On the Semantics of Temporal Specifications of Component-Behavior for Dynamic Architectures

Diego Marmsoler
Technical University of Munich
Email: diego.marmsoler@tum.de

***Abstract*—In component-based design, temporal logic is a means to specify the temporal behavior of components. If these components are deployed to a dynamic architecture, they can be activated and deactivated over time. Thus, the traditional semantics of temporal specifications of component-behavior does no longer reflect the actual behavior of the components within such dynamic architectures. To address this problem, we provide an alternative semantics of temporal specifications of component-behavior for dynamic architectures, taking into account component activation and deactivation. We show soundness and relative completeness of our semantics w.r.t. the traditional one. The new semantics can then be used to support in the formal specification of dynamic architectures by separating the specification of component-behavior from other aspects such as component activation and architecture reconfiguration.**

***Index Terms*—Dynamic Architectures, Temporal Logic, Formal Semantics**

I. Introduction

In component-based development, the temporal behavior of components is often specified by means of temporal logic formulæ over the components interface [1, 6, 8]. Consider, for example, a component c_3 with output port o_1 whose behavior is given by a temporal specification $'\bigcirc(o_1 = 8)'$, meaning that it outputs an 8 on its port o_1 at time-point 1 (assuming that time starts at 0).

Architectures are then specified by a set of components and connections between their ports. Consider, for example, another component c_2 with input port i_2. An architecture A can now be specified by a set of components containing c_2 and c_3, and a connection between the ports i_2 of component c_2 and o_1 of component c_3.

For these 'static' architectures, the original specification of temporal properties of single components remain valid for these components when they are deployed to an architecture. The original specification of component c_3, for example, is still valid for architecture A, i.e., c_3 will still output an 8 on its port o_1 at time-point 1, even if deployed to architecture A.

With the emergence of mobile computing, however, dynamic architectures became more and more important [3, 7, 15]. In such architectures, components can appear and disappear and connections can change, both over time. Consider, for example, the execution trace of a dynamic architecture depicted in Fig. 1. The figure shows the first three configurations of one possible execution of a dynamic architecture composed of three components c_1, c_2, and c_3.

For such architectures, the traditional interpretation of temporal specifications of the behavior of components is not valid anymore. For example, it is not clear, whether the trace in Fig. 1 actually fulfills the original specification of component c_3, since c_3 is not active at time-point 1 ($n = 1$).
What is then the correct interpretation of temporal specifications of a component's behavior when the component can be activated and deactivated over time?

This is indeed the question we are going to investigate in this paper. To answer it, we follow the approach depicted in Fig. 2: We first introduce the notion of *behavior trace assertions*, a language to specify the temporal behavior of components by means of linear temporal logic [13]. Then, we provide two interpretations for it based on two different semantic domains: *behavior traces* and *configuration traces*. The former reflecting the traditional interpretation of temporal specifications for component-behavior, while the latter reflects an alternative interpretation of temporal specifications of component-behavior for dynamic architectures. Moreover, we provide a mapping Π from the domain of configuration traces to the domain of behavior traces to extract the behavior of a single component out of a configuration trace. Finally, we show that the diagram in Fig. 2 commutes, i.e., that the alternative semantics for dynamic architectures indeed corresponds to the traditional interpretation when projected to a component's behavior:

$$\Pi_c(t)\ {}^{t}_{b}\models \gamma \iff t\ {}^{t}_{k}\models \gamma,$$

for assertion γ, component c, and configuration trace t.
Thereby, we show *soundness* and *(relative) completeness* of our alternative interpretation for dynamic architectures w.r.t. the traditional interpretation.

Our alternative interpretation allows to interpret temporal specifications of component-behavior over dynamic architectures. Hence, it allows for separating the specification of component-behavior from other aspects, such as component activation and architecture reconfiguration, when specifying dynamic architectures. Thus, our results can be used to support in the specification of dynamic architectures.

$$\begin{tikzcd}
 & \text{Behavior Trace Assertions} \arrow[dl, dash, "{}^{t}_{b}\models"'] \arrow[dr, dash, "{}^{t}_{k}\models"] & \\
\text{Behavior Traces} & \circlearrowright & \text{Configurtion Traces} \arrow[ll, "\Pi"]
\end{tikzcd}$$

Figure 2. Behavior trace assertions with semantic domains.

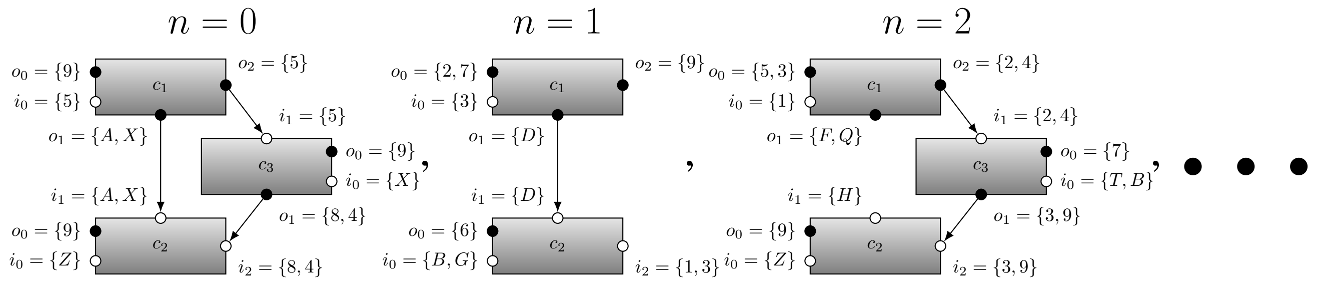

Figure 1. Execution trace of a dynamic architecture.

II. A Model of dynamic Architectures

In [14] we introduce a model for dynamic architectures based on the notion of configuration traces. Our model is based on Broy's Focus theory [4] and an adaptation of its dynamic extension [5]. In this section, we briefly summarize the main concepts of the model and extend it by the notion of behavior traces to model the behavior of single components.

A. Foundations: Ports, Valuations, and Components

In our model, components communicate by exchanging *messages* over *ports*. Thus, we assume the existence of sets M and P containing all messages and ports, respectively.

1) Port-valuations: Ports can be valuated by messages. Roughly speaking, a valuation for a set of ports is an assignment of messages to each port.

Definition 1 (Port-valuation): For a set of ports $P \subseteq \mathsf{P}$, we denote by $\overline{P}$ the set of all possible *port-valuations (PVs)*, formally:

$$\overline{P} \overset{\text{def}}{=} (P \to \wp(\mathsf{M})) \ .$$

Moreover, we denote by $[p_1, p_2, \ldots \mapsto \{m_1\}, \{m_2\}, \ldots]$ the valuation of ports $p_1, p_2, \ldots$ with sets $\{m_1\}, \{m_2\}, \ldots$, respectively. For singleton sets we shall sometimes omit the set parentheses and simply write $[p_1, p_2, \ldots \mapsto m_1, m_2, \ldots]$. Note that in our model, ports can be valuated by a *set* of messages, meaning that a component can send/receive no message, a single message, or multiple messages at each point in time.

2) Components: In our model, the basic unit of computation is a component. It consists of an identifier and a set of input and output ports. Thus, we assume the existence of set C_{id} containing all component identifiers.

Definition 2 (Component): A *component* is a triple (id, I, O) consisting of:

- a component identifier $id \in \mathsf{C}_{id}$ and
- two *disjoint* sets of input and output ports $I, O \subseteq \mathsf{P}$.

The set of all components is denoted by $\mathcal{C}$. For a set of components $C \subseteq \mathcal{C}$ we denote by:

- $\mathsf{in}(C) \overset{\text{def}}{=} \bigcup_{(id,I,O) \in C}(\{id\} \times I)$ the set of *component input ports*,
- $\mathsf{out}(C) \overset{\text{def}}{=} \bigcup_{(id,I,O) \in C}(\{id\} \times O)$ the set of *component output ports*,
- $\mathsf{port}(C) \overset{\text{def}}{=} \mathsf{in}(C) \cup \mathsf{out}(C)$ the set of all *component ports*, and
- $\mathsf{id}(C) \overset{\text{def}}{=} \bigcup_{(id,I,O) \in C}\{id\}$ the set of all *component identifiers*.

A set of components $C \subseteq \mathcal{C}$ is called *healthy* if a component is uniquely determined by its name:

$$\begin{aligned} &\forall (id, I, O), (id', I', O') \in C\colon \\ &\quad id = id' \implies I = I' \wedge O = O' \ . \end{aligned} \tag{1}$$

For a set of component-ports $P \subseteq \mathsf{C}_{id} \times \mathsf{P}$, we denote by $\overline{P}$ the set of all possible *component-port-valuations (CPVs)*, formally:

$$\overline{P} \overset{\text{def}}{=} (P \to \wp(\mathsf{M})) \ .$$

B. Modeling Component-Behavior

A component's behavior is modeled by a set of execution traces over the component's interface.

Definition 3 (Behavior trace): A behavior trace (BT) for a component (id, I, O) is a sequence $(\overline{I \times O})^\infty$. The set of all BTs for component c is denoted by $\mathcal{B}(c)$.

Note that a component's behavior is actually modeled as a set of behavior traces, rather than just a single trace. This is to handle non-determinism for inputs to, as well as outputs from components.

Example 1 (Behavior trace): In the following, we provide a possible BT for a component c_3 with two input ports i_0 and i_1, and two output ports o_0 and o_1: $[i_0, i_1, o_0, o_1 \mapsto X, 5, 9, \{8, 4\}]$, $[i_0, i_1, o_0, o_1 \mapsto \{T, B\}, \{2, 4\}, 7, \{3, 9\}], \cdots$.

C. Modeling Dynamic Architectures

Dynamic architectures are modeled as sets of *configuration traces* which are sequences over *architecture configurations.*

1) Architecture configurations: In our model, an architecture configuration *connects* ports of *active* components.

Definition 4 (Architecture configuration): An architecture configuration (AC) over a *healthy* set of components $C \subseteq \mathcal{C}$ is a triple (C', N, μ), consisting of:

- a set of active components $C' \subseteq C$,
- a connection $N\colon \mathsf{in}(C') \to \wp(\mathsf{out}(C'))$, and
- a CPV $\mu \in \overline{\mathsf{port}(C')}$.

We require connected ports to be consistent in their valuation, that is, if a component provides messages at its output port, these messages are transferred to the corresponding, connected input ports:

$$\begin{aligned} &\forall p_i \in \mathsf{in}(C')\colon N(p_i) \neq \emptyset \\ &\qquad \implies \mu(p_i) = \bigcup_{p_o \in N(p_i)} \mu(p_o) \ . \end{aligned} \tag{2}$$

The set of all possible ACs over a healthy set of components $C \subseteq \mathcal{C}$ is denoted by $\mathcal{K}(C)$.

Note that connection N is modeled as a set-valued, partial function from component input ports to component output ports, meaning that: (i) input/output ports can be connected to several output/input ports, respectively, and (ii) not every input/output port needs to be connected to an output/input port, respectively. Moreover, Eq. (2) requires all connected ports to be consistent in their valuation.

2) Configuration traces: A configuration trace consists of a series of configuration snapshots of an architecture during system execution. Note that in the following we denote with $(E)^\infty$ the set of all *infinite* sequences over the elements of a set E. Moreover, we denote with $[C]^i = c_i$ (with $1 \leq i \leq n$) the projection to the i-th component of an n-tuple $C = (c_1, \ldots, c_n)$.

Definition 5 (Configuration trace): A configuration trace (CT) over a healthy set of components $C \subseteq \mathcal{C}$ is a sequence $(\mathcal{K}(C))^\infty$. The set of all CTs over C is denoted by $\mathcal{R}(C)$.

A CT $t \in \mathcal{R}(C)$ is called *fair* w.r.t. component $c \in C$ iff the component is guaranteed to be activated infinitely many times:

$$\mathit{fair}(c,t) \overset{\text{def}}{\iff} \forall n \in \mathbb{N}\ \exists n' > n\colon c \in [t(n')]^1 \quad . \tag{3}$$

Example 2 (Configuration trace): Figure 1 shows the first three ACs of a possible CT. The first AC, $t(0) = (C', N, \mu)$, e.g., consist of:

- components $C' = \{C_1, C_2, C_3\}$, with
  - $C_1 = (c_1, \{i_0\}, \{o_0, o_1, o_2\})$,
  - $C_2 = (c_2, \{i_0, i_1, i_2\}, \{o_0\})$, and
  - $C_3 = (c_3, \{i_0, i_1\}, \{o_0, o_1\})$;
- connection N, with $N((c_2, i_1)) = \{(c_1, o_1)\}$, $N((c_3, i_1)) = \{(c_1, o_2)\}$, and $N((c_2, i_2)) = \{(c_3, o_1)\}$; and
- valuation $\mu = [(c_1, i_0), (c_1, o_0), (c_2, i_2), \cdots \mapsto 5, 9, \{8, 4\}, \cdots]$.

Note that a dynamic architecture is modeled as a *set* of CTs rather than just one single trace. Again, this allows for non-determinism in inputs to an architecture as well as its reaction. Moreover, note that our notion of architecture is dynamic in the following sense: (i) *components* may appear and disappear over time and (ii) *connections* may change over time.

D. From Configuration Traces to Behavior Traces

In the following, we introduce the notion of projection to extract the behavior of a certain component out of a given CT. Note that in the following we denote with $(E)^*$ the set of all *finite and infinite* sequences over the elements of a set E. Moreover, for a sequence s, we denote with $s\,|_n$ the subsequence of s up to its n-th element (excluding $s(n)$) and with $s@e$ the sequence resulting from appending element e to s.

Definition 6 (Projection): Given a (finite or infinite) CT $t \in (\mathcal{K}(C))^*$ over a healthy set of components $C \subseteq \mathcal{C}$. The projection to component $c = (id, I, O) \in C$ is denoted by $\Pi_c(t) \in (\mathcal{B}(c))^*$ and defined recursively by the following equations:

$$\begin{aligned}
\Pi_c(t\,|_0) &\overset{\text{def}}{=} \langle\rangle \ , \\
c \in [t(n)]^1 \implies \Pi_c(t\,|_{n+1}) &\overset{\text{def}}{=} \Pi_c(t\,|_n) \\
&\quad @\ \big(\lambda p \in I \cup O\colon\ [t(n)]^3\,(id, p)\big), \\
c \notin [t(n)]^1 \implies \Pi_c(t\,|_{n+1}) &\overset{\text{def}}{=} \Pi_c(t\,|_n) \ .
\end{aligned}$$

Example 3 (Projection): Applying projection of component c_3 to the CT given by Ex. 2 results in a BT starting as described by Ex. 1.

III. Specifying Component-Behavior

In the following we introduce the notion of *behavior trace assertions*, a language to specify component-behavior over a given interface specification. Then, we provide two interpretations thereof: the traditional one over *behavior traces* and an alternative one over *configuration traces*. Finally, we show *soundness* and *(relative) completeness* of our alternative semantics w.r.t. the traditional one.

A. Behavior Trace Assertions

Component-behavior can be specified by means of behavior trace assertions, i.e., temporal logic [13] formulæ over behavior assertions. Behavior assertions, on the other hand, are used to specify a components state at a certain point in time. They are specified over a given interface specification.

1) Interface specifications: Interfaces declare a set of port identifiers and associate a sort with each port. Thus, in the following, we postulate the existence of the set of all port identifiers P_{id}. Moreover, interfaces are specified over a given signature $\Sigma = (S, F, B)$ consisting of a set of sorts S, function symbols F, and predicate symbols B.

Definition 7 (Interface specification): An interface specification (IS) over a signature $\Sigma = (S, F, B)$ is a triple (P_{in}, P_{out}, t^p), consisting of:

- two *disjoint* sets of input and output port identifiers $P_{in}, P_{out} \subseteq \mathsf{P}_{id}$,
- a mapping $t^p\colon P_{in} \cup P_{out} \to S$ assigning a sort to each port identifier.

The set of all interface specifications over signature Σ is denoted by $\mathcal{S}_I(\Sigma)$.

2) Behavior assertions: Behavior assertions specify a component's state (i.e.: valuations of its ports with messages) at a certain point in time. In the following, we are not going into the details of how to specify such assertions, rather, we assume the existence of a set containing all type-compatible behavior assertions over a given interface specification.

Definition 8 (Behavior assertions): Given IS $S_i = (P_{in}, P_{out}, t^p)$ over signature $\Sigma = (S, F, B)$ and family of variables $V = (V_s)_{s \in S}$ with variables V_s for each sort $s \in S$. With $\varphi_\Sigma^V(S_i)$ we denote the set of all type-compatible (w.r.t. t^p) behavior assertions (BAs) for S_i, Σ, and V.

$$
\begin{aligned}
\phi \in \varphi_\Sigma^{V\cup V'}(S_i) &\implies \phi \in \Gamma_\Sigma^{(V,V')}(S_i) \ , \\
\gamma \in \Gamma_\Sigma^{(V,V')}(S_i) &\implies \text{“}\bigcirc\gamma\text{”},\ \text{“}\Diamond\gamma\text{”},\ \text{“}\Box\gamma\text{”} \in \Gamma_\Sigma^{(V,V')}(S_i) \ , \\
\gamma,\gamma' \in \Gamma_\Sigma^{(V,V')}(S_i) &\implies \text{“}(\gamma\ \mathcal{U}\ \gamma')\text{”} \in \Gamma_\Sigma^{(V,V')}(S_i) \ , \\
\gamma \in \Gamma_\Sigma^{(V,V')}(S_i) &\implies \text{“}\forall x\colon X.\ \gamma\text{”} \in \Gamma_\Sigma^{(V,V')}(S_i)\ \wedge \\
&\qquad \text{“}\exists x\colon X.\ \gamma\text{”} \in \Gamma_\Sigma^{(V,V')}(S_i) \\
&\qquad [\text{for } X \in S \text{ and } x \in V'_X] \ .
\end{aligned}
$$

Figure 3. Inductive definition of behavior trace assertions.

a) Algebras and variable assignments: A BA is always interpreted over a given algebra for the signature used in the corresponding IS. Thus, in the following, we denote by $\mathcal{A}(\Sigma)$ the set of all algebras $(S', F', B', \alpha, \beta, \gamma)$ for signature $\Sigma = (S, F, B)$, consisting of sets S', functions F', predicates B', and corresponding interpretations $\alpha\colon S \to S'$, $\beta\colon F \to F'$, and $\gamma\colon B \to B'$. Moreover, with $\mathcal{I}_A^V$ we denote the set of all *variable assignments (VAs)* $\iota = (\iota_s)_{s\in S}$ (with $\iota_s\colon V_s \to \alpha(s)$ for each $s \in S$) for a family of variables $V = (V_s)_{s\in S}$ in an algebra A.

b) Semantics of behavior assertions: In the following we define the semantics of BAs. Thereby we define which CPVs and which ACs satisfy a given BAs. Note that in the following we denote with $A \leftrightarrow B$ the set of bijective functions from set A to set B.

Definition 9 (Behavior assertions: semantics): Given interface specification $S_i = (P_{in}, P_{out}, t^p) \in \mathcal{S}_I(\Sigma)$, a healthy set of components $C \subseteq \mathcal{C}$, component $c = (id, I, O) \in C$, algebra $A \in \mathcal{A}(\Sigma)$, and VA $\iota = (\iota_s)_{s\in S} \in \mathcal{I}_A^V$. We denote with $\mu\ {}_b{\models}^{(\delta^i,\delta^o)}_{(A,\iota)}\gamma$ that $\mu \in \overline{I \cup O}$ satisfies BA $\gamma \in \varphi_\Sigma^V(S_i)$ for port-interpretations (PIs) $\delta^i\colon I \leftrightarrow P_{in}$ and $\delta^o\colon O \leftrightarrow P_{out}$. Moreover, we denote with $k\ {}_k{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota)}\gamma$ that $k \in \mathcal{K}(C)$ satisfies BA $\gamma \in \varphi_\Sigma^V(S_i)$ for δ^i and δ^o, respectively. For BA $\gamma \in \varphi_\Sigma^V(S_i)$, we require the following property for the satisfaction relations:

$$
\frac{\forall p \in I \cup O\colon \mu(p) = [k]^3\,(id, p)}{\mu\ {}_b{\models}^{(\delta^i,\delta^o)}_{(A,\iota)}\gamma \iff k\ {}_k{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota)}\gamma}\ \mu \in \overline{I \cup O}, k \in \mathcal{K}(C) \ . \tag{4}
$$

Note that we do not provide an explicit definition of the corresponding satisfaction relations in this text. Rather, with Eq. (4), we provide a sufficient property which characterizes possible satisfaction relations: Roughly speaking, for a valuation μ of the ports of a component c and an AC k, which assigns the same messages to the ports of component c as μ does, we require that μ fulfills a BA γ whenever k fulfills γ.

3) Behavior trace assertions: Behavior trace assertions are a means to specify a component's behavior in terms of temporal specifications over BAs.

Definition 10 (Behavior trace assertions): For a family of variables $V = (V_s)_{s\in S}$, *rigid*[1] variables $V' = (V'_s)_{s\in S}$, the set of all behavior trace assertions (BTAs) for IS $S_i \in \mathcal{S}_I(\Sigma)$ is given by $\Gamma_\Sigma^{(V,V')}(S_i)$ and defined inductively by the equations provided in Fig. 3.

[1]Variables whose value is fixed for the whole execution.

$$
\begin{aligned}
(t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\phi &\iff \exists \iota \in \mathcal{I}_A^V : t(n)\ {}_b{\models}^{(\delta^i,\delta^o)}_{(A,\iota\cup\iota')}\phi \\
&\qquad [\text{for } \phi \in \varphi_\Sigma^V(S_i)] \ , \\
(t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\text{“}\bigcirc\gamma\text{”} &\iff (t,n+1)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma \ , \\
(t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\text{“}\Diamond\gamma\text{”} &\iff \exists n' \geq n\colon (t,n')\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma \ , \\
(t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\text{“}\Box\gamma\text{”} &\iff \forall n' \geq n\colon (t,n')\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma \ , \\
(t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\text{“}(\gamma\ \mathcal{U}\ \gamma')\text{”} &\iff \exists n' \geq n\colon (t,n')\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma' \ \wedge \\
&\qquad \forall n \leq m < n'\colon (t,m)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma \ , \\
(t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\text{“}\exists x\colon X.\ \gamma\text{”} &\iff \exists x' \in \alpha(X)\colon \\
&\qquad (t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota'[X\colon x\mapsto x'])}\gamma \\
&\qquad [\text{for } X \in S \text{ and } x \in V'_X] \ , \\
(t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\text{“}\forall x\colon X.\ \gamma\text{”} &\iff \forall x' \in \alpha(X)\colon \\
&\qquad (t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota'[X\colon x\mapsto x'])}\gamma \\
&\qquad [\text{for } X \in S \text{ and } x \in V'_X] \ .
\end{aligned}
$$

Figure 4. Recursive definition of satisfaction relation for behavior traces.

In the following we define the semantics of BTAs for two different domains: BTs and CTs. The former represents the interpretation of a BTA over a sequence of CPVs while the latter represents its interpretation over a sequence of ACs containing the corresponding component.

B. Semantics: Behavior Traces

In the following, we define what it means for a BT to satisfy a BTA.

Definition 11 (Semantics BTs): Given algebra A and corresponding VAs $\iota' = (\iota'_s)_{s\in S} \in \mathcal{I}_A^{V'}$ for variables V'. With $(t,n)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma$, defined recursively by the equations listed in Fig. 4, we denote that BT $t \in \mathcal{B}(c)$ satisfies BA $\gamma \in \Gamma_\Sigma^{(V,V')}(S_i)$ at time $n \in \mathbb{N}$. A BT $t \in \mathcal{B}(c)$ satisfies BA $\gamma \in \Gamma_\Sigma^{(V,V')}(S_i)$ iff $(t,0)\ {}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma$.

C. Semantics: Configuration Traces

In the following, we define what it means for a CT to satisfy a BTA.

Definition 12: Given algebra A and corresponding VAs $\iota' = (\iota'_s)_{s\in S} \in \mathcal{I}_A^{V'}$ for variables V'. With $(t,n)\ {}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\gamma$, defined recursively by the equations listed in Fig. 5, we denote that CT $t \in \mathcal{R}(C)$ satisfies BA $\gamma \in \Gamma_\Sigma^{(V,V')}(S_i)$ at time $n \in \mathbb{N}$. A CT $t \in \mathcal{R}(c)$ satisfies BA $\gamma \in \Gamma_\Sigma^{(V,V')}(S_i)$ iff $(t,0)\ {}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\gamma$.

An important property of this interpretation is that whenever a component is not activated anymore, then every formula is satisfied.

Lemma 1: Given IS $S_i = (P_{in}, P_{out}, t^p) \in \mathcal{S}_I(\Sigma)$, healthy set of components $C \subseteq \mathcal{C}$, component $c = (id, I, O) \in C$, algebra $A \in \mathcal{A}(\Sigma)$, VA $\iota' = (\iota'_s)_{s\in S}$ with $\iota'_s\colon \iota'_s \to \alpha(s)$ for each sort $s \in S$, and PIs $\delta^i\colon I \leftrightarrow P_{in}$ and $\delta^o\colon I \leftrightarrow P_{out}$.

$$
\begin{aligned}
(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\phi \iff{}& \forall \underset{\cdot}{i}\geq n\colon \big(c\in[t(i)]^1 \\
&\wedge\neg\exists i>\underset{\cdot}{k}\geq n\colon c\in[t(k)]^1\big) \\
&\implies \exists\iota\in\mathcal{I}^V_A\colon t(i)\,{}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota\cup\iota')}\phi \\
&[\text{for } \phi\in\varphi^V_\Sigma(S_i)]\ , \\
(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\text{“}\bigcirc\gamma\text{”} \iff{}& \forall \underset{\cdot}{i}\geq n\colon \big(c\in[t(i)]^1 \\
&\wedge\neg\exists i>\underset{\cdot}{k}\geq n\colon c\in[t(k)]^1\big) \\
&\implies (t,i+1)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\gamma\ , \\
(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\text{“}\Diamond\gamma\text{”} \iff{}& \exists \underset{\cdot}{n}'\geq n\colon (t,n')\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\gamma\ , \\
(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\text{“}\Box\gamma\text{”} \iff{}& \forall \underset{\cdot}{n}'\geq n\colon (t,n')\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\gamma\ , \\
(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\text{“}(\gamma\ \mathcal{U}\ \gamma')\text{”} \iff{}& \exists \underset{\cdot}{i}\geq n\colon (t,i)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota)}\gamma' \\
&\wedge\forall \underset{\cdot}{k}\geq n\colon \big(\exists k\leq \underset{\cdot}{i}'<i\colon c\in\big[t(i')\big]^1\big) \\
&\implies (t,k)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota)}\gamma\ , \\
(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\text{“}\exists x\colon X.\ \gamma\text{”} \iff{}& \exists x'\in\alpha(X)\colon \\
&(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota'[X\colon x\mapsto x'])}\gamma \\
&[\text{for } X\in S \text{ and } x\in V'_X]\ , \\
(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\text{“}\forall x\colon X.\ \gamma\text{”} \iff{}& \forall x'\in\alpha(X)\colon \\
&(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota'[X\colon x\mapsto x'])}\gamma \\
&[\text{for } X\in S \text{ and } x\in V'_X]\ .
\end{aligned}
$$

Figure 5. Recursive definition of satisfaction relation for configuration traces.

For each CT $t\in\mathcal{R}(C)$, BTA $\gamma\in\varphi^V_\Sigma(S_i)$ and point in time $n\in\mathbb{N}$, such that $\forall n'\geq n\colon c\notin[t(n')]^1$, we have:

$$(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota)}\gamma\ . \tag{5}$$

D. Soundness of Dynamic Interpretation

In the following, we provide an important property to ensure soundness of Def. 12. It ensures that every behavior specification which holds for the traditional interpretation is also true for our alternative one. Note that we denote with $s\circ s'$ the concatenation of a sequences s with a sequence s'.

Lemma 2: Given IS $S_i=(P_{in},P_{out},t^p)\in\mathcal{S}_I(\Sigma)$, a healthy set of components $C\subseteq\mathcal{C}$, component $c=(id,I,O)\in C$, algebra $A\in\mathcal{A}(\Sigma)$, VA $\iota'=(\iota'_s)_{s\in S}$ with $\iota'_s\colon\iota'_s\to\alpha(s)$ for each sort $s\in S$, and PIs $\delta^i\colon I\leftrightarrow P_{in}$ and $\delta^o\colon I\leftrightarrow P_{out}$. For each CT $t\in\mathcal{R}(C)$, BTA $\gamma\in\varphi^V_\Sigma(S_i)$ and points in time $n,\ i\in\mathbb{N}$ we have:

$$\frac{i\geq n \qquad c\in[t(i)]^1 \qquad \forall i>\underset{\cdot}{k}\geq n\colon c\notin[t(k)]^1}{\begin{gathered}(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\gamma \impliedby \\ \exists q\in\mathcal{B}(c)\colon \Big(\Pi_c(t)\circ q, \#(\Pi_c(t\mid_{i+1}))-1\Big)\,{}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma\end{gathered}}\ . \tag{6}$$

The final property is now a direct consequence of Lem. 2.

Theorem 1 (Soundness): Given IS $S_i=(P_{in},P_{out},t^p)\in\mathcal{S}_I(\Sigma)$, healthy set of components $C\subseteq\mathcal{C}$, component $c=(id,I,O)\in C$, algebra $A\in\mathcal{A}(\Sigma)$, VA $\iota'=(\iota'_s)_{s\in S}$ with $\iota'_s\colon\iota'_s\to\alpha(s)$ for each sort $s\in S$, and PIs $\delta^i\colon I\leftrightarrow P_{in}$ and $\delta^o\colon I\leftrightarrow P_{out}$. For each CT $t\in\mathcal{R}(C)$ and BTA $\gamma\in\varphi^V_\Sigma(S_i)$ we have:

$$t\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\gamma \impliedby \exists q\in\mathcal{B}(c)\colon \Pi_c(t)\circ q\,{}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma\ .$$

E. Completeness of Dynamic Interpretation

The forward direction of Thm. 1 does not hold, in general, i.e., not every formula which holds in our alternative interpretation is true in the traditional one.

Example 4 (Why does the forward direction not hold?): Consider a healthy set of components $C\subseteq\mathcal{C}$, CTs $t\in\mathcal{R}(C)$, and component $c\in C$, such that $\forall n\in\mathbb{N}\colon c\notin[t(n)]^1$. According to Def. 12, $t\,{}^{t}_{k}{\models}^{(c,\delta)}_{(A,\iota)}$`false` is vacuously true. On the other hand, $\Pi_c(t)\,{}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota)}$`false` is obviously false according to Def. 11.

Thus, our alternative semantics is not complete, in general. Nevertheless, it can be shown, that it is indeed complete for *fair* configuration traces.

Lemma 3: Given IS $S_i=(P_{in},P_{out},t^p)\in\mathcal{S}_I(\Sigma)$, a healthy set of components $C\subseteq\mathcal{C}$, component $c=(id,I,O)\in C$, algebra $A\in\mathcal{A}(\Sigma)$, VA $\iota'=(\iota'_s)_{s\in S}$ with $\iota'_s\colon\iota'_s\to\alpha(s)$ for each sort $s\in S$, and PIs $\delta^i\colon I\leftrightarrow P_{in}$ and $\delta^o\colon I\leftrightarrow P_{out}$. For each CT $t\in\mathcal{R}(C)$, such that $\mathit{fair}(c,t)$, BTA $\gamma\in\varphi^V_\Sigma(S_i)$, and points in time $n,\ i\in\mathbb{N}$, we have:

$$\frac{i\geq n \qquad c\in[t(i)]^1 \qquad \forall i>\underset{\cdot}{k}\geq n\colon c\notin[t(k)]^1}{\begin{gathered}(t,n)\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\gamma \implies \\ \exists q\in\mathcal{B}(c)\colon \Big(\Pi_c(t)\circ q, \#(\Pi_c(t\mid_{i+1}))-1\Big)\,{}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma\end{gathered}}\ . \tag{7}$$

The final property is now a direct consequence of Lem. 3.

Theorem 2 (Relative completeness): Given IS $S_i=(P_{in},P_{out},t^p)\in\mathcal{S}_I(\Sigma)$, a healthy set of components $C\subseteq\mathcal{C}$, component $c=(id,I,O)\in C$, algebra $A\in\mathcal{A}(\Sigma)$, VA $\iota'=(\iota'_s)_{s\in S}$ with $\iota'_s\colon\iota'_s\to\alpha(s)$ for each sort $s\in S$, and PIs $\delta^i\colon I\leftrightarrow P_{in}$ and $\delta^o\colon I\leftrightarrow P_{out}$. For each BTA $\gamma\in\varphi^V_\Sigma(S_i)$ and CT $t\in\mathcal{R}(C)$, such that $\mathit{fair}(c,t)$, we have:

$$t\,{}^{t}_{k}{\models}^{(c,\delta^i,\delta^o)}_{(A,\iota')}\gamma \implies \exists q\in\mathcal{B}(c)\colon \Pi_c(t)\circ q\,{}^{t}_{b}{\models}^{(\delta^i,\delta^o)}_{(A,\iota')}\gamma\ .$$

IV. Related Work

Related work can be found in applications of temporal logics for the specification of dynamic architectures as well as in works providing different semantic interpretations to temporal specifications.

A. Temporal Logics for the Specification of Dynamic Architectures

Over the last years, some approaches emerged which apply Temporal Logics to the specification of Dynamic Architectures. Aguirre and Maibaum [1] provide an approach to specify properties of dynamic architectures by means of temporal-logic formulas. Dormoy et. al [8] provide a temporal logic for dynamic reconfiguration called FTPL. FTPL allows for the specification of component architecture evolution which is modeled by a transition system over architecture configurations and so-called evolution operations. Recently, Fiadeiro and Lopes [11] provide an approach based on an abstract notion of state and configuration. Finally, Castro et. al [6] provide a category-theoretic approach to provide semantics to specification of component-behavior in dynamic architectures.

While all of these approaches use temporal logic for the specification of component-behavior, to the best of our knowledge, there is no work which studies the semantics of temporal specifications of component-behavior in terms of dynamic architectures. Thus, with our work we actually complement these approaches by providing a detailed (and verified) interpretation of the semantics of specifications of component-behavior for dynamic architectures.

B. Semantics of Temporal Logics

Another area of related work is formed by work studying different semantics for temporal specifications. One of the first studies in this area is provided by Emerson [9]. In this work, the author classifies various semantics for temporal specifications based on constraints they impose on the allowable set of computation paths. More recently, Bozzelli [2] studies the relationship between two interpretations of temporal specifications: hyper and epistemic temporal logic interpretations. Thereby they unify KCTL* (an extension of CTL* [10]) and HyperCTL*. Finally, Krishna et al. [12] provide a good overview of various timed interpretations of temporal specifications.

While all these studies investigate different semantics for temporal specifications, to the best of our knowledge, this work is the first attempt to study the semantics of temporal specifications for dynamic architectures.

V. Conclusion

With this article we provide an interpretation of temporal specifications of component-behavior for dynamic architectures. Thus, the major contributions can be summarized as follows: (i) We extend our model of dynamic architectures introduced in [14] by the notion of behavior traces to model behavior of single components. Thereby we introduce and study an operator to extract the behavior of single components out of a given configuration trace. (ii) We introduce the notion of behavior trace assertions to specify the behavior of single components. (iii) We provide two interpretations thereof for two different semantic domains: the domain of behavior traces resembling the traditional interpretation of temporal specifications over components and the domain of configuration traces resembling an alternative interpretation for dynamic architectures. (iv) We show soundness of our alternative interpretation w.r.t. the traditional interpretation. (v) Finally, we show completeness for *fair* configuration traces (i.e., configuration traces in which a component is guaranteed to be activated infinitely many times).

Our results allow to interpret temporal specifications of component-behavior over dynamic architectures. Thus, they can be used to support in the specification of such architectures by allowing to separate the specification of component-behavior from other aspects such as component activation and architecture reconfiguration. To further support the verification of dynamic architectures, future work should investigate reasoning about component-behavior in a dynamic context with the aim to develop a calculus for such architectures.

Acknowledgments: We would like to thank Manfred Broy, Mario Gleirscher, Vasileios Koutsoumpas, and all the anonymous reviewers of TASE 2017 for their comments and helpful suggestions on earlier versions of the paper. This work was partially funded by the German Federal Ministry of Education and Research (BMBF) under grant "01Is16043A".

References

[1] N. Aguirre and T. Maibaum. "A temporal logic approach to the specification of reconfigurable component-based systems." In: *Automated Software Engineering.* 2002, pp. 271–274. DOI: 10.1109/ase.2002.1115028.

[2] L. Bozzelli et al. "Unifying Hyper and Epistemic Temporal Logics." In: *Foundations of Software Science and Computation Structures.* Heidelberg, 2015, pp. 167–182. DOI: 10.1007/978-3-662-46678-0_11.

[3] J. S. Bradbury et al. "A survey of self-management in dynamic software architecture specifications." In: *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems - WOSS '04.* 2004, pp. 28–33. DOI: 10.1145/1075405.1075411.

[4] M. Broy. "A Logical Basis for Component-Oriented Software and Systems Engineering." In: *The Computer Journal* 53.10 (Feb. 2010), pp. 1758–1782. DOI: 10.1093/comjnl/bxq005.

[5] M. Broy. "A Model of Dynamic Systems." In: *From Programs to Systems. The Systems perspective in Computing.* Ed. by S. Bensalem et al. Vol. 8415. 2014, pp. 39–53. DOI: 10.1007/978-3-642-54848-2_3.

[6] P. F. Castro et al. "Towards Managing Dynamic Reconfiguration of Software Systems in a Categorical Setting." In: *Lecture Notes in Computer Science.* 2010, pp. 306–321. DOI: 10.1007/978-3-642-14808-8_21.

[7] P. C. Clements. "A survey of architecture description languages." In: *Proceedings of the 8th International Workshop on Software Specification and Design.* 1996, p. 16. DOI: 10.1109/iwssd.1996.501143.

[8] J. Dormoy et al. "Using temporal logic for dynamic reconfigurations of components." In: *Formal Aspects of Component Software.* 2010, pp. 200–217. DOI: 10.1007/978-3-642-27269-1_12.

[9] E. A. Emerson. "Alternative semantics for temporal logics." In: *Theoretical computer science* 26.1 (1983), pp. 121–130. DOI: http://dx.doi.org/10.1016/0304-3975(83)90082-8.

[10] E. A. Emerson and J. Y. Halpern. "'Sometimes' and 'Not Never' Revisited: On Branching Versus Linear Time Temporal Logic." In: *J. ACM* 33.1 (Jan. 1986), pp. 151–178. DOI: 10.1145/4904.4999.

[11] J. L. Fiadeiro and A. Lopes. "A Model for Dynamic Reconfiguration in Service-oriented Architectures." In: *Software & Systems Modeling* 12.2 (2013), pp. 349–367. DOI: 10.1007/s10270-012-0236-1.

[12] S. N. Krishna et al. "Metric Temporal Logic with Counting." In: *Foundations of Software Science and Computation Structures.* Vol. 9634. 2016, pp. 335–352. DOI: 10.1007/978-3-662-49630-5_20.

[13] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems.* 1992. DOI: 10.1007/978-1-4612-0931-7.

[14] D. Marmsoler and M. Gleirscher. "Specifying Properties of Dynamic Architectures using Configuration Traces." In: *Theoretical Aspects of Computing.* 2016. DOI: 10.1007/978-3-319-46750-4_14.

[15] N. Medvidovic. "ADLs and dynamic architecture changes." In: *Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints '96) on SIGSOFT '96 workshops -.* 1996, pp. 24–27. DOI: 10.1145/243327.243340.