Context-Based Behavioral Equivalence of Components in Self-Adaptive Systems

Narges Khakpour[1,2], Marjan Sirjani[3], Ursula Goltz[1]

[1] IPS, Technical University of Braunschweig, Germany
[2] Tarbiat Modares University, Iran
[3] Reykjavik University, Iceland
email: khakpour@ips.cs.tu-bs.de

Abstract An important challenge to realize dynamic adaptation is finding suitable components for substitution or interaction according to the current context. A possible solution is checking behavioral equivalence of components in different contexts. Two components are equivalent with respect to a context, if they behave equivalently in that context. In this work, we deal with context-specific behavioral equivalence of PobSAM components. PobSAM is a flexible formal model for developing and modeling evolving self-adaptive systems. A PobSAM model is a collection of actors, views, and autonomous managers. Autonomous managers govern the behavior of actors by enforcing suitable context-based policies. Views provide contextual information for managers to control and adapt the actors behavior. Managers are the core components used to realize adaptation by changing their policies. They are modeled as meta-actors whose configurations are described using a multi-sorted algebra called CA. The behavior of mangers depends on the context in which they are executing. In this paper, we present an equational theory to reason about context-specific behavioral equivalence of managers independently from actors. To this end, we introduce and axiomatize a new operator to consider the interaction of managers and the context. This equational theory is based on the notion of statebased bisimilarity and allows us to reason about the behavioral equivalence of managers as well as the behavioral equivalence of the constitutes of managers (i.e., policies and configurations). We illustrate our approach through an example.

1 Introduction

Today's complex systems often need to operate in dynamic, open and heterogeneous environments, so they must be able to adapt themselves at run-time to handle varying resources, user mobility, changing user needs, and system faults. PobSAM (Policy-based Self-Adaptive Model) [8] is a flexible formal model to develop, specify and verify self-adaptive systems which uses policies as the fundamental mechanism to govern the system behavior. A PobSAM model is composed of a collection of autonomous managers, views and actors. Autonomous managers are meta-actors responsible for monitoring and handling events by enforcing suitable policies. Each manager has a set of configurations where one

of the configurations is active at a time. The manager changes its active configuration dynamically in response to the changing circumstances according to adaptation policies. The managers monitor actors through views, i.e. views provide contextual information for the managers. One of the distinguished advantages of PobSAM is that it allows us to modify the configurations (or policies) of managers at runtime. This feature makes PobSAM a suitable model to develop evolving self-adaptive systems.

In dynamic environments such as ubiquitous computing world, many systems must cope with variable resources (bandwidth, server availability, etc.), system faults (servers and networks going down, failure of external components, etc.), and changing user priorities (high-fidelity video streams at one moment, low-fidelity at another, etc.) [3]. In such environments, the system requires to continue running with only minimal human intervention, and the component assessment and integration process must be carried out automatically. We refer to the component assessment as the problem of identifying a component with desired behavior that can replace another component or can be used for interaction in a specific context. A possible solution to this problem relies on detecting the behavioral equivalence of a particular component with desired behavior and a candidate component that could maintain that behavior. Generally, we categorize the behavioral equivalence of two components as context-independent or context-specific. The *context* of a component is defined as the environment in which the component is running. Two components that are context-independent equivalent behave equivalently in any environment, while the equivalence of two components that are context-specific equivalent, depends on the environments in which they are running.

Managers are the main components to control and adapt the system behavior in PobSAM. Thus, it is an important issue to analyze the behavioral equivalence of managers when studying the dynamic replacement and interaction of components for software adaptation. In order to ensure the correctness of the whole system behavior, we have to provide approaches to analyze the behavioral equivalence of the original manager and the adapted one.

Contribution. We previously proposed PobSAM in [8] which has a formal foundation that employs an integration of algebraic formalisms and actor-based models. The actors of PobSAM are modeled using actor-based models while the algebra CA (Configuration Algebra) is proposed to specify the configurations of managers. Due to the fact that the managers control and adapt the system behavior using dynamic policies which are context-dependent rules, the behavior of a manager depends on the context in which it is enforcing policies. We must investigate context-specific behavioral equivalence of managers. Furthermore, we can modify the policies and the configurations of a manager dynamically. Thus, this equational theory should allow us to reason about context-specific behavioral equivalence of policies and configurations as well. In this paper, we develop an equational theory to analyze context-specific behavioral equivalence of managers, based on a notion of behavioral equivalence called statebased bisimulation. The context of managers is specified by a labeled state transition system. The context

interacts with the managers by synchronous message passing. We extend CA with a new operator to consider the interaction of managers and the context. Then, we present the axioms for this operator to check behavioral equivalence of managers. In our equational theory, we can reason about context-specific behavioral equivalence of policies, configurations and managers separately. As the manager may evolve by changing its policies or configurations, this theory allows us to only reason about the modified constitutes without the need to check the whole model of the system. *An important advantage of this equational theory is that it analyzes the behavioral equivalence of the manager layer independently from the actor layer using the context.*

The remainder of this paper is organized as follows. In Section 2 we introduce an example to illustrate our approach. In Section 3, we have a brief review on PobSAM. Section 4 deals with modeling our case study in PobSAM. We introduce the notion of statebased bisimulation in Section 5. An equational theory is proposed to check context-specific behavioral equivalence of managers in Section 6. In Section 7, we give a summary of related work and Section 8 presents our conclusions.

2 Illustrating Example

We use a simple example borrowed from [16] to illustrate our approach. In this example, a team of collaborating unmanned autonomous vehicles (UAVs) are used for a search and rescue operation. Assume a person with a body sensor network (BSN) is wounded in an area and needs help. The BSN sends a help message to a mission commander. A mission is defined by the commander to save the wounded person: one or more UAVs with video cameras act as surveyors and others perform a communication relay function. The UAVs are required to adapt their behavior according to the changes of the environment. According to the role of a UAV in the mission, a set of policies is used by that UAV to control its behavior. However, the role of a UAV is not fixed, and subsequently, the policies used to control the UAV behavior must change dynamically. For instance, the video camera of a surveyor may break down and that surveyor would act as a communication relay. Thus, various sets of policies are defined for a UAV and one of those sets of policies is active at a time, i.e. adaptation is performed by changing the set of policies used to control the UAV behavior.

3 PobSAM

A PobSAM model is composed of three layers:

- The *actor layer* is dedicated to the functional behavior of the system and contains computational entities.
- The *view layer* consists of view variables that provide an abstraction of the actors' states for the managers. A view variable is an actual state variable, or a function or a predicate applied to state variables of actors.

– The main layer of PobSAM is the *manager layer* containing the autonomous managers. Managers control the behavior of actors according to the predefined policies. A manager may have different configurations and dynamic adaptation is performed by switching among those configurations. A configuration consists of two classes of policies: governing policies and adaptation policies. A manager directs the actor behavior by sending messages to the actors according to governing policies. Adaptation policies are used for dynamic adaptation by switching among configurations. However, the adaptation cannot be done immediately and when the system reaches a safe state, the manager switches to the new configuration. A new mode of operation called adaptation mode is introduced in which a manager runs before switching to the new configuration. There are two kinds of adaptations called *loose adaptation* and *strict adaptation*. Under loose adaptation, the manager handles events in the adaptation mode by enforcing the governing policies of old configuration, while in the strict adaptation mode all the events are postponed until the system passes the adaptation mode safely.

A manager is defined as a tuple $m = \langle V_m, C_m, c_{init} \rangle$, with C_m the (finite) set of configurations of m, $c_{init} \in C_m$ its initial configuration, and V_m the (finite) set of view variables observable by m. A configuration $c \in C_m$ is defined as $c = \langle g, p \rangle$, where $g = \{g_1, ..., g_n\}$ and p indicate the governing policy set and the adaptation policies of c, respectively. The constants $\top$ and $\bot$ stand for "True" and "False", respectively.

Governing Policies A simple governing policy $g_i = \langle o, e, \psi \rangle \bullet a$, $1 \leq i \leq n$ consists of priority $o \in \mathbb{N}$, event $e \in E$ where E is an assumed set of possible events, condition ψ (a Boolean term) and an action a. The actions in the governing policies are specified using an algebra CA^a defined as follows. We let a, a', a'' denote action terms, while an (atomic) action α could be an internal action, an output action ($\alpha!$) in form of $r.msg$ (i.e. sending the message msg to actor r), or an input action ($\alpha?$).

$$a \stackrel{\text{def}}{=} a; a' \mid a \parallel a' \mid a \lfloor\!\lfloor a' \mid a + a' \mid \phi :\to a \mid \alpha \mid \alpha! \mid \alpha? \mid \delta_a$$

Thus an action term can be a sequential composition (;), a parallel composition ($\parallel$), a left parallel composition ($\lfloor\!\lfloor$ which is as $\parallel$ but the first action that is performed comes from the left operand), a non-deterministic choice (+), or a conditional choice ($\phi :\to a$). Moreover, we have the special constant δ_a as the deadlock action for governing policies. Operator precedences are assigned, from highest precedence to the lowest, to the conditional choice, the parallel composition operators, the sequential composition and the non-deterministic choice operators. Whenever a manager receives an event e, it identifies all simple governing policies that are triggered by that event , i.e. are of the form $\langle o, e, \psi \rangle \bullet a$ for some o, ψ, and a. For each of these activated policies, if the policy condition ψ evaluates to true and there is no other triggered governing policy with priority higher than o, then action a is executed. Table 1 shows CA^a axioms.

Adaptation Policies Adaptation policies are specified using the algebra CA^p as follows:

$a + a' = a' + a$	A1	$a \parallel a' = a' \parallel a$	AP1
$(a + a') + a'' = a + (a' + a'')$	A2	$(a \parallel a') \parallel a'' = a \parallel (a' \parallel a'')$	AP2
$a + a = a$	A3	$(a + a') \between a'' = (a \between a'') + (a' \between a'')$	AP3
$a + \delta_g = a$	A4	$a \parallel a' = a \between a' + a' \between a$	AP4
$\delta_g ; a = \delta_g$	A5	$\alpha \between a = \alpha ; a$	AP5
$(a + a') ; a'' = a ; a'' + a' ; a''$	A6	$(\alpha ; a) \between a' = \alpha ; (a \parallel a')$	AP6
$(a ; a') ; a'' = a ; (a' ; a'')$	A7		

$\top :\to a = a$	C1	$\bot :\to a = \delta$	C2
$\phi :\to (a + a') = \phi :\to a + \phi :\to a'$	C3	$\phi :\to (a ; a') = \phi :\to a ; a'$	C4
$\phi :\to (\psi :\to a) = (\phi \wedge \psi) :\to a$	C5	$(\phi \vee \psi) :\to a = \phi :\to a + \psi :\to a$	C6
$\phi :\to \delta = \delta$	C7	$\phi :\to a \between a' = \phi :\to (a \between a')$	C8

Table 1. Action Algebra CA^a

$$p \stackrel{\text{def}}{=} \langle o, e, \psi, \lambda, \phi \rangle \bullet c \mid p \oplus p \mid \delta_p$$

which consists of priority $o \in \mathbb{N}$, event $e \in E$, and a condition ψ (a Boolean term) for triggering the adaptation. Moreover, condition ϕ is a Boolean term indicating the conditions for applying the adaptation, λ is the adaptation type (loose, denoted $\bot$, or strict, denoted $\top$), and c is the new configuration. Informally, simple adaptation policy $\langle o, e, \psi, \lambda, \phi \rangle \bullet c$ indicates that when event e occurs and the triggering condition ψ holds, if there is no other triggered adaptation policy with priority higher than o, then the manager evolves to the strict or loose adaptation mode as given by λ. When the condition ϕ is true, it will perform adaptation and switch to the configuration c. The adaptation policy of a manager is defined as composition($\oplus$) of the simple adaptation policies. Furthermore, δ_p indicates the unit element for the composition operator.

4 Formal Modeling of Collaborating UAVs

Figure 1 shows the PobSAM model of a UAV partially. This model contains actors `motor, video camera, GSM` and `infrared sensors` where Rebeca[15] specification of `motor` is given in figure 1. Rebeca is an actor-based model used to specify the actor layer in [8]. The view layer has a number of views denoting the current location, speed, energy level etc of UAVs. As an example, the view `UAV1speed` indicates the speed of `UAV1` which reflects the value of the statevar `speed` of actor `UAV1motor`.

A UAV has a manager named `UAVCntrlr` for controlling different components of the UAV. A `UAVCntrlr` has three different configurations including `surveyorConf, idleConf` and `relayConf`. It enforces different sets of policies in each configuration to control the behavior of UAV. For instance, the configuration `surveyorConf` contains the adaptation policies `{p1,p2}` and the governing policy set `{g1,g2,g3}`. Assume a situation that the video camera of a surveyor breaks down and we need to use this UAV as a relay. We define the adaptation

policy `p1` which informally states that "when the video camera is broken down, if the wounded person has not been found and the UAV has required capability to act as a relay, it should switch to the `relayConf` configuration". We specify this policy formally as follows in which *brokencamera* is an event. The view variable *canRelay* indicates if the UAV has required capability to act as a relay, and the view variable *success* denotes whether the wounded person has been found or not.

$$p_1 \stackrel{def}{=} \langle 1, brokenCamera, \neg success \wedge canRelay, \top, \top \rangle \bullet relayConf$$

The simple governing policy `g1` states that when the wounded person is found, the UAV must request his health information from his BSN and send a "success" message to the commander. The algebraic form of this policy is $g_1 \stackrel{\text{def}}{=} \langle 1,\ found(x,y),\ \top \rangle \bullet\ a_1$ where *found(x,y)* denotes an event that the wounded person has been found at location (x, y), and

$$\begin{aligned} a_1 = {} & BSN.reqHealthinfo()? \parallel \\ & relay1.send(success(x,y), commander)! \end{aligned}$$

5 Statebased Bisimulation

In PobSAM, the managers are running concurrently with the actors; the computation at the actor layer is reflected at the view layer, the state changing of the view layer leads to triggering and enforcing policies by the managers. Subsequently, the enforcement of policies results in new computations to be done at the actor layer. We specify the context based on specification of the view layer, the actor interfaces and possibly the interfaces of other managers. Given the formal specification of a context, we check the behavioral equivalence of managers in that context. A context is defined as follows:

Definition 1. *A context is defined as tuple* $T_c = \langle V, S_c, s_c^0, \mathcal{A}_c^I, \mathcal{A}_c^O, \mathcal{A}_c^H, \rightarrow_c \rangle$ *where*

- $V = \{v_1, ..., v_n\}$ *is the set of view variables.*
- S_c *is the set of states where a state* $s \in S_c$ *is of the form* $\langle v_1, ..., v_n \rangle$*.*
- s_c^0 *is the initial state.*
- $\mathcal{A}_c^I, \mathcal{A}_c^O$ *and* $\mathcal{A}_c^H$ *are disjoint sets of input, output and internal actions where* $\mathcal{A}_c = \mathcal{A}_c^I \cup \mathcal{A}_c^O \cup \mathcal{A}_c^H$*.*
- $\rightarrow_c \subseteq S_c \times \mathcal{A}_c \times S_c$ *is the set of transitions.*

In this paper, we extend CA^a with a new operator, called CA^a_Θ. We present a context-specific behavioral equivalence theory for CA^a_Θ. Then we use this basic theory to reason about context-specific behavioral equivalence of policies, configurations and managers. We define the operational meaning of CA^a_Θ terms by a transition system with data whose states are pairs of a CA^a term and a context state. Let A denote the set of CA^a terms. The set of all pairs over $A \times S_c$ is denoted by $S_{A \times S_c}$. We define a state transition system with data as follows:

```
manager UAVCntrlr
{
statevars {
}
configurations{
   surveyorConf=[p1,p2] [g1,g2,g3];
   //definition of relayConf and idleConf configurations
  }
policies{
 p1[strict]:on brokenCamera if (!success && canRelay)
           switchto relayConf when true priority 1 ;
  g1 : on found(x,y) if true do
(BSN.reqhealthinfo() || relay.send(success(x,y),commander))
          priority 1 ;
//definition of governing and adaptation policies
  }
}
views {
   byte UAV1speed as UAV1motor.speed;
   //definition of other views
}
Actors {
 reactiveclass motor() {
    knownobjects {}
    statevars{public byte speed; }
    msgsrv forward()  {
      ...
     }
    msgsrv stop()  {
      ...
     }
 //definition of other message servers
 }
 //definition of other reactive classes
}
```

Fig. 1. The Partial PobSAM Model of a UAV

Definition 2. *A state transition system with data defined over the context T_c, is $T(a, s_c^0) = \langle S_{A\times S_c}, \rightarrow, \mathcal{A}^I, \mathcal{A}^O, \mathcal{A}^H, (a, s_c^0)\rangle$ where $S_{A\times S_c}$ is a set of states, (a, s_c^0) is the initial state, $\rightarrow \subseteq S_{A\times S_c} \times \mathcal{A} \times S_{A\times S_c}$ and $\mathcal{A} = \mathcal{A}^I \cup \mathcal{A}^O \cup \mathcal{A}^H$.*

It worth mentioning that $\mathcal{A}_c \subseteq \mathcal{A}$, $\mathcal{A}_c^I \subseteq \mathcal{A}^I$, $\mathcal{A}_c^O \subseteq \mathcal{A}^O$ and $\mathcal{A}_c^H \subseteq \mathcal{A}^H$. We use a notion of bisimilarity called statebased bisimulation [5] for expressing context-specific behavioral equivalence of CA_Θ^a terms defined as follows:

Definition 3. ***Statebased bisimulation*** *A binary relation $\mathcal{R} \subseteq S_{A\times S_c} \times S_{A\times S_c}$ is a statebased bisimulation iff for all $(r, s), (q, s) \in S_{A\times S_c}$ with $((r, s), (q, s)) \in \mathcal{R}$:*

- *whenever $(r, s) \xrightarrow{\alpha} (r', s')$ for some $\alpha \in \mathcal{A}$ and (r', s'), then, for some q', also $(q, s) \xrightarrow{\alpha} (q', s')$ and $((r', s'), (q', s')) \in \mathcal{R}$.*
- *Conversely, whenever $(q, s) \xrightarrow{\alpha} (q', s')$ for some $\alpha \in \mathcal{A}$ and (q', s'), then, for some r', also $(r, s) \xrightarrow{\alpha} (r', s')$ and $((r', s'), (q', s')) \in \mathcal{R}$.*

A pair $(r, s) \in S_{A\times S_c}$ is statebased bisimilar with a pair $(r', s') \in S_{A\times S_c}$ with respect to the context T_c, written by $(r, s) \leftrightarrows_{T_c} (r', s')$ iff $s = s'$ and there is a statebased bisimulation containing the pair $((r, s), (r', s'))$.

A state transition system with data $T(r, s) = \langle S_{A\times S_c}, \rightarrow, \mathcal{A}^I, \mathcal{A}^O, \mathcal{A}^H, (r, s)\rangle$ is statebased bisimilar with the transition system with data $T(q, s') = \langle S_{A\times S_c}, \rightarrow', \mathcal{A}'^I, \mathcal{A}'^O, \mathcal{A}'^H, (q, s')\rangle$, written by $T(r, s) \leftrightarrows_{T_c} T(q, s')$ iff $(r, s) \leftrightarrows_{T_c} (q, s')$. Furthermore, two closed terms r , q over CA^a are statebased bisimilar with respect to the context T_c, written by $r \leftrightarrows_{T_c} q$, iff $T(r, s) \leftrightarrows_{T_c} T(q, s)$ for all $s \in S_c$.

6 Context-specific Behavioral Equivalence

In this section, we use the notion of statebased bisimulation to reason about context-specific behavioral equivalence of managers and their constituents. We introduce a new operator (Θ) to consider interactions of managers and the context. The axiom system of CA^a is extended to check statebased bisimulation of CA^a_Θ terms. Then, context-specific behavioral equivalence of policies, configurations and managers are defined based on the proposed equational theory for CA^a_Θ.

6.1 Context-specific Behavioral Equivalence of Actions

In our model, the context and the managers run concurrently and interact by synchronous message passing. Since the conditions of an action are evaluated over the context state, therefore the concrete action carried out by the manager depends on the context. There are three types of computation steps: (i) the manager and the context synchronize on their shared input-output actions, (ii) the context performs an internal action, or (iii) the manager performs an internal action. In the cases (i) and (iii), the conditions of the action are evaluated over the state of context. We introduce the operator Θ to compute the concrete action done by a manager, regarding the interactions of the manager and a context.

Let a denote a term of CA^a which must be performed by a manager, and $T_c = \langle V, S_c, s^0_c, \mathcal{A}^I_c, \mathcal{A}^O_c, \mathcal{A}^H_c, \rightarrow_c\rangle$ denote an arbitrary context. Assume the current state of context is $s \in S_c$ and the manager starts the enforcement of action a. The operator $\Theta_s(a)$ gives the concrete action performed by the manager as the result of performing action a when the context starts its execution in state $s_c \in S_c$. The structural operational semantics of CA^a_Θ extended is described by the transition rules given in Figure 2 in addition to the transition rules proposed in [8]. The transition $a \xrightarrow{[\phi]\alpha} a'$ means that a can evolve to a' by performing action α under condition ϕ.

Figure 3 presents the axioms for Θ in which $\alpha' \in \mathcal{A}^H_c$. This axiom system together with the axioms of Table 1 are used to check context-specific behavioral equivalence of actions. We can formulate action a in form of $a = \sum_i a_i$ using the axioms presented in Table 1 where term a_i is the sequential composition of conditional actions (i.e. of the form $\phi :\rightarrow \alpha$). Thus, we give the axioms for conditional choice, non-deterministic choice and sequential composition operators. Due to the lack of space, we restrict ourselves to present axioms for output

$$\frac{a \xrightarrow{[\phi]\alpha!} \surd \quad s \xrightarrow{\alpha?} s'}{\Theta_s(a) \xrightarrow{\alpha} \surd}\sigma_s(\phi) = \top \quad \text{LTR1} \qquad \frac{a \xrightarrow{[\phi]\alpha?} \surd \quad s \xrightarrow{\alpha!} s'}{\Theta_s(a) \xrightarrow{\alpha} \surd}\sigma_s(\phi) = \top \quad \text{LTR2}$$

$$\frac{a \xrightarrow{[\phi]\alpha!} a' \quad s \xrightarrow{\alpha?} s'}{\Theta_s(a) \xrightarrow{\alpha} \Theta_{s'}(a')}\sigma_s(\phi) = \top \quad \text{LTR3} \qquad \frac{a \xrightarrow{[\phi]\alpha?} a' \quad s \xrightarrow{\alpha!} s'}{\Theta_s(a) \xrightarrow{\alpha} \Theta_{s'}(a')}\sigma_s(\phi) = \top \quad \text{LTR4}$$

$$\frac{s \xrightarrow{\alpha} s' \quad \neg((\alpha = \alpha'? \wedge a \xrightarrow{\alpha'!} a') \vee (\alpha = \alpha'! \wedge a \xrightarrow{\alpha'?} a'))}{\Theta_s(a) \xrightarrow{\alpha} \Theta_{s'}(a)} \quad \text{LTR5}$$

$$\frac{a \xrightarrow{[\phi]\alpha} a'}{\Theta_s(a) \xrightarrow{\alpha} \Theta_s(a')}\sigma_s(\phi) = \top \quad \text{LTR6}$$

Fig. 2. Transition rules for the operator Θ

and internal actions. A number of axioms similar to TA3-4 are defined for input actions. TA2 asserts that non-deterministic choice of two actions a and a' from state s is equivalent to either execution of a or execution of a' from state s. In axioms TA3-6, the fist term $\sum_{(s,\alpha',s')} \Theta_{s'}(a)$ describes the case that an internal action (α') is executed by the context, and a will be evaluated from the next state of the context (s'). If the condition of an action is evaluated to false (i.e., $\sigma_s(\psi)$), action δ_a is executed. Moreover, if ψ is evaluated to true in state s, (i) execution of $\psi :\rightarrow \alpha!$ in state s can result in performing simple action α by synchronization with $\alpha?$ of the context (TA3), (ii) execution of $\psi :\rightarrow \alpha!; a$ in state s results in execution of simple action α synchronized with $\alpha?$ in the context, followed by execution of a from next state s' (TA4), (iii) execution of $\psi :\rightarrow \alpha$ in state s can result in performing the internal action α by the manager (TA5), (iv) execution of $\psi :\rightarrow \alpha; a$ in state s leads to execution of internal action α, followed by execution of a from state s (TA6).

Proposition 1. ***(Congruence)*** *Let a_1, a_2, a'_1 and a'_2 be terms of CA^a, ψ be an arbitrary boolean formula and $T_c = \langle V, S_c, s_c^0, \mathcal{A}_c^I, \mathcal{A}_c^O, \mathcal{A}_c^H, \rightarrow_c \rangle$ indicate the context. If for all $s \in S_c$, $\Theta_s(a_1) \leftrightarroweq_{T_c} \Theta_s(a'_1)$ and $\Theta_s(a_2) \leftrightarroweq_{T_c} \Theta_s(a'_2)$, then for all $s \in S_c$, $\Theta_s(a_1 + a_2) \leftrightarroweq_{T_c} \Theta_s(a'_1 + a'_2)$, $\Theta_s(a_1 \ ; \ a_2) \leftrightarroweq_{T_c} \Theta_s(a'_1 \ ; \ a'_2)$, $\Theta_s(\psi :\rightarrow a_1) \leftrightarroweq_{T_c} \Theta_s(\psi :\rightarrow a'_1)$, and $\Theta_s(a_1 \parallel a_2) \leftrightarroweq_{T_c} \Theta_s(a'_1 \parallel a'_2)$.*

Proof. See [7].

Theorem 1. ***(Soundness)*** *Let $T_c = \langle V, S_c, s_c^0, \mathcal{A}_c^I, \mathcal{A}_c^O, \mathcal{A}_c^H, \rightarrow_c \rangle$ be a context, and a and a' indicate two arbitrary terms of CA^a. If for all $s \in S_c$, $CA^a + (\text{TA1} - \text{TA6}) \vdash \Theta_s(a) = \Theta_s(a')$ then $\Theta_s(a)$ and $\Theta_s(a')$ are statebased bisimilar with respect to T_c, i.e. $\Theta_s(a) \leftrightarroweq_{T_c} \Theta_s(a')$.*

Proof. See [7].

6.2 Context-specific Behavioral Equivalence of Governing Policies

We have presented an axiomatized operator to check context-specific behavioral equivalence of actions in Section 6.1. We use this proposed equational theory

$$\Theta_s(\delta_a) = \delta_a \quad \textbf{TA1} \qquad \Theta_s(a + a') = \Theta_s(a) + \Theta_s(a') \quad \textbf{TA2}$$

$$\Theta_s(\psi :\rightarrow \alpha!) = \sum_{(s,\alpha',s')} \Theta_{s'}(\psi :\rightarrow \alpha!) \; + \; \sum_{(s,\alpha?,s')} \begin{cases} \alpha & \sigma_s(\psi) = \top \\ \delta_a & \sigma_s(\psi) = \bot \end{cases} \quad \textbf{TA3}$$

$$\Theta_s(\psi :\rightarrow \alpha! \; ; \; a) = \sum_{(s,\alpha',s')} \Theta_{s'}(\psi :\rightarrow \alpha! \; ; \; a) \; + \; \sum_{(s,\alpha?,s')} \begin{cases} \alpha; \; \Theta_{s'}(a) & \sigma_s(\psi) = \top \\ \delta_a & \sigma_s(\psi) = \bot \end{cases} \quad \textbf{TA4}$$

$$\Theta_s(\psi :\rightarrow \alpha) = \sum_{(s,\alpha',s')} \Theta_{s'}(\psi :\rightarrow \alpha) \; + \; \begin{cases} \alpha & \sigma_s(\psi) = \top \\ \delta_a & \sigma_s(\psi) = \bot \end{cases} \quad \textbf{TA5}$$

$$\Theta_s(\psi :\rightarrow \alpha \; ; \; a) = \sum_{(s,\alpha',s')} \Theta_{s'}(\psi :\rightarrow \alpha \; ; \; a) \; + \; \begin{cases} \alpha; \; \Theta_s(a) & \sigma_s(\psi) = \top \\ \delta_a & \sigma_s(\psi) = \bot \end{cases} \quad \textbf{TA6}$$

Fig. 3. Axioms of the operator Θ_s

to reason about context-specific behavioral equivalence of governing policies. A simple governing policy is a set of actions performed by a manager. Two simple governing policies are equivalent if and if they are activated by the same transitions of the context and their enforcement results in the same sequences of actions done by the manager.

Definition 4. *Let $T_c = \langle V, S_c, s_c^0, \mathcal{A}_c^I, \mathcal{A}_c^O, \mathcal{A}_c^H, \rightarrow_c \rangle$ denote an arbitrary context. Two simple governing policies $g_1 = \langle o_1, e, \psi_1 \rangle \bullet a_1$ and $g_2 = \langle o_2, e, \psi_2 \rangle \bullet a_2$ are equivalent with respect to T_c, denoted by $g_1 \overset{T_c}{\equiv} g_2$, if for all $t = (s_1, \alpha, s_2) \in \rightarrow_c$,*

(i) $t \models \tau(g_1, g) \Leftrightarrow t \models \tau(g_2, g)$ where $\tau(g_i, g), i = 1, 2,$ indicates the triggering conditions of g_i and g denotes the governing policy set of manager [9].

(ii) $\Theta_{s_2}(a_1) = \Theta_{s_2}(a_2)$

To reason about the behavioral equivalence of governing policy sets, we formulate the behavior of a governing policy set as a CA^a term. Then, we use the axiom system of CA^a_Θ to check the equivalence of corresponding action terms. Therefore, context-specific behavioral equivalence of two governing policy sets is reduced to checking context-specific behavioral equivalence of their corresponding action terms.

Definition 5. *Let g and g' indicate two arbitrary governing policy sets and $T_c = \langle V, S_c, s_c^0, \mathcal{A}_c^I, \mathcal{A}_c^O, \mathcal{A}_c^H, \rightarrow_c \rangle$ be an arbitrary context. The function $\Psi(g, t)$ returns the action term due to the enforcement of the governing policy set g, when the transition t occurs [9]. We say g and g' are equivalent with respect to T_c, denoted by $g \overset{T_c}{\equiv} g'$, iff for all $t = (s_1, \alpha, s_2) \in \rightarrow_c$, $\Theta_{s_2}(\Psi(g, t)) = \Theta_{s_2}(\Psi(g', t))$.*

Example 1. Suppose a situation that $relay_1$ becomes overloaded. The messages of a number of the surveyors which are transmitted by this relay should be transmitted through the low-loaded $relay_2$. To this end, first we have to find those

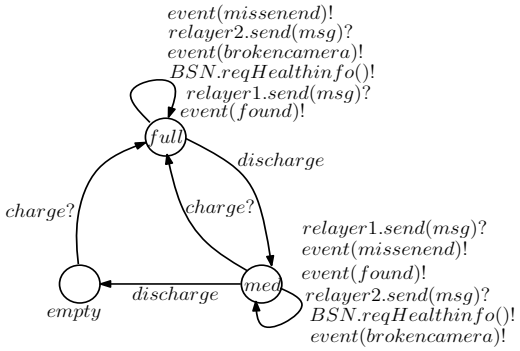

Fig. 4. The context of $surveyor_1(T_{surv})$

surveyors which transmit their messages through $relay_1$. Assume that the surveyors communicate with the relays only in the case that the wounded person is found. In order to check if a surveyor transmits its messages through $relay_1$, we check context-specific behavioral equivalence of its governing policies for transmitting data, and the simple governing policy $g_2 \stackrel{\text{def}}{=} \langle 1,\ found(x,y),\ \top \rangle \bullet\ a_2$ where $a_2 \stackrel{def}{=} BSN.reqHealthinfo()?\ \parallel relay1.send(msg)!$. This simple policy states that when the wounded person is found, the information should be transmitted through $relay_1$.

Suppose $surveyor_1$ has the simple governing policy $g_1 \stackrel{\text{def}}{=} \langle 1,\ found(x,y),\ \top \rangle \bullet\ a_1$ where

$$a_1 \stackrel{def}{=} BSN.reqHealthinfo()? \parallel (lowenergy :\rightarrow relay2.send(msg)! \\ +\neg lowenergy :\rightarrow relay1.send(msg)!)$$

Figure 4 shows the abstract context of $surveyor_1$ (T_{surv}) which has three states $full$ (full energy), med (medium energy) and $empty$ (no energy). Furthermore, $\{found, brokencamera, missionend\} \subseteq \mathcal{A}_c^O$ indicate the event set, $charge$ is an input action, and $discharge$ is an internal action. We should check context-specific behavioral equivalence of g_1 and g_2 with respect to T_{surv}. Both policies are triggered in the states $full$ and med in which the condition $lowenergy$ does not hold and event $found$ is activated. Hence, we must check the equations $\Theta_{full}(a_1) = \Theta_{full}(a_2)$ and $\Theta_{med}(a_1) = \Theta_{med}(a_2)$ (Definition 4). For the sake of readability, let's denote $relay1.send(msg)$ and $relay2.send(msg)$ by α_1 and α_2 respectively. According to the axiom systems in Table 1 and Figure 3,

$$\Theta_{full}(\neg lowenergy :\rightarrow \alpha_1! + lowenergy :\rightarrow \alpha_2!) \stackrel{\mathbf{TA2}}{=} \\ \Theta_{full}(\neg lowenergy :\rightarrow \alpha_1!) + \Theta_{full}(lowenergy :\rightarrow \alpha_2!) \stackrel{\mathbf{TA3,A3}}{=} \\ \top :\rightarrow \alpha_1 + \bot :\rightarrow \alpha_2 + \\ \Theta_{med}(\neg lowenergy :\rightarrow \alpha_1! + lowenergy :\rightarrow \alpha_2!) \stackrel{\mathbf{C_1,C_2}}{=}$$

$$\alpha_1 + \Theta_{med}(\neg lowenergy :\rightarrow \alpha_1! + lowenergy :\rightarrow \alpha_2!) \quad (1)$$

and

$$\Theta_{full}(\alpha_1!) = \alpha_1 + \Theta_{med}(\alpha_1!) \quad (2)$$

It is trivial to prove that

$$\Theta_{med}(\neg lowenergy :\rightarrow \alpha_1! + lowenergy :\rightarrow \alpha_2!) = \Theta_{med}(\alpha_1!) \quad (3)$$

and subsequently, the equation $\Theta_{full}(a_1) = \Theta_{full}(a_2)$ is concluded from equations (1)-(3). According to Theorem 1, the actions of g_1 and g_2 are statebased bisimilar with respect to the context T_{surv}, if they are activated in state $full$. Similarly, we can prove that the actions of g_1 and g_2 are statebased bisimilar when they are activated in state *med*. We conclude that g_1 and g_2 are equivalent according to Definition 4. Therefore, $surveyor_1$ always transmits its data through $relay_1$.

6.3 Context-specific Behavioral Equivalence of Adaptation Policies

Informally, two simple adaptation policies are equivalent if and only if (a) they are activated by the same transitions of the context, (b) their enforcement leads to switching to the identical adaptation modes and configurations, and (c) the manager switches to the new configuration in the same set of context states:

Definition 6. *Suppose* $T_c = \langle V, S_c, s_c^0, \mathcal{A}_c^I, \mathcal{A}_c^O, \mathcal{A}_c^H, \rightarrow_c \rangle$ *be an arbitrary context. Two adaptation policies* $p_1 = \langle o_1, e_1, \psi_1, \lambda_1, \phi_1 \rangle \bullet c_1$ *and* $p_2 = \langle o_2, e_2, \psi_2, \lambda_2, \phi_2 \rangle \bullet c_2$ *are equivalent with respect to* T_c*, denoted by* $p_1 \overset{T_c}{\equiv} p_2$*, if for all transitions* $t = (s_1, \alpha, s_2) \in \rightarrow_c$*,*

(i) $t \models \tau(p_1, p) \Leftrightarrow t \models \tau(p_2, p)$ *where* $\tau(p_i, p), i = 1, 2$*, gives the triggering conditions of* p_i *and* p *is the adaptation policy of the manager,*
(ii) $c_1 = c_2$ *and* $\lambda_1 = \lambda_2$*,*
(iii) $s' \models \phi_1 \Leftrightarrow s' \models \phi_2$ *for all reachable states* $s' \in S_c$ *from* s_2*, where there is a path such as* σ *between* s_2 *and* s'*, and for all* $s'' \in \sigma$*,* $s'' \not\models \phi_1 \vee \phi_2$ *.*

Similar to governing policies, enforcement of adaptation policies leads to a sequence of actions carried out by the manager. We say two adaptation policies are equivalent with respect to context T_c, if their enforcement in a system with context T_c leads to the same sequence of actions carried out by the manager. We introduce the operator Ω which gives the actions done by the manager to apply an adaptation policy. Let p indicate an adaptation policy of a manager, and $p_i = \langle o, e, \psi, \lambda, \phi \rangle \bullet c$ denote an arbitrary simple adaptation policy of p, i.e. $p = p_i \oplus p'$. The function $\Omega(p_i, p)$ returns a CA^a term due to enforcing p_i, where $\tau(p_i, p)$ denotes the triggering conditions of p_i:

$$\Omega(p_i, p) = \begin{cases} \tau(p_i, p) :\rightarrow event(e)?;\ tostrict();\ \phi :\rightarrow switch(c) & \lambda = \top \\ \tau(p_i, p) :\rightarrow event(e)?;\ toloose();\ \phi :\rightarrow switch(c) & \lambda = \bot \\ \delta_a & p_i = \delta_p \end{cases}$$

The action *tostrict*() denotes an internal action performed by the manager to evolve to the strict adaptation mode, *toloose*() denotes an internal action for evolving to loose adaptation mode, and $switch(c)$ is an internal action for switching to configuration c. Furthermore, the behavior of an adaptation policy $p = p_1 \oplus ... \oplus p_n$ is defined as follows:

$$\Omega(p) = \sum_{1 \leq i \leq n} \Omega(p_i, p)$$

Given the behavior of adaptation policies as CA^a terms, we use context-specific behavioral equivalence theory of CA^a_Θ to reason about their behavioral equivalence.

Definition 7. *Let p and p' indicate two arbitrary adaptation policy and $T_c = \langle V, S_c, s_c^0, \mathcal{A}_c^I, \mathcal{A}_c^O, \mathcal{A}_c^H, \rightarrow_c \rangle$ be an arbitrary context. We say p and p' are equivalent with respect to T_c, denoted by $p \overset{T_c}{=} p'$, iff for all $s \in S_c$, $\Theta_s(\Omega(p)) \overset{T_c}{=} \Theta_s(\Omega(p'))$.*

6.4 Context-specific Behavioral Equivalence of Configurations and Managers

A configuration consists of a set of governing policies and a set of adaptation policies. As mentioned above, we can change the configurations of a manager dynamically. Therefore, we require a theory to assure that the behavior of a configuration is equivalent to the behavior of a desired configuration, with respect to a context. In order to reason about the behavioral equivalence of two configurations, we reason about the behavioral equivalence of their governing policies as well as the behavioral equivalence of their adaptation policies:

Definition 8. *Let $c = \langle g, p \rangle$ and $c' = \langle g', p' \rangle$ be two arbitrary configurations, and $T_c = \langle V, S_c, s_c^0, \mathcal{A}_c^I, \mathcal{A}_c^O, \mathcal{A}_c^H, \rightarrow_c \rangle$ denote an arbitrary context. We say $c \overset{T_c}{=} c'$ iff $g \overset{T_c}{=} g'$ and $p \overset{T_c}{=} p'$.*

Example 2. Consider a situation that $surveyor_1$ breaks down, and should be replaced by another UAV with surveying capabilities, named UAV_2. Hence, we should check if the current configuration of UAV_2 (c') is equivalent to the configuration of $surveyor_1$(c) with respect to context T_{surv} shown in Figure 4. Suppose both $surveyor_1$ and UAV_2 have the same set of governing policies, i.e. $g \overset{T_{surv}}{=} g'$ where g indicates the governing policy set of $surveyor_1$ and g' denotes the governing policy set of UAV_2. Thus, we need to check the behavioral equivalence of their adaptation policies with respect to T_{surv}. Let p and p' indicate the adaptation policies of $surveyor_1$ and UAV_2, respectively, defined as follows:

$$\begin{aligned} p = {} & \langle 1, brokencamera, \neg lowenergy, \top, \top \rangle \bullet relayConf \oplus \\ & \langle 1, missionend, \top, \bot, \top \rangle \bullet idle \oplus \\ & \langle 1, found, lowenergy, \bot, \top \rangle \bullet idle \\ p' = {} & \langle 1, brokencamera, \top, \top, \top \rangle \bullet relayConf \oplus \\ & \langle 1, missionend, \top, \bot, \top \rangle \bullet idle \end{aligned}$$

We formulate p and p' in terms of CA^a terms as follows:

$$\begin{aligned}\Omega(p) = \neg lowenergy :\rightarrow event(brokencamera)?;\ tostrict();\ switch(relayConf) + \\ \top :\rightarrow event(missionend)?;\ toloose();\ switch(idleConf) + \\ \bot :\rightarrow event(found)?;\ toloose();\ switch(idleConf)\end{aligned}$$

$$\begin{aligned}\Omega(p') = \top :\rightarrow event(brokencamera)?;\ tostrict();\ switch(relayConf) + \\ \top :\rightarrow event(missionend)?;\ toloose();\ switch(idleConf)\end{aligned}$$

For the sake of readability, we show $\Omega(p)$ and $\Omega(p')$ by a and a', respectively. When the context is in state "$full$", if the events "$discharge$" and "$found$" are raised, non of the policies p and p' are triggered, however both policies are activated when the events "$brokencamera$" and "$missionend$" are raised:

$$\begin{aligned}\Theta_{full}(a) = \Theta_{full}(a') = event(brokencamera); \Theta_{med}(tostrict();\ switch(relayConf)) + \\ event(missionend); \Theta_{med}(toloose();\ switch(idleConf))\end{aligned}$$

It is trivial to prove that $\Theta_{med}(\Omega(p)) = \Theta_{med}(\Omega(p'))$ and $\Theta_{empty}(\Omega(p)) = \Theta_{empty}(\Omega(p'))$. Consequently, according to definition 7, it is concluded that $p \overset{T_c}{=} p'$. According to definition 8, we conclude that $surveyor_1$ and UAV_2 are substitutable,

$$\left.\begin{array}{l} p \overset{T_c}{=} p' \\ g \overset{T_c}{=} g' \end{array}\right\} \Rightarrow c \overset{T_c}{=} c'$$

Checking context-specific behavioral equivalence of two managers is the most important part of our behavioral equivalence theory. As mentioned above, a manager runs one of its configurations at a time, and switches between the configurations to perform dynamic adaptation. Informally, two managers are behavioral equivalent with respect to T_c iff (i) the managers have equivalent initial configurations with respect to T_c, and (ii) switching from equivalent configurations leads to the equivalent configurations in both managers. We reason about the equivalence of managers in terms of behavioral equivalence of their simple configurations:

Definition 9. *Let $m = \langle V_m, C, c_{init}\rangle$ and $m' = \langle V_{m'}, C', c'_{init}\rangle$ be two managers with configuration sets $C = \{c_1, ..., c_k\}$ and $C' = \{c'_1, ..., c'_{k'}\}$, initial configurations $c_{init} \in C$ and $c'_{init} \in C'$, and set of views V_m and $V_{m'}$, respectively. Furthermore, $T_c = \langle V, S_c, s_c^0, \mathcal{A}_c^I, \mathcal{A}_c^O, \mathcal{A}_c^H, \rightarrow_c\rangle$ indicates an arbitrary context. We say m and m' are equivalent with respect to context T_c, written by $m \overset{T_c}{=} m'$, (i) $c_{init} \overset{T_c}{=} c'_{init}$, (ii) for each equivalent configurations $c_i \in C$ and $c'_j \in C'$, if the manager m switches from c_i to c_k, the manager m' must switch from c'_j to $c'_l \in C'$ where $c_k \overset{T_c}{=} c'_l$ and vice versa.*

Example 3. Let $surveyor_1$ has the capability to search areas with chemical hazards. The manager of this UAV is defined as $survCntrlr = \langle V, \{survconf, hazardconf, relayconf\}, survconf\rangle$ where configuration $survconf$ is used to search areas without hazardous chemicals, $hazardconf$ is used to search areas with hazards, and $relayconf$ is used for acting as a relay. Let the situation that $surveyor_1$ has to be replaced by a UAV with the manager $survCntrlr' = \langle V, \{survconf', relayconf'\}, survconf'\rangle$. Let T_s denote the context of $survCntrlr$ and $survCntrlr'$. Assume we have $survconf \overset{T_s}{=} survconf'$ and $relayconf \overset{T_s}{=} relayconf'$, if the surveying area is not contaminated with hazardous chemicals. This is due to the fact that the adaptation policies of $survconf$ and $relayconf$ for switching to $hazardconf$ are not triggered, and switching is done between $survconf$ and $relayconf$. Therefore, according to Definition 9, we conclude $survCntrlr \overset{T_s}{=} survCntrlr'$. It worth mentioning that if we use theses two UAVs in another context, they might not behave equivalently.

7 Related Work

Although process algebra is used for structural adaptation (e.g. see [2], [12]), however to the best of our knowledge process algebraic approaches have not been used for behavioral adaptation. Zhang et al. [19] proposed a model-driven approach using Petri Nets for developing adaptive systems. They also presented a model-checking approach for verification of adaptive system [20, 18] in which an extension of LTL with "adapt" operator was used to specify the adaptation requirements. Furthermore, authors in [14, 1] used labeled transition systems to model and verify embedded adaptive systems. In [10], a generic classification of the policy conflicts are presented for PobSAM models and temporal patterns expressed in LTL are provided to detect each class of conflicts. We studied the comparison of existing work in the area of formal verification of adaptive systems and PobSAM in [8].

The issue of component substitutability has already been addressed in literature with the purpose of checking compatibility of a new component with other components (e.g. [11]), substituting a component with another component with the equivalent behavior (e.g. [17]), replacing a component such that the reconfigured system preserves a specific property etc. Some approaches specify the components behavior by modeling the components interfaces while a few approaches are concerned with specifying the internal behavior of components, as we have done in this work. We use specification of components interfaces to build and specify the context of managers. Among the existing approaches, [17] uses a formalism named component-interaction automata to specify the behavior of components interfaces. They define a notion of equivalence on these automata in addition to a composition operator to prove substitutability and independent implementability properties. In a similar work, [6] specifies the components behavior using max/plus automata and defines four kinds of substitutivity considering QoS aspects. However, the main difference compared to our work is that they do not consider data states of components. Furthermore, to the best

of our knowledge, none of the approaches for checking behavioral equivalence of components are based on a process algebraic formalism. The existing approaches model the components behavior using automata-based or LTS formalisms. Due to the fact that PobSAM has a different formal foundation, it requires special analysis techniques such as the equational theory presented in this paper which is not provided by other existing work.

Schaeffer-Filho et al. [13] use Alloy Analyzer [2] for formal specification and verification of policy-based systems, however they are not concerned with behavioral equivalence of components. Moreover, Georgas et al [4] uses policies as a mechanism for structural adaptation in robotic domain, but this work has no formal foundation.

8 Conclusions

In this paper, we presented an equational theory to analyze context-specific behavioral equivalence of policies, configurations and managers in PobSAM models based on the notion of statebased bisimilarity. Given the context as a labeled state transition system, we analyze context-specific behavioral equivalence of the manager layer independently from the actor layer. To this aim, we introduced and axiomatized an operator to consider interactions of the managers and the context. We demonstrated the approach using an example for search and rescue operations.

9 Acknowledgments

This work was funded by the NTH School for IT Ecosystems. NTH (Niedersachsische Technische Hochschule) is a joint university consisting of Technische Universitat Braunschweig, Technische Universitat Clausthal, and Leibniz Universitat Hannover.

References

1. Rasmus Adler, Ina Schaefer, Tobias SchLule, and Eric Vecchie. From model-based design to formal verification of adaptive embedded systems. In Proceedings of the formal engineering methods 9th international conference on Formal methods and software engineering, ICFEM'07, pages 76-95, Berlin, Heidelberg, 2007. Springer-Verlag.
2. Jeremy S. Bradbury, James R. Cordy, JLurgen Dingel, and Michel Wermelinger. A survey of self-management in dynamic software architecture specifications. In Proceedings of 1st ACM SIGSOFT Workshop on Self-managed Systems, pages 28-33. ACM, 2004.
3. David Garlan, Shang-Wen Cheng, and Bradley R. Schmerl. Increasing system dependability through architecture-based self-repair. In WADS, pages 61-89, 2002.
4. John C Georgas and Richard N Taylor. Policy-based self-adaptive architectures: a feasibility study in the robotics domain. In Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems, SEAMS'08, pages 105-112, New York, NY, USA, 2008. ACM.
5. Jan Friso Groote and Alban Ponse. Process algebra with guards: Combining hoare logic with process algebra. Formal Asp. Comput., 6(2):115-164, 1994.

6. Pierre-Cyrille Heam, Olga Kouchnarenko, and Jerome Voinot. Component simulation-based substitutivity managing qos aspects. Electron. Notes Theor. Comput. Sci., 260:109-123, January 2010.
7. Narges Khakpour. Context-based behavioral equivalence of components in self-adaptive systems. Technical report, Technical Report of TU Bruanschweig, 2011.
8. Narges Khakpour, Saeed Jalili, Carolyn L. Talcott, Marjan Sirjani, and Mohammad Reza Mousavi. Pobsam: Policy-based managing of actors in self-adaptive systems. Electr. Notes Theor. Comput. Sci., 263:129-143, 2010.
9. Narges Khakpour, Saeed Jalili, Carolyn L. Talcott, Marjan Sirjani, and Mohammad Reza Mousavi. Formal modeling of evolving adaptive systems. submitted, 2011.
10. Narges Khakpour, Ramtin Khosravi, Marjan Sirjani, and Saeed Jalili. Formal analysis of policy-based self-adaptive systems. In SAC, pages 2536-2543, 2010.
11. Fabrice Legond-Aubry, Daniel Enselme, and Gerard Florin. Assembling contracts for components. In Formal Methods for Open Object-based Distributed Systems (FMOODS-DAIS), Lecture Notes in Computer Science, pages 35-43, Paris, FR (France), November 2003. Springer-Verlag.
12. Radu Mateescu, Pascal Poizat, and Gwen Salaun. Adaptation of service protocols using process algebra and on-the-fly reduction techniques. IEEE Transactions on Software Engineering, 99(PrePrints), 2011.
13. Alberto Schaeffer-Filho, Emil Lupu, Morris Sloman, and Susan Eisenbach. Verification of policy-based self-managed cell interactions using alloy. In Proceedings of the 10th IEEE international conference on Policies for distributed systems and networks, POLICY'09, pages 37-40. IEEE Press, 2009.
14. Klaus Schneider, Tobias Schuele, and Marion Trapp. Verifying the adaptation behavior of embedded systems. In Proceedings of the 2006 international workshop on Self-adaptation and self-managing systems, SEAMS06, pages 16-22, New York, NY, USA, 2006. ACM.
15. Marjan Sirjani, Ali Movaghar, Amin Shali, and Frank S. de Boer. Modeling and verification of reactive systems using rebeca. Fundam. Inform., 63(4):385-410, 2004.
16. Morris Sloman and Emil C. Lupu. Engineering policy-based ubiquitous systems. Comput. J., 53(7):1113-1127, 2010.
17. Ivana Cerna, Pavlna Varekova, and Barbora Zimmerova. Component substitutability via equivalencies of component-interaction automata. Electron. Notes Theor. Comput. Sci., 182:39-55, June 2007.
18. Ji Zhang and Betty H. C. Cheng. Specifying adaptation semantics. ACM SIGSOFT Software Engineering Notes, 30(4):1-7, 2005.
19. Ji Zhang and Betty H. C. Cheng. Model-based development of dynamically adaptive software. In Proceedings of the 28th international conference on Software engineering, ICSE'06, pages 371-380, New York, NY, USA, 2006. ACM.
20. Ji Zhang, Heather Goldsby, and Betty H. C. Cheng. Modular verification of dynamically adaptive systems. In Proceedings of the 8th ACM international conference on Aspect-oriented software development, pages 161-172, 2009.