning complete robot control systems, we then (Sec. 4) sketch how we assume the deliberation component interferes the BBS control component. Sec. 5 contains the technical contribution of the paper, describing in general as well as by way of example the technique of extracting facts from BBS activation value histories. Sec. 6 discusses the approach and relates it to the literature. Sec. 7 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. 4. 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 contradictory conditions to the respective behavior (off forces).

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.

The reason for updating behavior activation in the form of Eq. 2 is this. By referring to the previous activation value $\dot{\alpha}_b$, 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, deleting 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 (like in [BGG+99] for a RoboCup application of a BBS of the same type) 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.

The relative smoothness of activation values achieved by using differential equations and bistability will be helpful later in the technical contribution of this paper (Sec. 5), when it comes to derive facts from the time series of activation values of the behaviors.

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:

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

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.

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 1 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.

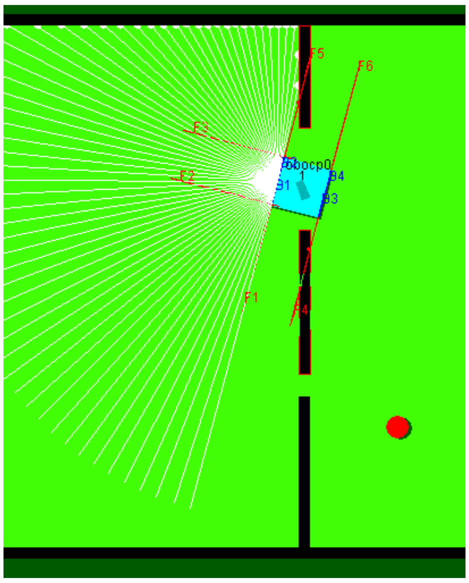

Fig. 1. The demo arena and the final robot pose in Example 1. The robot has started its course at the lower right corner, below to the round obstacle.

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, see Fig. 2. $RobotDirection$ and $RobotVelocity$ are the references for the two respective actuators. We have the following HLBs (cf. Fig. 2):

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, see Sec. 5;

The LLBs are the following:

TurnToDoor is 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.

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.

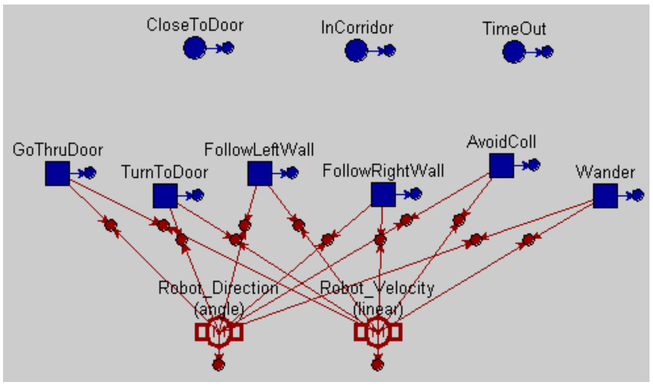

Fig. 2. The behavior inventory for our examples as in a screen shot from the DDDesigner tool. Big circles denote HLBs; squares denote LLBs; the hollow icons at the bottom denote robot actuator reference values. Arrows denote control flow between LLBs and actuators. The influence structure between behavior activations is not shown; the small circles are of no importance here.

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

$$\begin{aligned} OnF_{\text{Wander}} &= k_1(1-\alpha_{\text{CloseToDoor}}) * (1-\alpha_{\text{FollowRightWall}}) * \\ &\quad (1-\alpha_{\text{FollowLeftWall}}) * (1-\alpha_{\text{AvoidColl}}) \quad (6) \\ OffF_{\text{Wander}} &= k_2\alpha_{\text{AvoidColl}} + k_3\alpha_{\text{CloseToDoor}} + \\ &\quad k_4\alpha_{\text{FollowRightWall}} + k_5\alpha_{\text{FollowLeftWall}} \quad (7) \\ RobotDirection_{\text{Wander}} &= randomDirection() \quad (8) \\ RobotVelocity_{\text{Wander}} &= mediumSpeed \quad (9) \\ t_{\text{Wander}} &= \begin{bmatrix} RobotDirection_{\text{Wander}} \\ RobotVelocity_{\text{Wander}} \end{bmatrix} \quad (10) \\ \dot{\alpha}_{\text{Wander}} &= g_{\text{Wander}}(\alpha_{\text{Wander}}, OnF_{\text{Wdr}}, OffF_{\text{Wdr}}, OCT_{\text{Wdr}}) \quad (11) \end{aligned}$$

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 remarkably 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 briefly explained in the next section.

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 Fig. 1 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.

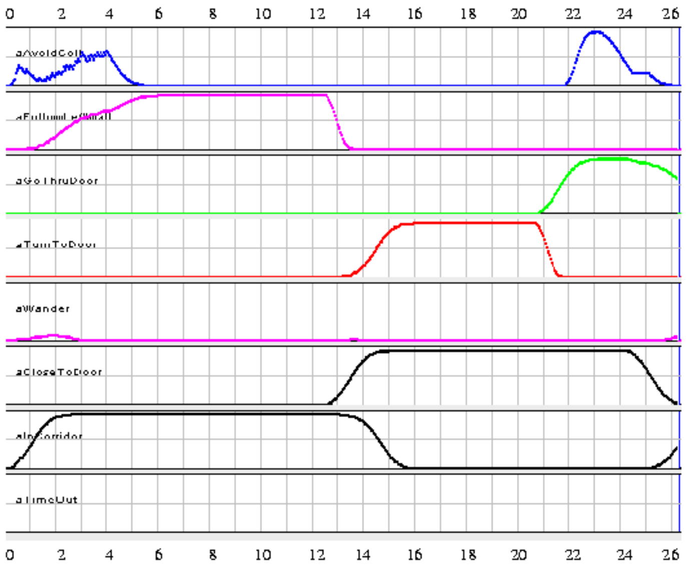

Fig. 3. The activation value histories for Example 1. The numbers are time unit for reference.

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 (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.

4 From plans to BBSs: Blending behaviors with operators

The technical contribution of this paper is an approach to enhancing the information flow from the BBS to the deliberation part in hybrid robot control systems. Before coming to that in Section 5, we want to sketch the control flow in the opposite direction, from the planner to the BBS, to make complete the picture of the entire robot control architecture that we have in mind. Not in the focus of the present paper, this description just consists of stating the basic principle, and we refer to work on the DD&P control architecture [HJZM98,HS01], which elaborates on the approach.

The basic idea is this: An action planner continually maintains a current action plan, based on the current situation and the current set of user-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] 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. 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.

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:

$$OCT_b = \sum_{op \in OP} s_b^{op} c_b^{op} (Z_b^{op} - \alpha_b) \tag{12}$$

where $s_b^{op}, Z_b^{op} \in \{0, 1\}$ and c_b^{op} is a constant. $s_b^{op} = 1$ iff *op* influences the behavior b. c_b^{op} models the immediacy or delay of the operator influence on the behavior. Z_b^{op} expresses whether the operator influence is of the stimulating or the muting sort: If $Z_b^{op} = 1$, then the respective behavior is stimulated, and muted if $Z_b^{op} = 0$. Z may be a boolean function, returning 0 or 1 conditionally.

To give an example, assume that the domain model for the deliberation component includes an operator **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 s, Z, and c variables of these behaviors and how they should be affected by the ground operator **GO-IN-RM**(A) :

$s_{\mathsf{GoThruDoor}}^{\textbf{GO-IN-RM}(A)}$ set to 1 as the operator does influence the behavior;

$c_{\mathsf{GoThruDoor}}^{\textbf{GO-IN-RM}(A)}$ set to some medium value, causing a tendency to influence activation soon after the operator is chosen as being active;

$Z_{\mathsf{GoThruDoor}}^{\textbf{GO-IN-RM}(A)}$ set to *charFct*$(\mathsf{CloseTo}(A))$, i.e., the characteristic function that returns 1 if $\mathsf{CloseTo}(A)$ is currently in the fact base, and 0 else;

$s_{\mathsf{Wander}}^{\textbf{GO-IN-RM}(A)}$ set to 1 as the operator should affect its activation (namely, muting it);

$s_{\mathsf{AvoidColl}}^{\textbf{GO-IN-RM}(A)}$ set to 0 as collision avoidance should not be affected by the operator (note the difference between not affecting and actively muting an activation value)

5 From BBSs to plans: Extracting facts from activation values

We now turn to the method how to extract 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/environment 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

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

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, which we skip here for brevity. 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:

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

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. 1. $\mathsf{InRoom}[t]$ is extracted if the following defining chronicle is true within the interval $[t_0, t]$, where the t_i are existentially quantified:

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

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. As this logging process is local to every behavior, the complexity is linear $O(B)$ in the number B of behaviors.

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.

For complexity considerations, assume that C defining chronicles are defined that involve a maximum of $N - 1$ internal interval boundary points. Assume further that A is the maximal number of qualitative activations occurring in single activation gestalt conjuncts of all defining chronicles. (A is bounded by the number B of behaviors.) Then, one cycle of the online matching of defining chronicles runs in $O(ACN)$ time.

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, $\Uparrow\mathsf{e}(\mathsf{GoThruDoor})$, 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.

To give an example where the derivation of the InRoom fact fails, consider Fig. 4. The scenario is like before, i.e., the robot starts at the lower right corner, driving upward and trying to enter any door large enough. Different to Fig. 1, no obstacle is present at the beginning, and while the robot tries to enter into the detected door, another robot comes from within the room and blocks it. Fig. 4 right shows the scene after the robot has failed to enter the door, and left are the respective activation values.

Like before, while InCorridor ($t_0 = 3$), the CloseToDoor and TurnToDoor activations rise with InCorridor falling ($t_1 = 9, t_2 = 13$); then, TurnToDoor is Hi, while InCorridor is Lo ($t_3 = 20$). But then, mischief strikes. After the long high period of TurnToDoor, TimeOut jumps up, terminating its Lo period, before GoThruDoor has risen. In consequence, t_4 cannot be bound, and the fact InRoom not extracted. If $\Uparrow\mathsf{e}(\mathsf{GoThruDoor})$ was used as a trigger condition in the first place, then no unnecessary matching effort was wasted.

Some more general remarks are in place here. Defining chronicles exclusively in terms of activation gestalts is a special case that we have used in this paper to keep matters focused. 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. Our intention is to provide fact extraction from activation values as a *main* source of information, not the exclusive one. 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

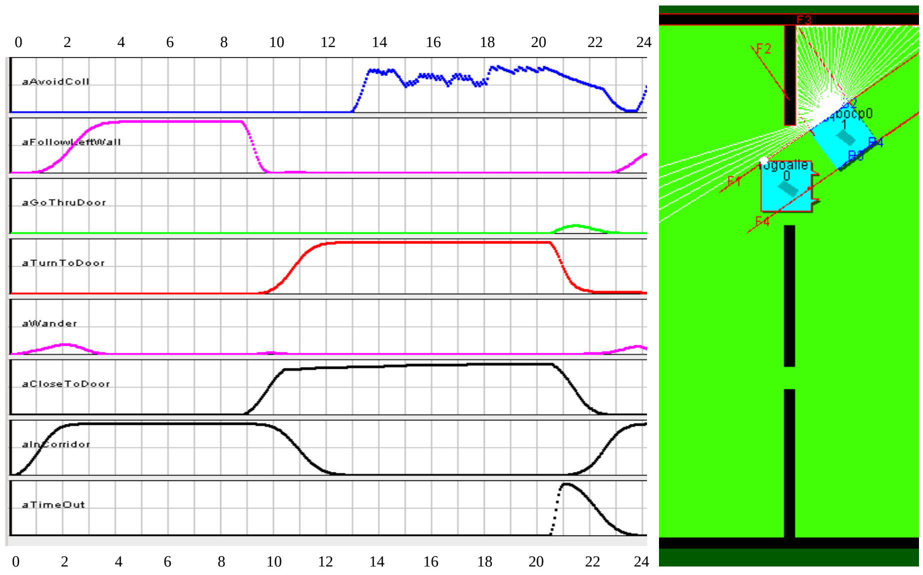

Fig. 4. Robot example 2: final state and activation curves. See text for explanations.

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 (as derived from a normal localization process). Then At$(Door_A)[t_4]$ could be added to the defining chronicle (15) above to derive not only InRoom$[t]$, but more specifically InRoom$(A)[t]$.

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 solutions would be to add some integrity constraints to the defining chronicles. However, this is not within the scope of this paper.

6 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 Figures 3 and 4 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 smoothness is a practical rather than a theoretical issue, and other BBS approaches may serve as bases for fact extraction from activation values.

We want to emphasize that the activation values serve two purposes in our case: first, their normal one to provide a reliable BBS, and second, to deliver the basis for extracting persistent facts, based on their distinctive patterns. 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 be used in addition.

As activation values are present in a BBS anyway, it is possible to ”plug-in” the fact extraction machine for a 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

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

on the left, may be satisfactory from the viewpoint of behavior design for a given robot application; for fact extraction, 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 directly.

Apart from such design-level interdependencies, which are non-trivial, but not special for our approach, we are aiming at a control architecture with a deliberative and a behavior-based part as two abreast modules with no hierarchy, as sketched in [HJZM98]. The fact extraction 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.

7 Conclusion

We have presented a new approach for extracting information about symbolic facts from activation curves in behavior-based robot control systems. 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.

We have presented the technique 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. The computational complexity of the recognition process is in $O(BCN)$, where B is the number of behaviors, C the number of chronicle definitions, and N the maximal “length” of a chronicle definition in terms of intermediate time points internal to the chronicle definition.

The approach will be applied in the context of the hybrid robot control architecture DD&P [HJZM98] to generate an important part of the information that is used to update the symbolic world model from the sensor data stream. Work is ongoing towards a physically concurrent implementation of DD&P on physical robots, as described in [HS01].

Acknowledgements

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

References

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

[BFG$^+$97] P. Bonasso, J. Firby, E. Gat, D. Kortenkamp, D. Miller, and M. Slack. Experiences with an architecture for intelligent, reactive agents. *J. Expt. Theor. Artif. Intell.*, 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.

[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.*, to appear, Seattle, WA, August 2001. Online at http://www.aass.oru.se/˜asaffio/.

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

[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.

[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 Congress 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.

[KMSR97] K. Konolige, K. Myers, A. Saffiotti, and E. Ruspini. The Saphira architecture: A design for autonomy. *J. Expt. Theor. Artif. Intell.*, 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. 4th Eur. Conf. on Planning, ECP-97*, pages 273–285. Springer (LNAI 1348), 1997.

[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, Cambridge, Massachusetts, 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.

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

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

[SRK99] A. Saffiotti, E.H. Ruspini, 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.