Verification of Behavior-Based Networks - Using Satisfiability Modulo Theories

M. Sc. Thorsten Ropertz, Robotics Research Lab TU Kaiserslautern, ropertz@cs.uni-kl.de, P.O.Box 3049, D-67653 Kaiserslautern, Germany
Prof. Dr. Karsten Berns, Robotics Research Lab TU Kaiserslautern, berns@cs.uni-kl.de, Germany

Abstract

This paper describes a new approach for verifying complex behavior-based systems (BBS). BBS consist of rather simple interconnected behaviors. The strength of behavior-based control systems lies in their robustness against environmental changes which is realized by the sophisticated interaction between the numerous behaviors. But with a growing amount of behaviors it is no longer trivial to ensure the correctness of this interaction. As incorrect connections between behaviors are one main cause of severe failures, the network structure validity should be guaranteed. The presented approach using the Satisfiability Modulo Theories allows for an automated system modelling that enables an efficient and fine grained network analysis for ensuring safety and reliability.

1 Introduction

Today, many hard, dangerous, and monotonic tasks are delegated to robots in order to support humans. The rising task complexity requires a sophisticated architecture that supports the functional requirements as well as the non-functional ones. One promising architecture class is the class of behavior-based architectures [1]. They have been shown to be robust against environmental changes and can be easily extended due to their modular and distributed nature. This robustness is achieved by a sophisticated interaction between rather simple controllers called behaviors. With a rising task complexity also the control system complexity grows such that it is no longer trivial to check the correctness of the network structure. In many areas of software engineering, testing is an adequate tool for ensuring the required level of quality. But in robotics, safety and reliability often has to be ensured in all possible situations in order to prevent harm to goods and people. One solution to this problem is to apply formal verification, which allows for an extensive examination of the robot and its control system. In contrast to testing, formal verification allows for guaranteeing properties under all circumstances. Unfortunately, the computational complexity usually prevents its application to larger projects. A common technique to keep the verification computationally feasible is to check a simplified model derived from the actual system. Thereby, the quality of the verification results strongly depends on the chosen modelling technique. The model should abstract from unimportant implementation details and focus on substantial property relevant details.

This paper describes the modelling and verification of behavior-based networks using the Satisfiability Modulo Theories (SMT). Thereby, the behavior interaction is modelled using the mathematical relationships given by the architecture specification. Due to the close relation between the model and the architecture specification, the modelling technique is able to represent all structural elements and allows for an extensive investigation of the control system structure.

The remainder of this paper is structured as follows: Section 2 provides an overview of different approaches for verifying software systems using model checking. In Section 3 the prerequisites and the modelling approach itself are presented as well as the verification procedure. Afterwards, the approach is demonstrated using a real world example (see Section 4). Finally, the advantages and disadvantages are summarized in Section 5 and an outlook on future work is given.

2 Related Work

Model checking is a common technique for formally verifying system properties against their specification. Many authors have applied model checking to various problems in the area of robotics. The authors of [2] present a technique for verifying the control system of an unmanned aircraft using model checking. Formal analysis techniques can also be used to optimize systems as shown in [4], where model checking is applied to a learning control system. In [3] model checking is applied to verify a distributed coordination algorithm for robot swarms. Thereby, the algorithm is modelled using the Protocol Meta Language (PROMELA).

Model checking can be further categorized according to the underlying model type. In [5] X-machines are purposed for model checking multi-agent systems. Thereby, an X-machine is a computational machine resembling an FSM extended by memory. Hybrid automata have been used in [6] to verify the European Train Control System. The train control system is modelled using finite state automata and the train and its environment is modelled using differential equations. The model is implemented using differential dynamic logic [7] and verified using the
3 Behavior Network Modelling and Verification

In this section the behavior network modelling and verification is described. First, the actual architecture to be verified is presented in Section 3.1. Afterwards, the Satisfiability Modulo Theories are introduced in Section 3.2. Section 3.3 describes the actual behavior network modelling approach. Finally, the verification procedure is presented in Section 3.4.

3.1 The Behavior-Based Architecture IB2C

The modelling approach presented in the work at hand is based on the behavior-based architecture iB2C (see [11]). An iB2C system consists of interconnected entities called behaviors (Figure 1). Behaviors have a common interface consisting of so called behavior signals and a specialized interface for transmitting control and sensor data. Each behavior can be stimulated by another behavior and inhibited by up to \( k \) other behaviors. The combination of the stimulation \( s \in [0,1] \) and inhibition \( i = \max(i) \in [0,1] \) forms the activation \( a = s \times (1 - i) \in [0,1] \) which represents the intended relevance of the behavior. The amount of influence of a behavior in the current system state is reflected by its activity \( a \in [0,1] \), which is limited by the activation \( a \leq i \). Additionally, each behavior provides a Target Rating \( r \in [0,1] \) which serves as an indicator for the contentment with the current system state. The specialized interface consists of the Input vector \( \vec{c} \) and the Output vector \( \vec{u} \) containing arbitrary typed sensor and control data.

In addition to the generic behaviors, there are special fusion behavior types for coordinating behaviors and their outputs respectively. The Maximum Fusion Behavior forwards the outputs of the connected behavior that has the highest activity, i.e. \( a = i \times \arg\max_c(a_c) \), \( \vec{u} = \vec{u}_s \), and \( r = r_s \), where \( s = \arg\max_c(a_c) \). The Weighted Average Fusion Behavior weights the control values of the competing behaviors with their activities while the Weighted Sum Fusion Behavior sums up the control values according to their activity. For more details see [11]. The coordination strategies are based on the activity such that the selection for the overall network output is primarily influenced by the common interface. I.e. the relevant network structure is given by the common interfaces and their connections. Therefore, the modeling approach presented in the work at hand focuses on the common interface and abstracts from the specialized interface.

3.2 Satisfiability Modulo Theories

Satisfiability Modulo Theories (SMT) are generalizations of a boolean SAT instance in which sets of variables are replaced by predicates of the underlying theories. They deal with methods for checking the satisfiability of first-order formulæ with respect to combinations of background theories like the theory of real numbers and integers. The SMT-LIB [12] initiative has defined a standard for descriptions of background theories used in SMT systems. This initiative was created for providing a common standard and a library of benchmarks to facilitate the evaluation and the comparison of SMT systems. The developed standard comprises several theories like integers, reals, bit-vectors and arrays as well as various logics for processing the theories. The combination of these different theories allows for modelling complex systems like behavior-based networks.

The generated SMT model basically consists of symbol declarations and assertions:

\[
\text{(declare-fun name (arguments) type)}
\]

\[
\text{(assert name term)}
\]

Thereby, the symbol has a name, zero or more arguments, and a type (e.g. bool,int,real). The range of possible interpretations (values) of a symbol can be restricted using assertions. Symbols can also be defined to have a fixed value and thus represent functions with 0 or more arguments like a maximum function:

\[
\text{(define-fun f_max ((a Real) (b Real)) Real (ite (> a b) a b))}
\]

Thereby, the term \( \text{ite} (> a \ b) \ a \ b \) completely defines the symbol.

Two famous theorem provers supporting the SMT-LIB standard are CVC4 (see [see http://cvc4.cs.nyu.edu/web/]) and Z3 (see [see http://z3.codeplex.com/]). Most common SMT solvers only support decidable theories like the first order theory of the natural numbers with addition. The behavior signals are defined using a non-linear arithmetic (compare Section 3.1) which is in general undecidable. Fortunately, the required formulæ are instances of the first order theory of the reals which
allows for quantifier elimination and thus is decidable according to Tarski (1951).

### 3.3 Modelling of Behavior Networks Using Satisfiability Modulo Theories

In this section, the modelling technique using SMT is presented. The complete modelling procedure is illustrated using the example shown in Figure 2.

**Figure 2:** Simple behavior network consisting of 3 generic behaviors and 1 fusion behavior. Behavior A stimulates behavior B (blue line) and inhibits behavior C (red line). Control values of all three behaviors are combined by a maximum fusion behavior FB.

The example consists of 3 behaviors that are combined using the maximum fusion behavior. Thereby, A stimulates B as illustrated by the blue line and inhibits C as illustrated by the red line. The modelling process consists of 2 stages. In the first stage (declaration stage), the network is examined and the representation of the behavior signals for each behavior are declared. The declaration has the form

\[
\begin{align*}
&\text{(declare-fun name_i () Real)} \\
&\text{(declare-fun name_stimulation () Real)} \\
&\text{(declare-fun name_target_rating () Real)} \\
&\text{(declare-fun name_activity () Real)} \\
&\text{(declare-fun name_activation () Real)} \\
&\text{(declare-fun name_i () Real)} \\
\end{align*}
\]

, where name is replaced by the corresponding behavior name. Each behavior signal is represented by a separate variable symbol in order to allow for later property specification based on the behavior state. In addition to the declarations, also basic assertions limiting the value range of the declared variables to \([0,1]\) as well as the activity limitation are defined according to the architecture specification.

\[
\begin{align*}
&\text{(assert (> name_i 0.0))} \\
&\text{(assert (< name_i 1.0))} \\
&\text{(assert (> name_stimulation 0.0))} \\
&\text{(assert (< name_stimulation 1.0))} \\
&\text{(assert (> name_target_rating 0.0))} \\
&\text{(assert (< name_target_rating 1.0))} \\
&\text{(assert (< name_activity_name_activation))} \\
&\text{(assert (= name_activation}
\]

\[
\text{(* name_stimulation (- 1.0 name_i)))}
\]

The internal behavior functionality shall not be modelled in order to keep the model computationally feasible. Hence, the specialized interface is not modelled at all. One option to model the internal activity computation is to assume the behavior to have an activity equal to its activation. Regarding precedences, this representation could give wrong results. For example, Figure 3 shows a behavior network consisting of 3 generic behaviors, where behavior A stimulates B, which in turn inhibits C. If the activity would always be equal to the activation, C would always be inhibited if A is active. But, according to the iB2C specification, the activation only provides and upper bound for the activity. Thus, if A’s activity is 0.8 then C’s inhibition can be much lower than 0.8 since B’s activity could be 0.0 in spite of its strong stimulation. Hence, A has no precedence over C while the model would approve the precedence. The presented approach indirectly models the internal behavior dependent activity computation by just restricting the activity value to the range \([0,\text{name_activation}]\) as defined in the architecture specification. Considering the behavior network shown in Figure 3, the model will not approve the precedence and thus provide a correct result.

**Figure 3:** Behavior network consisting of 3 generic behaviors. Behavior A stimulates behavior B (blue line) which in turn inhibits behavior C (red line).

In the second stage (assertion stage), the behavior interconnections are modelled. A’s stimulation is not connected and it is not set to be stimulated by default. Therefore, its stimulation input is set to zero as well as its inhibition input.

\[
\begin{align*}
&\text{(assert (= A_stimulation 0))} \\
&\text{(assert (= A_i 0.0))} \\
\end{align*}
\]

B’s stimulation input is connected to A’s activity port. Thus, B’s stimulation input always equals A’s activity value. The inhibition input is not connected at all and therefore set to zero.

\[
\begin{align*}
&\text{(assert (= B_stimulation A_activity)} \\
&\text{(assert (= B_i 0.0))} \\
\end{align*}
\]

C’s stimulation input is not connected and therefore set to zero, while its inhibition input is connected to A’s activity output.

\[
\begin{align*}
&\text{(assert (= C_stimulation 0.0)} \\
&\text{(assert (= C_i Behavior_A_activity))} \\
\end{align*}
\]

The described modelling approach for generic behaviors abstracts from the internal functionality and focuses on the interface and the network structure. Regarding the three standard fusion behaviors, the modelling approach can be refined to gain a more detailed model. In the following, the maximum fusion (FB) is exemplary modelled.

The first stage is processed as described above. Additionally, the fusion inputs are declared.

\[
\begin{align*}
&\text{(declare-fun FB_i () Real)} \\
&\text{(declare-fun FB_stimulation () Real)} \\
&\text{(declare-fun FB_activity () Real)} \\
&\text{(declare-fun FB_target_rating () Real)} \\
&\text{(assert (> FB_i 0.0))} \\
\end{align*}
\]

The internal behavior functionality shall not be modelled in order to keep the model computationally feasible. Hence, the specialized interface is not modelled at all.
The internal computations require the maximum function which is not integrated in the SMT base theory. Therefore, the maximum function is generated for the corresponding amount of compared values (three in this case). Afterwards, the actual internal computation is modelled.

\[
\text{(define-fun } f_{\text{max}}(a) \text{ (} b \text{ ) (} c \text{ ) Real)} \text{ Real } (\text{ite } (> a b) \text{ ) } (\text{ite } (> a c) \text{ ) } (\text{ite } (> b c) \text{ ) })
\]

\[
\text{(define-fun } f_{\text{indirect}_{\text{max}}} \text{ (} a_0 \text{ ) Real)} \text{ Real } (\text{ite } (> a_0 b_0) \text{ ) } (\text{ite } (> a_0 c_0) \text{ ) } (\text{ite } (> b_0 c_0) \text{ ) } (\text{ite } (> b_0 c_1) \text{ ) })
\]

\[
\text{(assert } (= \text{FB}_{\text{activity}} \text{ (} \times \text{FB}_{\text{activation}} \text{ )} \text{ } \text{FB}_{\text{activity}} \text{ ) Real)}
\]

\[
\text{(assert } (= \text{FB}_{\text{target}_{\text{rating}}} \text{ (} \text{FB}_{\text{activity}} \times \text{FB}_{\text{activation}} \text{ )} \text{ ) Real)}
\]

In the second stage, the variables have to be connected according to the network structure as described for generic behaviors. The complete modelling of fusion behaviors with respect to the common interface allows for a more fine grained analysis of the activity propagation while the modelling procedure complexity is only slightly increased.

### 3.4 Verification

After the automatic system modelling, the system can be verified using standard SMT-LIB capable solvers like the Z3 solver. An important requirement regarding the solver is the support of non-linear arithmetic since the activation is defined by a non-linear real numbered term. It is possible to check different network properties by adding assertions describing the regarded property. In contrast to the work described in [9], the system state within this model is not boolean, which makes it harder to formulate appropriate properties. E.g. it is not possible to check whether one behavior can be active while another one is not active due to the missing preciseness of the word “active”. The precise modelling allows for more precise conclusions but also requires precise property specifications. Thus, properties should be defined at least based on the ordinal scale, e.g. “B’s activity can be higher than A’s activity”. The corresponding SMT specification could look as follows:

\[
\text{(assert } (> \text{B}_{\text{activity}} \text{ A}_{\text{activity}})\text{ )}
\]

This property asks for the existence of a state in which B’s activity is higher than A’s activity and can therefore be directly checked using the sat solver. The property is fulfilled if the complete system model together with the assertion is satisfiable. Properties that specify invariants like “B’s activity is always higher than A’s activity”, have to be transformed in order to be checked using a sat solver. Sat solvers look for a system state that satisfies the specification and abort the search after a satisfying state has been found. Thus, in order to examine the full state space, invariants have to be negated:

\[
\text{(assert } (\not(> \text{B}_{\text{activity}} \text{ A}_{\text{activity}}))\text{ )}
\]

and the sat solver return value has to be reinterpreted, i.e. the property is fulfilled if the model is not satisfiable.

In behavior networks precedences are very important properties. Precedences are usually implemented by creating inhibition links between the affected behaviors. For example, let behavior A have a higher precedence than behavior C as shown in Figure 2. Thus, A inhibits C using its activity. Let A have an activity greater than 0.5. Then, C’s activity will be less than 0.5 and thus less than A’s activity, independent of its other inputs according to the definition. Hence, a high priority behavior cannot be overruled in the activity range of [0.5, 1]. If A’s activity is less than or equal to 0.5, then A can be overruled but C has to be stimulated in a much higher manner:

\[
a_A < a_C
\]

\[
\Leftrightarrow a_A < s_C \ast (1 - i_C)
\]

\[
\Leftrightarrow a_A < s_C \ast (1 - a_A)
\]

\[
\Leftrightarrow a_A < \frac{a_A}{1 - a_A}
\]

Therefore, A would win through even if A and B would have comparable activities without the inhibition link. The precedence rule can be verified by checking whether

\[
a_C > a_A \Rightarrow s_C > \frac{a_A}{1 - a_A}
\]

holds.

The higher complexity of the required property specification seems to be a weakness of this approach. But in fact, the quantitative interaction between different behaviors is of importance and can be addressed using the approach described in the work at hand. E.g. regarding the example illustrated in Figure 2 let B have a high activity of 0.9. B seems to be active, but the network output at FB could also be the determined by A since A can have an activity equal to B. Thus, the overall network output is strongly affected by the activity differences.

### 4 Example

The described approach is further investigated using a part of the behavior-based navigation system used to control the off-road robot RAYON [13] as described in [9]. The behavior network called (G) Drive Control (see Figure 4) coordinates three different navigation approaches:

1. A high-level navigation component that provides target coordinates via the behaviors (F) Nav. Direct Point Access Interface (DPA_IF) and (F) Nav. Point Access with Orientation Interface (PAO_IF),
2. A classic A*-based local path planner (G) **Local Path Planner (LPP)**.

3. A specialized local path planner based on so called "passages".

**Figure 4:** Behavior network coordinating different navigation approaches (gray: generic behavior; blue: maximum fusion; double-bordered gray: behavioral group)

Thereby, the local path planners are encapsulated by the (G) **Mediator** (see Figure 5) to group, which allows for a better manual inspection of the network. The specialized path planner is based on passages, which are defined as paths leading through obstacle formations (see [14]).

**Figure 5:** Unfolded Mediator group encapsulating the passage based path planner and a classic A*-based path planner (gray: generic behavior; blue: maximum fusion; double-bordered gray: behavioral group)

Behavior **New Passage (NP)** gets active if a new passage was detected in the environment and forwards its coordinates to the **Passage Manager (PM)**, which in turn decides whether the robot should enter it. **Same Passage (SP)** is intended to be active if a previously detected passage is seen again to guide the robot through a passage. The coordinates provided by PM and SP are combined by the maximum fusion behavior (F) **Passage Driver Target (PDT)**. The result is transmitted to the **Passage Driver (PD)**, which processes the coordinates and published them to the rest of the network.

The lower level control network requires the navigator to provide at most one set of target coordinates at a time, i.e. only one navigation component is allowed to send its coordinates. Therefore, the navigator shall ensure an order of precedence according to the following specifications (compare [9]):

1. The classic local path planner has precedence over the high-level navigation;
2. A new passage is preferred to previously detected ones;
3. The passage based planner has precedence over the classic local path planner;

To verify the first property, the behaviors LPP, DPA_IF, and POA_IF have to be regarded. According to the precedence implementation described in section 3.4, the classic local path planner has precedence over the high-level navigation if

\[ a_{DPA_IF} > a_{LPP} \Rightarrow s_{DPA_IF} > \frac{a_{LPP}}{1 - a_{LPP}} \]  

(2)

and

\[ a_{POA_IF} > a_{LPP} \Rightarrow s_{POA_IF} > \frac{a_{LPP}}{1 - a_{LPP}} \]  

(3)

holds. Since the validity of the formulae combined with the system model is required, the formulae have to be negated and the sat solver result reinterpreted. The following shows the SMT formula for the first part of the property exemplary:

```smt
(assert (not (=> (> DPA_IF_activity LPP_activity) (> DPA_IF_stimulation (/ LPP_activity (- 1 LPP_activity)))))
```

The sat solver result is *unsat* as expected. Thus, the classic local path planner has precedence over the high level direct point access interface. The second part of the property is checked analogously and yields the expected result. Hence, the system model fulfills the property. This property shows the strength of the formal verification approach in contrast to the manual inspection. The inhibition link realising the precedence rule does not directly connect the ordered behaviors. Instead, the inhibition link origins in the associated fusion module. Additionally, the link crosses group boundaries and is
therefore hard to trace manually especially in large networks containing several abstraction hierarchies. The described approach automatically checks the required property while abstracting from the actual realization.

2: The second property states that NP has precedence over SP. The corresponding SMT representation can be developed analogously to the first property:

\[ \text{assert } (\text{not } (\Rightarrow (\Rightarrow \text{SP_activity} \text{NP_activity}) \Rightarrow \text{SP_stimulation} (\text{NP_activity} (\text{NP_activity} 1)))) \]

The sat solver result is \textit{unsat} as expected.

3: Finally, the third property is verified using the SMT formula:

\[ \text{assert } (\text{not } (\Rightarrow (\Rightarrow \text{LPP_activity} \text{PD_activity}) \Rightarrow \text{LPP_stimulation} (/ \text{PD_activity} (\text{PD_activity} 1)))) \]

The sat solver result is \textit{unsat} as expected.

To sum up, the precedences rules can be verified by adding the rather simple formulae to the system model and checking for satisfiability. Thereby, the properties to be verified can be categorized and corresponding patterns specified that allow for a simple property specification and verification.

5 Conclusion and Future Work

In this paper, a new verification approach for behavior-based networks using Satisfiability Modulo Theories has been presented. The automatic modelling of behavior networks has been described and the basic idea and problem of the property specification has been discussed. The described modelling approach reflects the mathematical interrelationships and is close to the specification such that it allows for a fine-grained verification of the network structure while abstracting from internal implementation details of the individual behaviors. Thus, an efficient identification of wrong interconnections is supported even in large networks while preserving important details concerning the actual behavior states. The fine-grained modelling requires a precise specification of the properties to be verified. Therefore, tools that support the property specification have to be developed in future works. Additionally, the complete abstraction of the computations within the individual behaviors leads to an over-approximation of the actual system behavior. The presented model does neither include a representation of time nor a representation of the system dynamics, but only the discrete combinatorial network structure. Therefore, no temporal logic can be used to verify the system, i.e. it is not possible to check time varying properties. For example, it is not possible to check whether a certain state will eventually be reached or whether the robot will stop 5 seconds after an obstacle was detected. To overcome these problems, behavior internal computations have to be modelled. These internal computations often depend on sensor values such that a more detailed modelling also requires the modelling of parts of the environment. Environmental conditions are usually described using differential equations which cannot be represented using SMT formulae. Thus, further investigations will have to deal with hybrid automata for modelling the discrete control system in combination with the dynamics of the robot and its environment.

References


