USING BEHAVIOUR ACTIVITY SEQUENCES FOR MOTION GENERATION AND SITUATION RECOGNITION

Christopher Armbrust, Lisa Kiekbusch and Karsten Berns
Robotics Research Lab, Department of Computer Science, University of Kaiserslautern
P.O. Box 3049, 67653 Kaiserslautern, Germany

Keywords: Behaviour-based, Activity sequences, Behaviour sequences, Autonomous off-road robot.

Abstract: In this paper, the problem of employing behaviour-based approaches to realise complex, deliberative functionalities on high navigation layers is addressed. Behaviour-based architectures typically target at the lower, more reactive aspects of robot navigation. Hence, it is usually not possible to profit from the advantages of behaviours at higher layers. The authors believe that this limitation is not necessary and suggest the use of behaviour activity sequences as a solution. As central element for the realisation of these sequences, a special generic coordination behaviour is introduced in this paper. It is explained how behaviour activity sequences can be used to generate robot motions as well as to recognise certain situations which a robot may encounter. As examples, the behaviour-based generation of turning manoeuvres and the recognition of dead ends with behaviour activity sequences are shown. The developed concepts demonstrate how to benefit from the typical properties of behaviour-based approaches without limiting the developer to mainly reactive systems.

1 INTRODUCTION

Since the advent of behaviour-based architectures (Brooks, 1986), various types have been created and used in different systems (Arkin, 1998). Their component-based, distributed nature paired with the ability to create large networks by combining numerous rather simple behaviours yields several advantages over classic, typically monolithic systems. For example, a single element of a behaviour-based system can easily be developed, implemented, and tested on its own before being integrated into the target system. Furthermore, a behaviour can easily be used in different systems, which facilitates the reuse of functionality. This is usually more difficult or even impossible in monolithic architectures. Behaviour-based systems often possess strong reactive elements and thus are especially well-suited for realising low-level navigation tasks like collision avoidance or point approaching. By combining behaviour-based elements with classic, deliberative components, hybrid systems are created that feature high-level navigation functionalities but still profit from the advantages of behaviour-based approaches (Habib, 1999).

Already in (Matarić, 1997) it is explained that "behavior-based approaches are an extension of reactive systems". According to that work, they are not limited to looking-up and computing simple functional mappings. Furthermore, they are able to store arbitrary forms of state information. Despite this definition, most (so-called) behaviour-based architectures lack direct support for the realisation of complex, deliberative functionalities on high navigation layers. As a result, developers of high-level navigation systems cannot benefit from the numerous advantages of behaviour-based approaches. However, the limitation to mainly reactive tasks is unnecessary. In the work at hand, the use of behaviour activity sequences to overcome this limitation is proposed. Using such sequences, a navigation system can create complex manoeuvres or recognise situations by activating a number of behaviours one after the other. Each single behaviour does not have to be complex, but the complexity emerges from the interaction of the single elements of the system. That way, the developer can benefit from the advantages of behaviour-based systems while being able to create a system offering complex, high-level functionalities. For example, using behaviour-based approaches to realise complex state transitions helps to get an insight into a system's state in a fast and easy way as the complexity of the system is not "hidden" inside a monolithic component, but spread over the network.

The paper at hand is structured as follows: After

Armbrust C., Kiekbusch L. and Berns K..
USING BEHAVIOUR ACTIVITY SEQUENCES FOR MOTION GENERATION AND SITUATION RECOGNITION.
DOI: 10.5220/0003443501200127
In *Proceedings of the 8th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2011), pages 120-127
ISBN: 978-989-8425-75-1

the introduction, Section 2 gives an overview of the related work, while Section 3 introduces the concept for realising behaviour activity sequences. In Section 4, two applications are presented as examples of the use of said sequences in the control system of an autonomous off-road robot. A conclusion along with an outlook on future work finishes the paper.

2 RELATED WORK

As already mentioned in the preceding section, many complex robot control systems are hybrids. They are often built up of three layers dealing with navigation (Gat, 1998; Ranganathan and Koenig, 2003), with the lowest layer being mostly reactive and the highest layer being designed in a deliberative manner. The layer in-between is typically a more or less wide interface between the two, which takes commands from the highest layer, controls and monitors the reactive elements of the lowest one, and sends feedback upwards.

In outdoor robotics, the top layer usually deals with long-range (i.e. global), coarse-grained navigation (Giesbrecht, 2004), while the bottom layer realises short-range (i.e. local), fine-grained navigation like collision avoidance. One way to build up the middle layer is to realise motion planning with a scope and a granularity in-between. However, such deliberative approaches tend to be based on monolithic components and not on single elements like behaviour-based systems. Thus, they lack the advantages of the latter. Furthermore, there is a breach between the elements following different architectural paradigms, which renders the creation of the interface especially crucial.

But there are also behaviour-based concepts which support the realisation of more deliberative tasks. In (Maes, 1990) an architecture is described in which activation transfer between behaviours can be used to implicitly create activity sequences. Unfortunately, the paper only provides results obtained with simulated pick-and-place tasks. The theory presented is not applied to a complex robotic system. By contrast, the concepts developed in the paper at hand have been implemented into a system consisting of over 500 behaviours that controls a mobile robot within complex environments (see Section 4).

Another approach that allows for the use of behaviours for more sophisticated tasks than the typical reactive ones is followed by the authors of (Nicolescu and Matarić, 2002), whose work is a theoretical basis for this paper. In order to add support for temporal sequences in behaviour-based systems, they created an architecture in which behaviours can be connected using their so-called effects output and preconditions input ports. The values of the precondition ports are checked with respect to the fulfillment of certain conditions, of which there are three different types: enabling, ordering, and permanent ones. By this means, complex networks realising behaviour activity sequences can be realised. However, the behaviour signals in this approach were restricted to $\{0,1\}$, a limitation that is overcome by the work presented here. Furthermore, their experiments were conducted in simple, artificial environments and the architecture was not used to recognise complex structures which can be found in off-road environments.

3 BEHAVIOUR SEQUENCES

The work at hand is an extension of the behaviour-based architecture iB2C[1] (Proetzsch, 2010). In iB2C, all behaviours have a common interface for transferring so-called behaviour signals between them (see Figure 1). While the *stimulation* s is used to gradually enable a behaviour, the vector $\vec{\imath}$ of k inhibitory inputs is used to gradually disable it. The combined value $\iota = s \cdot (1 - i)$ with the *inhibition* $i = \max_{j=0,\dots,k-1} \{i_j\}$ is called *activation* and defines the maximum influence of a behaviour within a behaviour network. The degree of influence a behaviour intends to have and its satisfaction with the current situation are expressed by its *activity* a and *target rating* r, respectively. So-called derived activities $\underline{a}_0, \underline{a}_1, \dots, \underline{a}_{q-1}$ with $\underline{a}_i \le a \ \forall i \in \{0,1,\dots,q-1\}$ together with a behaviour's activity build the activity vector $\vec{a} = (a, \underline{\vec{a}})^T$. To allow for an easy connection of several behaviours, the values of these behaviour signals are limited to $[0,1]$. In addition to the standardised ports, a behaviour can have an arbitrary number of ports for control data. The output vector $\vec{u}$ is calculated as $\vec{u} = F(\vec{e}, \iota)$ with $\vec{e}$ being a vector of control inputs and F the behaviour's transfer function.

3.1 Coordinating Behaviour

In contrast to the use of complex behaviours which coordinate the activation of multiple other behaviours and thus realise behaviour activity sequences, a much more simple yet generic iB2C behaviour shall be used here for this task, called *Conditional Behaviour Stimulator* or simply CBS (see Figure 2). The idea is that it gets active if certain *input conditions* concerning the values at a set of its input ports are fulfilled. To these ports, activity or target rating outputs of so-called *in-*

[1] iB2C: integrated Behaviour-Based Control.

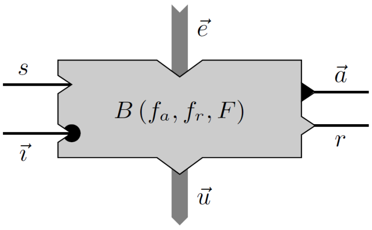


Figure 1: The general symbol of a behaviour showing stimulation s, inhibition vector $\vec{\imath}$, activity vector $\vec{a}$, and target rating r. $\vec{e}$ and $\vec{u}$ denote the input and output vectors, respectively. The transfer function F calculates u with respect to e and the internal activation ι.

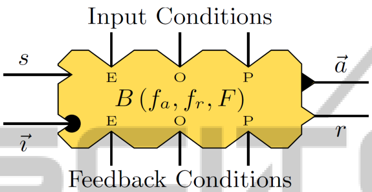


Figure 2: The symbol of a CBS module depicting the three different types of ports (enabling, ordering, and permanent) for input conditions (top) and feedback conditions (below). As a CBS is a behaviour, it also features the standard behaviour ports.

put behaviours are connected. Hence the input conditions are fulfilled depending on the activities or target ratings of other behaviours. With the CBS' activity, another behaviour of the network, the *stimulated behaviour*, can then be stimulated.

Once active, a CBS monitors the values at a second set of its input ports. If certain conditions concerning these values are fulfilled, the CBS' activity goes down to zero again. As this can be used to send a feedback from a stimulated behaviour to the CBS, these conditions are called *feedback conditions*. By cascading such constructs, arbitrarily complex behaviour activity sequences can be created.

A CBS acts as a coordinating element in a behaviour-based network. (Saffiotti, 1997) identified two aspects of the behaviour coordination problem, namely how to decide which behaviour(s) to activate and how to combine the outputs of different behaviours into one command sent to a robot's actuators. In (Pirjanian, 1999), these two aspects are further distinguished, resulting in a taxonomy with seven different types of action selection mechanisms. The behaviour coordination described in the paper at hand falls into the category of state-based arbitration. The state is determined by the input behaviours connected to a CBS. By processing the input values as described below, the CBS evaluates the state, and this evaluation can constitute the basis for a behaviour arbitration.

3.2 Condition Evaluation

The conditions belonging to a CBS are evaluated as described in the following. A relation ir_j comparing an input value with a threshold is attributed to each input condition ic_j. Such a relation is defined as follows:

$$\mathrm{ir}_j(t) = \begin{cases} 1 & \text{input\_value}_j\ \mathrm{REL}_j\ \text{input\_threshold}_j \\ 0 & \text{else} \end{cases}, \qquad (1)$$

where $j = \{1, \ldots, m\}$, m is the number of conditions, t is the time, and $\mathrm{REL}_j \in \{<, \leq, =, \geq, >\}$. Feedback relations fr_j are attributed to the feedback conditions fc_j accordingly. Note that $\text{input\_value}_j(t) \in [0,1]$, i.e. there is no limitation to $\{0,1\}$.

As in (Nicolescu and Matarić, 2002), three different types of conditions are distinguished here. They can be used for the connection of input behaviours as well as feedback behaviours to the CBS. In favor of a better understanding, the focus shall be on input conditions in the following and feedback conditions shall be neglected. The three different types of conditions are:

1. **Permanent:** The corresponding relation has to be fulfilled during the whole time when the behaviour shall be active, i.e. the condition is fulfilled if and only if the relation is fulfilled (cp. Equation 2).
2. **Ordering:** The corresponding relation has to be fulfilled at some point in time before the behaviour shall get active. The condition will stay fulfilled independent of whether the relation stays fulfilled or not (cp. Equation 3).
3. **Enabling:** The corresponding relation has to be fulfilled at the exact point in time when the behaviour shall get active. After that, the condition stays fulfilled independent of the fulfillment of the relation (cp. Equation 4).

The following equations express this in a more formal way.

Permanent:

$$(\mathrm{ic}_{\mathrm{CBS}})_j(t) = \begin{cases} 1 & \text{if } \mathrm{ir}_j(t) = 1 \\ 0 & \text{else} \end{cases} \qquad (2)$$

The fulfillment of a permanent condition at time t is equal to the fulfillment of the corresponding relation at time t.

Ordering:

$$(\mathrm{ic}_{\mathrm{CBS}})_j(t) = \begin{cases} 1 & \text{if } \exists t_0 \leq t : \mathrm{ir}_j(t_0) = 1 \\ 0 & \text{else} \end{cases} \qquad (3)$$

An ordering condition is fulfilled at time t if there is a point in time $t_0 \leq t$ at which the corresponding relation was fulfilled. Note that no assumption is made about the fulfillment of the relation after this moment t_0.

Enabling:

$$
(\mathrm{ic}_{\mathrm{CBS}})_j(t) = \begin{cases} 1 & \text{if } \exists t_0 \leq t : \left(\bigwedge_{\substack{k=1 \\ \mathrm{ic}_k \text{enabling}}}^{m} \mathrm{ir}_k(t_0) = 1 \right) \\ & \wedge \left(\bigwedge_{\substack{k=1 \\ \mathrm{ic}_k \text{ordering}}}^{m} \mathrm{ic}_k(t_0) = 1 \right) \\ & \wedge \left(\bigwedge_{\substack{k=1 \\ \mathrm{ic}_k \text{perm.}}}^{m} \mathrm{ic}_k(t_1) = 1 \quad \forall t_1 : t_0 \leq t_1 \leq t \right) \\ 0 & \text{else} \end{cases} \tag{4}
$$

Enabling conditions are the most complex ones. In order to determine whether an enabling condition is fulfilled at time t, it has to be checked whether there is a point in time $t_0 \leq t$ at which all other conditions were fulfilled. Furthermore, all conditions have to be fulfilled from t_0 on. The fulfillment at t_0 is easy to check for all types of conditions. To avoid cyclic dependencies between different enabling conditions, the relations ir instead of the conditions are checked for them. The fulfillment for all t_1 with $t_0 \leq t_1 \leq t$ follows directly for enabling and ordering conditions. For permanent conditions, by contrast, it has to be checked separately.

The behaviour signals of a CBS are calculated as follows:

$$
\begin{aligned}
a_{\mathrm{CBS}}(t) &= s_{\mathrm{CBS}}(t) \cdot (1 - i_{\mathrm{CBS}}(t)) \cdot \prod_{j=1}^{m} (\mathrm{ic}_{\mathrm{CBS}})_j(t) \\
&= \iota_{\mathrm{CBS}} \cdot \prod_{j=1}^{m} (\mathrm{ic}_{\mathrm{CBS}})_j(t)
\end{aligned} \tag{5}
$$

$$
r_{\mathrm{CBS}}(t) = \prod_{j=1}^{m} (\mathrm{ic}_{\mathrm{CBS}})_j(t) \tag{6}
$$

3.3 Example Network

Figure 3 shows a simple behaviour network in which a CBS connects three input behaviours and one stimulated behaviour. The input conditions are fulfilled from the point in time at which the enabling input behaviour has an activity > 0 and the ordering input behaviour had (or still has) an activity of 1. Furthermore, the activity of the permanent input behaviour has to be < 0.5 as long as the input conditions shall be fulfilled.

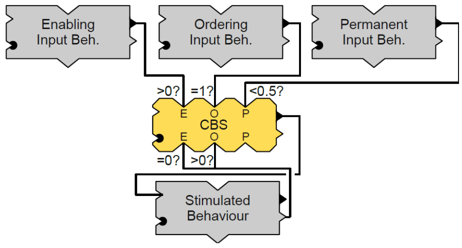


Figure 3: A simple exemplary network.

Figure 4: The autonomous off-road robot RAVON.

There are three connections between the CBS and the stimulated behaviour: The activity output of the CBS is connected to the stimulating input of the stimulated behaviour, resulting in a stimulation as soon as the activity of the CBS rises. Two further connections have been drawn from the target rating output of the stimulated behaviour to two inputs of the CBS associated with two feedback conditions. If the stimulated behaviour is not fully satisfied with the situation ($r > 0$), it will start executing its task as soon as it gets stimulated by the CBS. The execution of the task can take an arbitrary amount of time. During this time, the stimulated behaviour will alter the situation in a way that increases its satisfaction. Full satisfaction is expressed by its target rating going back down to 0. This signals the CBS that the stimulated behaviour has completed its job and needs no further stimulation. It shall be mentioned here that, of course, the stimulated behaviour can fail in achieving its goal, resulting in its target rating stay above 0. This kind of situation is not unusual in complex system and can be dealt with, for example, by a behaviour that monitors the situation and takes action if no progress is made.

4 APPLICATIONS

Two examples shall illustrate the use of networks with behaviour sequences. The first one shows how a turning manoeuvre can be realised, while the second one describes how dead ends can be recognised using behaviour sequences. Both networks have been integrated into the control system of the off-road vehicle RAVON[2] (see Figure 4 and (Armbrust et al., 2010)). The control system has been implemented using the software framework MCA2-KL[3]. In order to limit the influence of other subsystems that might react on external influences and thereby complicate the interpretation of the robot's overall behaviour, the experiments shown here have been conducted in simulation. However, the behaviour-based network used is the same as the one controlling the real robot and consists of over 500 single behaviours.

4.1 Turning Manoeuvre

If a vehicle with a kinematics that does not allow for turning on the spot shall be turned around in a narrow place between obstacles, several back and forth manoeuvres are necessary. A component of a robot control system realising this task has to determine whether the robot is in a situation where conducting such manoeuvres is desired and then decide in which direction to turn around. Figure 5 shows a behaviour-based network realising this task.

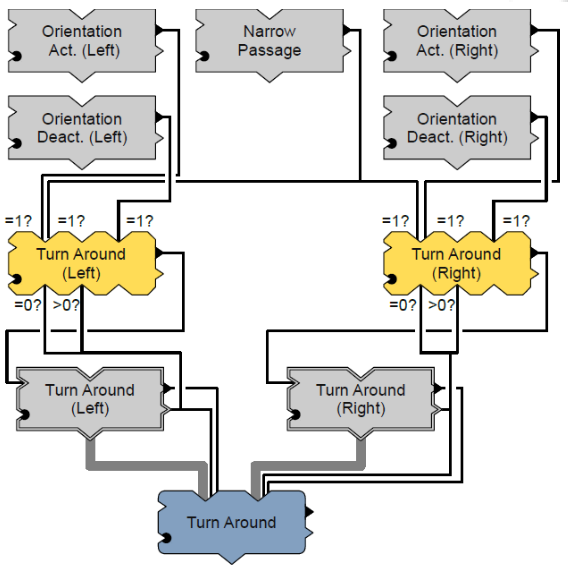


Figure 5: The behaviour-based network for turning around.

The network consists of two strings, one for turning back and forth to the left and one for turning to the right. For each side, two behaviours (*Orientation Activation* and *Orientation Deactivation*) check whether

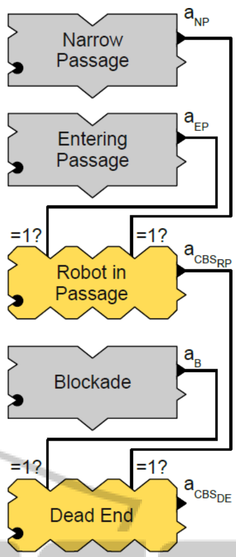


Figure 6: The behaviour-based network that recognises dead ends.

the robot's angle to the target is within certain limits. Their activities are determined by the following equations:

$$a_{\text{Orient. Act.}}(t) = \begin{cases} \iota_{\text{Orient. Act.}}(t) & \text{if } \alpha_{\text{Target}}(t) \geq \alpha_{\text{Act.}} \\ 0 & \text{else} \end{cases} \tag{7}$$

$$a_{\text{Orient. Deact.}}(t) = \begin{cases} \iota_{\text{Orient. Deact.}}(t) & \text{if } \alpha_{\text{Target}}(t) \geq \alpha_{\text{Deact.}} \\ 0 & \text{else} \end{cases} \tag{8}$$

α_{Target} denotes the robot's angle to the target. $\alpha_{\text{Act.}}$ and $\alpha_{\text{Deact.}}$ are two thresholds used to determine when turning manoeuvres shall be executed. They have been set to 90° and 45°, respectively. Together with the use of the behaviours' activities as enabling and permanent input conditions, respectively, for a CBS, this results in turning manoeuvres being initiated if the robot's angle to the target is above 90° and being continued until the angle falls below 45°, realising a hysteresis function. The activity of an additional behaviour (*Narrow Passage*) is used as enabling input condition: It checks whether the robot is situated in a narrow place. If this is not the case, no back and forth manoeuvres are initiated. The reason for this is that RAVON's main sensor systems (like the ones of many robots) are mounted at the front. Therefore, it shall only drive backwards if absolutely necessary.

Depending on the fulfillment of the input conditions, motion commands turning the robot to the left or the right will be initiated by the behavioural groups at the bottom (*Turn Around (Left)* and *Turn Around (Right)*). Each of them is connected to the feedback condition ports of the corresponding CBS in the way

[2] RAVON: Robust Autonomous Vehicle for Off-road Navigation.

[3] MCA2-KL: Modular Controller Architecture Version 2 - Kaiserslautern Branch.

the stimulated behaviour in Figure 3 is. Hence, a CBS will stay active and thus continue to stimulate a group until the group's target rating has risen and then gone back down to zero. The group's outputs are combined by a behaviour fusion, which yields the network's output. Note that the change from forward to backward motion and vice versa is also realised with the help of CBSes in *Turn Around (Left)* and *Turn Around (Right)*, but is beyond the scope of the work at hand.

The advantage of realising such a functionality using behavioural networks is that the activity of the behaviours directly provides information about the state of the system. For example, if the robot unexpectedly does not execute back and forth movements, then by looking at the behaviour network with a special analysis tool (see Figures 7c and 7d as well as (Proetzsch, 2010)) one can directly evaluate which of the conditions is not fulfilled. This is a benefit especially for more complex tasks and hence more complex networks than the one given here as example.

4.2 Dead End Recognition

Properly dealing with dead ends is an important ability for robots operating in complex environments. Approaches using classic path planning algorithms can deal with them as long as the whole dead end is represented in their world model. RAVON, for example, features path planning on a robot-centred grid map with an extent of 50 m along each axis. That means dead ends cannot be detected as complete structure directly if they do not fit into this area. As a result, a classic path planning algorithm will probably lead the robot into a dead end as soon as the corresponding blockade is not within the map anymore. The approach presented here uses a behaviour network to recognise dead ends that are larger than the available world model.

In the context of this work, a dead end is defined as a structure consisting of three elements:

1. *Passage Entry*: an opening between two obstacles that leads into the dead end.
2. *Narrow Passage*: a corridor between obstacles in which the robot cannot drive left or right but has to move on (or back off, of course).
3. *Blockade*: the end of the passage, which keeps the robot from moving on.

The idea here is to detect these three elements in a certain sequence. A behaviour of the network shall get active if the network has found out that the robot is within a dead end. Of course, it must have driven into it sometime. So driving through a passage entry is the first condition, while the second is that the robot is within a narrow corridor all the time after entering the (probable) dead end. If at some point the robot could also drive on to the left or the right, the structure would not be regarded as dead end. Finally, the robot must be faced with a blockade of its path, which is the third condition.

Figure 6 depicts a behaviour network that realises this functionality. Three rather simple behaviours are used for the detection of the different aspects: *Entering Passage*, *Narrow Passage*, and *Blockade*. The CBS *Robot in Passage* indicates that the robot has driven through a passage entry and is now within a narrow corridor, while the CBS *Dead End* shows whether the robot is in a dead end or not.

The activity of *Entering Passage* (a_{EP}) is defined as follows:

$$\begin{aligned} a_{\text{EP}}(t) &= \iota_{\text{EP}}(t) \cdot \text{passage\_entered}(t) \\ &= \iota_{\text{EP}}(t) \cdot \begin{cases} 1 & \text{if the robot has driven} \\ & \text{through a passage's entry} \\ & \text{before time } t \\ 0 & \text{else} \end{cases} \end{aligned} \tag{9}$$

Assuming that *Entering Passage*, *Narrow Passage*, and *Blockade* as well as the two CBSes *Robot in Passage* and *Dead End* are always fully stimulated and not inhibited, Equation 9 becomes:

$$a_{\text{EP}}(t) = \begin{cases} 1 & \text{if the robot has driven} \\ & \text{through a passage's entry} \\ & \text{before time } t \\ 0 & \text{else} \end{cases} \tag{10}$$

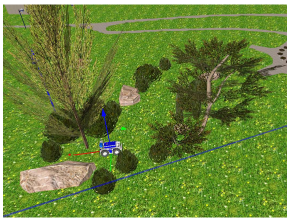
Accordingly, the following equations hold:

$$a_{\text{NP}}(t) = \begin{cases} 1 & \text{if the robot is within a narrow} \\ & \text{passage at time } t \\ 0 & \text{else} \end{cases} \tag{11}$$

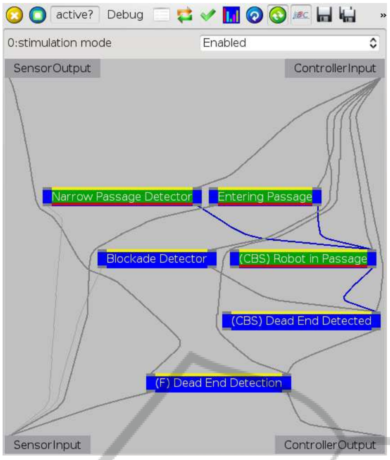
$$a_{\text{B}}(t) = \begin{cases} 1 & \text{if a blockade is detected in front} \\ & \text{of the robot at time } t \\ 0 & \text{else} \end{cases} \tag{12}$$

As the detection of a dead end shall be indicated by $a_{\text{CBS DE}} = 1$, the question is when this is the case. Intuitively, this will be the case when the robot faces a blockade after it has entered a narrow passage which it has not left before encountering the blockade. Hence, the behavioural network fulfills the task of recognising dead ends. A formal analysis of the activity of *Robot in Passage* with Equations 2 and 4

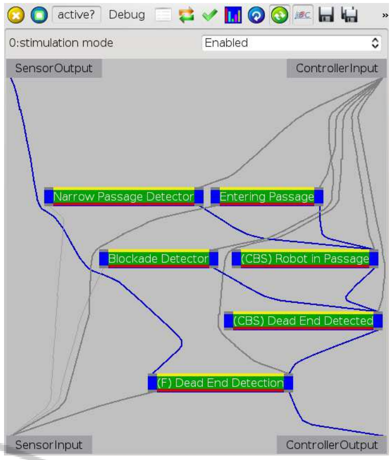
(a) RAVON is within the narrow passage.



(b) RAVON has reached the blockade at the end of the passage.



(c) The behaviours *Entering Passage* and *Narrow Passage* are fully active. Hence, the CBS *Robot in Passage* also is. The fusion behaviour (marked with "(F)") determines this networks activity and target rating.



(d) The behaviour *Blockade Detector* is now also active, resulting in a high activity of the CBS *Dead End Detected*. The group's fusion behaviour is also active, signalling external components that the robot has driven into a dead end.

Figure 7: The behaviour-based network *Dead End Detection* recognises that RAVON has driven into a dead end. Figures (a) and (b) depict two situations that occurred when the robot drove into the dead end, while Figures (c) and (d) illustrate the state of the network in these two situations. The names of the behaviours differ slightly from the ones used in the work at hand.

in mind yields:

$$a_{\mathrm{CBS\,RP}}(t) = 1 \Leftrightarrow \left(\mathrm{ic}_{\mathrm{EP}}(t) = 1\right) \wedge \left(\mathrm{ic}_{\mathrm{NP}}(t) = 1\right)$$
$$\Leftrightarrow \left(\exists t_0 \leq t : a_{\mathrm{EP}}(t_0) = 1\right) \wedge$$
$$\left(\forall t_1 : t_0 \leq t_1 \leq t : a_{\mathrm{NP}}(t_1) = 1\right) \quad (13)$$

A similar equation can be set up for *Dead End*:

$$a_{\mathrm{CBS\,DE}}(t) = 1 \Leftrightarrow \left(\mathrm{ic}_{\mathrm{B}}(t) = 1\right) \wedge \left(\mathrm{ic}_{\mathrm{RP}}(t) = 1\right)$$
$$\Leftrightarrow \left(\exists t_0 \leq t : a_{\mathrm{B}}(t_0) = 1\right) \wedge$$
$$\left(\forall t_1 : t_0 \leq t_1 \leq t : a_{\mathrm{CBS\,RP}}(t_1) = 1\right) \quad (14)$$

Using Equation 13 to replace $a_{\mathrm{CBS\,RP}}$ in Equation 14 yields:

$$a_{\mathrm{CBS\,DE}}(t) = 1 \Leftrightarrow \left(\exists t_0 \leq t : a_{\mathrm{B}}(t_0) = 1\right)$$
$$\wedge \Big(\forall t_1 : t_0 \leq t_1 \leq t : \left(\exists t_2 \leq t_1 : a_{\mathrm{EP}}(t_2) = 1\right.$$
$$\left.\wedge \forall t_3 : t_2 \leq t_3 \leq t_1 : a_{\mathrm{NP}}(t_3) = 1\right)\Big) \quad (15)$$

These considerations show that even rather simple behaviour-based networks containing CBSes can realise functionalities that would require long, hard to read terms if expressed in a mathematical way. By contrast, diagrams depicting iB2C networks can be understood fast by anyone who knows some basic symbols for the behaviour components and their interaction methods. Additionally, special tools for analysing behaviour networks can provide significant support during the implementation and testing phases of a new component. Several such tools exist for the iB2C. One of them, the MCABrowser, is introduced below.

Figure 7a shows RAVON in a simulated off-road environment. It stands at the entry of a narrow passage that is blocked at its end by a large rock, making it a dead end. The robot is commanded to drive towards a target that is situated ahead of it, but outside of the passage. Hence, it drives deeper into the passage until it detects the blockade (see Figure 7b).

Figures 7c and 7d are screenshots of the MCABrowser in iB2C mode, a tool used to analyse and configure MCA2 programs. The iB2C mode features special support for behavioural networks. Every blue rectangle depicts a behaviour. The yellow, green, and red bars visualise a behaviour's activation, activity, and target rating, respectively. With this visualisation, a user can very fast get an insight into the state of a behaviour-based system. Figure 7c shows that when the robot has just entered the passage, the behaviours *Entering Passage* and *Narrow Passage* are fully active. In compliance with Equation 13, the CBS *Robot in Passage* is also active and thus indicates that the robot is situated in a passage belonging to the passage entry through which it has just driven. When RAVON has driven deep enough into the passage, the blockade gets into its sensor range (see Figure 7b). The *Blockade Detector*'s activity rises to 1 and so does the one of *Dead End Detected* (see Figure 7d), which conforms to Equation 14.

5 CONCLUSIONS AND OUTLOOK

The paper at hand presented a concept for using behaviour-based approaches to realise complex tasks. This allows for benefiting from the advantages of such approaches on high navigation layers. The authors proposed the creation of behaviour activity sequences, which was illustrated using the example of the behaviour-based architecture iB2C. It was explained how such sequences can be built up using a generic coordination behaviour. The proposed approach allows for using analysis tools for behaviour-based systems to examine networks consisting of activity sequences and supports the reuse of single elements. Two examples were given to show how behaviour activity sequences can be used to generate robot motions or recognise complex situations in off-road environments.

In the future, the authors plan to use the coordinating behaviour for generating more complex manoeuvres and assessing other types of situations which a robot can encounter in typical off-road environments. Experiments with the real robot shall confirm the results obtained in simulation. Furthermore, an interaction between this layer of the robot's navigation and the high-level navigation shall be established so that the lower of the two layers can give feedback to the higher one if the robot finds itself within a complex situation. Thereupon the high-level navigation can alter its strategy and trigger the execution of complex manoeuvres in the layer below. For example, the information about the presence of a dead end could be sent to the higher layer, where it would be stored in a map and where an alternative route to the target would be calculated.

REFERENCES

Arkin, R. (1998). *Behaviour-Based Robotics*. MIT Press.

Armbrust, C., Braun, T., Föhst, T., Proetzsch, M., Renner, A., Schäfer, B.-H., and Berns, K. (2010). RAVON – the robust autonomous vehicle for off-road navigation. In Baudoin, Y. and Habib, M. K., editors, *Using robots in hazardous environments: Landmine detection, demining and other applications*, chapter RAVON – The Robust Autonomous Vehicle for Off-road Navigation. Woodhead Publishing Limited.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE J. Robotics and Automation*, RA-2(1):14–23.

Gat, E. (1998). Three-layer architectures. In Kortenkamp, D., Bonasso, R., and Murphy, R., editors, *Artificial Intelligence and Mobile Robots*, pages 195–210. AAAI Press / The MIT Press.

Giesbrecht, J. (2004). Global path planning for unmanned ground vehicles. Technical report, Defence Research and Development Canada, Suffield, Canada. Technical Memorandum DRDC Suffield TM 2004-272.

Habib, M. K. (1999). Can planning and reactive systems realize an autonomous navigation? In *Proc. 30th Int. Symposium on Robotics (ISR)*, Tokyo, Japan.

Maes, P. (1990). Situated agents can have goals. *Designing autonomous agents: Theory and practice from biology to engineering and back*, pages 49–70.

Matarić, M. J. (1997). Behavior-based control: Examples from navigation, learning, and group behavior. *J. Experimental and Theoretical Artificial Intelligence - Special Issue on Software Architectures for Physical Agents*, 9(2–3):323–336.

Nicolescu, M. and Matarić, M. (2002). A hierarchical architecture for behavior-based robots. In *Proc. First Int. Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 227–233, Bologna, Italy.

Pirjanian, P. (1999). Behaviour coordination mechanisms — state-of-the-art. Technical Report IRIS-99-375, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California.

Proetzsch, M. (2010). *Development Process for Complex Behavior-Based Robot Control Systems*. RRLab Dissertations. Verlag Dr. Hut.

Ranganathan, A. and Koenig, S. (2003). A reactive robot architecture with planning on demand. In *Proc. IEEE/RSJ IROS 2003*, pages 1462–1468, Las Vegas, Nevada, USA.

Saffiotti, A. (1997). The uses of fuzzy logic in autonomous robot navigation. *J. Soft Computing*, 1(4):180–197.