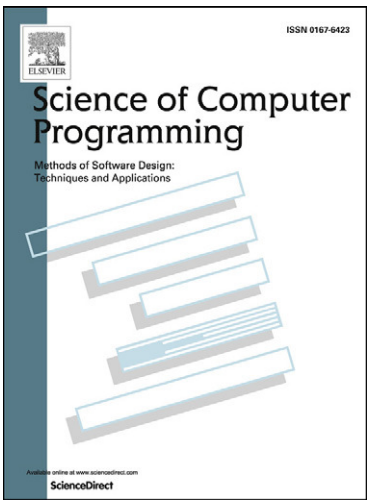
Adaptability checking in complex systems

Emanuela Merelli, Nicola Paoletti, Luca Tesei



PII: S0167-6423(15)00065-9
DOI: http://dx.doi.org/10.1016/j.scico.2015.03.004
Reference: SCICO 1882

To appear in: *Science of Computer Programming*

Received date: 17 February 2013
Revised date: 16 December 2014
Accepted date: 26 March 2015

Please cite this article in press as: E. Merelli et al., Adaptability checking in complex systems, *Sci. Comput. Program.* (2015), http://dx.doi.org/10.1016/j.scico.2015.03.004

Highlights

- Hierarchical model for multi-level adaptive systems.
- Relational characterisation of strong and weak adaptability.
- Adaptability checking is reduced to a CTL model checking problem.
- Application to the case study of ATVs motion control

Adaptability Checking in Complex Systems

Emanuela Merelli[a,*], Nicola Paoletti[b], Luca Tesei[a]

[a] *School of Science and Technology, Computer Science Division, University of Camerino, Via del Bastione 1, 62032, Camerino, Italy*
[b] *Department of Computer Science, University of Oxford, Parks Road, OX1 3QD, Oxford, UK*

Abstract

A hierarchical approach for modelling the adaptability features of complex systems is introduced. It is based on a structural level S, describing the adaptation dynamics of the system, and a behavioural level B accounting for the description of the admissible dynamics of the system. Moreover, a unified system, called $S[B]$, is defined by coupling S and B. The adaptation semantics is such that the S level imposes structural constraints on the B level, which has to adapt whenever it no longer can satisfy them. In this context, we introduce weak and strong adaptability, i.e. the ability of a system to adapt for some evolution paths or for all possible evolutions, respectively. We provide a relational characterisation for these two notions and we show that adaptability checking, i.e. deciding if a system is weak or strong adaptable, can be reduced to a CTL model checking problem. We apply the model and the theoretical results to the case study of a motion controller of autonomous transport vehicles.

Keywords: adaptive systems, state machine, adaptability relations, adaptability checking, S[B] model
2000 MSC: 68Q10, 68Q60

*Corresponding author
Email addresses: emanuela.merelli@unicam.it (Emanuela Merelli), nicola.paoletti@cs.ox.ac.uk (Nicola Paoletti), luca.tesei@unicam.it (Luca Tesei)

1. Introduction

Self-adaptive systems are particular systems able to modify their own behaviour according to their current configuration and the perception of the environment in which they operate. They develop new strategies in order to fulfil an objective, properly respond to changes of the environmental conditions or, more generally, maintain desired conditions.

From a broad viewpoint, self-adaptiveness is an intrinsic property of complex natural systems. Self-adaptation is a process driving both the evolution and the development of living organisms that adapt their features and change their phenotype in order to survive to the current habitat, to achieve higher levels of fitness and to appropriately react to external stimuli.

Nowadays, software systems are increasingly resembling complex systems; this motivates the development of methods for enabling software self-adaptiveness. Similarly to natural systems, *"Self-adaptive software evaluates its own behaviour and changes behaviour when the evaluation indicates that it is not accomplishing what the software is intended to do, or when better functionality or performance is possible"* [35]. Self-adaptive software finds application in fields like autonomic computing, service-oriented architectures, pervasive service ecosystems, mobile networks, multi-agent systems, and ultra-large-scale software systems [25], characterised by distributed, autonomous, interacting, heterogeneous, conflicting and evolvable sub-systems.

In this work, we develop a formal hierarchical model for self-adaptive systems, where two fundamental levels are defined: the *behavioural level* B, which describes the admissible dynamics of the system; and the *structural level* S, accounting for the invariant features of the system that regulate the behaviour of the system. More precisely, both levels are modelled as state machines, but each state of the S level is associated with a set of *constraints*, i.e. logical formulas over observable variables of the B level.

An S state in the structural level represents a relatively persistent situation, a steady region for the B level, identified by the set of B states satisfying the constraints. Therefore S is a *higher order structure* whose states can be interpreted as sets of B states, and whose transitions can be viewed as mappings among sets of B states. The coupled model will be referred to as $S[B]$, in order to highlight the two basic levels that compose the system.

The $S[B]$ model is broadly inspired by a previous work of some of the authors, a spatial bio-inspired process algebra called *Shape Calculus* [3, 4], where processes are characterised by a reactive behaviour B and by a shape

S that imposes a set of geometrical constraints on their interactions and occupancy in the three-dimensional Euclidean space. Here, instead, the computational approach is shifted in a more general context, where S and B are coupled by a hierarchical relation defined on the structural constraints of the S level and the state space of the B level.

The adaptation semantics can be briefly described as follows. Consider an $S[B]$ system, let q be the current state of B and assume that it satisfies the constraints of the current S state. The adaptation is triggered whenever q cannot evolve because there is no next B state that satisfies the current S constraints. During adaptation, the $S[B]$ system attempts to evolve towards a new B state that satisfies a new S state, chosen among the successors of the current one. In this phase B is no more constrained by S. Adaptation terminates successfully when B ends up in a state that fulfils the new global situation represented by one of the admissible S states.

A first general introduction of the $S[B]$ model was given in [38] and [39] by the authors. In this work, we provide several novelties and improvements, most of them devoted to the *adaptability checking problem*, i.e. the automatic checking of the adaptation capabilities of a given $S[B]$ system. In particular, we define the notion of *weak adaptability*, possessed by an $S[B]$ system that is able to adapt along some of all its possible evolution paths. *Strong adaptability* requires that the $S[B]$ system is able to adapt along all its possible evolution paths. We formulate the notions of weak and strong adaptability as relations between the states of B and S and also in logical form, as Computation Tree Logic (CTL) formulas over the given semantics of an $S[B]$ system. Then, we formally prove the equivalence between the relational and the logical formulation of strong and weak adaptability (Theorems 1 and 2), showing that the adaptability checking problem can be reduced to a classical model checking problem. We also discuss the computational complexity of our approach. The effectiveness of the $S[B]$ approach for self-adaptive systems is demonstrated using a case study in the context of adaptive software systems, a model for a motion controller of autonomous transport vehicles in a smart airport. The same case study is used for exemplifying the notions of weak and strong adaptability and the adaptability checking problem.

The paper is organized as follows. Section 2 introduces the formalism and the syntax of the $S[B]$ model. Section 3 illustrates the application of the model to the example of adaptive motion controller. In Section 4 we give the operational semantics of an $S[B]$ system by means of a flattened transition

system. In Section 5 we formalise the relations of weak and strong adaptation, which we equivalently characterise as CTL formulas in Section 6. Related works including adaptation features of $S[B]$, and conclusions are given in Section 7. Finally, proofs are presented in Appendix A.

2. A Formal Hierarchical Model for Adaptive Systems

In the S[B] approach, a model encapsulates both the behavioural (B) and the structural (S) level of an adaptive system. The behavioural level is classically described as a finite state machine of the form $B = (Q, q_0, \rightarrow_B)$ where Q is a set of B states, q_0 initial B state and $\rightarrow_B$ transition relation. The structural level is modelled as a finite state machine $S = (R, r_0, \mathcal{O}, \rightarrow_S, L)$ where R is a set of S states, r_0 is the initial S state, $\mathcal{O}$ is an observation function, $\rightarrow_S$ is a transition relation and L is a state labelling function. The function L labels each S state with a formula representing a set of constraints over an *observation* of the B states. Therefore an S state r can be directly mapped to the set of B states satisfying $L(r)$. Through this mapping, S can be viewed as a *second-order* structure $S = (R \subseteq 2^Q, r_0, \mathcal{O}, \rightarrow_S \subseteq R \times R)$ where each S state r is identified with its corresponding set of B states. An $S[B]$ system is the result of coupling the two levels S and B through the observation function $\mathcal{O}$ and an evaluation function $[[\cdot]]$, that maps each S state to the set of B states meeting the constraints.

The $S[B]$ adaptation is achieved by switching from an S state to another S state where a different set of constraints holds. During adaptation the B machine is no more regulated by the structural level, except for an adaptation invariant, called *adaptation invariant*, that must be fulfilled by the system while adapting. The system adapts by following any trajectory that is present at the B level that does not violate the invariant condition, which can be used as a safety condition if some activities of the B level must be avoided during adaptation.

In order to realise our notion of $S[B]$ adaptiveness, there must be some information flowing both from B to S and viceversa. In particular, the information from B to S is modelled here as a set of variables $A = \{a_1, \ldots, a_n\}$ called *observables* of the S on the B level. The values of these variables must always be *derivable* from the information contained in the B states. This makes our approach black-box-oriented, that is to say, S has not the full knowledge of B, but only some derived information.

In control-theoretic terms, as illustrated in Fig. 1, the adaptation model of an $S[B]$ system can be viewed as a closed-loop system where B is the plant and S is the controller. Let q and r be the current states of B and S. B outputs the vector[1] $\mathbf{B} = Post(q)$ of the states reachable from q with a single transition, i.e. an element q_i of $\mathbf{B}$ is such that $q \rightarrow_B q_i$ and each state is unique: $\bigwedge_{i \neq j} q_i \neq q_j$. Since S can only observe some features of B states, the observer $\mathcal{O}$ will provide S with a vector of observations made over the values of the variables characterizing each state in $\mathbf{B}$: $\mathbf{o} = \prod_i \mathcal{O}(q_i)$.

S will check the observables provided by $\mathbf{o}$ with respect to its current state r. If its constraints are satisfied, S will remain in the same state and $S[B]$ will proceed in *steady* mode. Otherwise, S will perform a transition to a new r state forcing the $S[B]$ system to enter an *adaptation* mode. Here we assume that the updated r is computed with a function *Check* that takes the current S state and observations. The feedback loop closes with the selection of the eligible next states $\mathbf{B}$ outputted to B, i.e. those states that satisfy the current constraints of r or those that satisfy the current adaptation invariant. The set $\mathbf{B}$ is obtained by applying the evaluation function $[[\cdot]]$ to either the constraints of r or the invariant. In the adaptation mode the output is calculated using the whole B machine without constraints except the adaptation invariant, to make B free to explore the state space. In turn, B updates its current state q by selecting one of the eligible states provided in $\mathbf{B}$. The concepts of observation function $\mathcal{O}$ and evaluation function $[[\cdot]]$ are formalized in Definition 1 and Definition 2.

2.1. Language for constraints

The constraints characterising the states of an S level are expressed using formulas of a many-sorted first order logic. More precisely, the definition of an S level includes the definition of a many-sorted signature Σ containing some function symbols, some predicate symbols and some sorts $D_1, \ldots, D_k$. Σ-terms and Σ-formulas are constructed in the standard way [24]. In addition, a particular set of sorted variables, which we call observables, must be fixed. Such a set is of the form $A = \{a_1 \colon D_{j_1}, \ldots, a_n \colon D_{j_n}\}$, where $j_i \in \{1, \ldots, k\}$ for all $i = 1, \ldots, n$. Then, constraints can be expressed as Σ-formulas ψ such that the variables that occur free in ψ, denoted by free(ψ), are a (possibly empty)

[1] With abuse of notation, we allow the *Post* operator to return an indexed vector of states instead of a set.

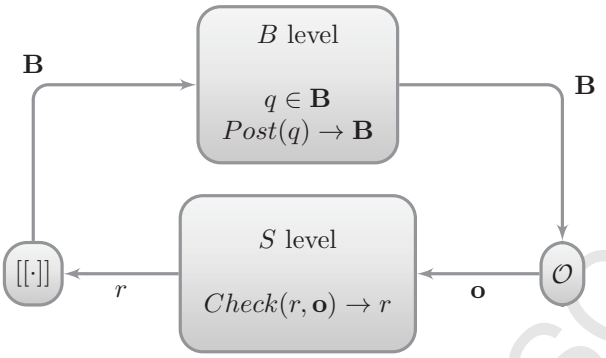


Figure 1: Adaptation loop in an $S[B]$ system. At each step, S observes B and decide if there is a need to change state. The loop closes with the selection of eligible next states $\mathbf{B}$ of B, i.e. those that satisfy the current constraints or those that satisfy the current adaptation invariant.

subset of A. This set will be denoted by $\Psi(\Sigma, A) = \{\psi \mid \psi \text{ is a } \Sigma\text{-formula} \wedge \text{free}(\psi) \subseteq A\}$.

We also impose that a particular structure M is fixed for the evaluation of Σ-formulas. M consists of k non-empty domains $M(D_1), \ldots, M(D_k)$, as carrier sets for sorts, together with interpretations for all function and predicate symbols of Σ. To obtain the full semantic evaluation of formulas in $\Psi(\Sigma, A)$ we will take the values for the free variables in A from an observation function.

Definition 1 (Observation Function)**.** *Let $\mathcal{Q}$ be the universe set of all states of machines possibly representing B levels. Let Σ be a many-sorted signature, let $A = \{a_1 \colon D_{j_1}, \ldots, a_n \colon D_{j_n}\}$ be a set of observables and let M be a structure for the evaluation of Σ-formulas. An* observation function $\mathcal{O}_M^{\Sigma,A}$ *on Σ, A and M is a partial function*

$$\mathcal{O}_M^{\Sigma,A} \colon \mathcal{Q} \hookrightarrow (A \to \mathcal{D})$$

where (i) $\mathcal{D} = \bigcup_{i=1}^{n} M(D_{j_i})$ and (ii) for any state $q \in \mathcal{Q}$, if $\mathcal{O}_M^{\Sigma,A}(q) \neq \perp$ then $\mathcal{O}_M^{\Sigma,A}(q)(a_i \colon D_{j_i}) \in M(D_{j_i})$, for all $i = 1, \ldots, n$. For a lighter notation, we will use $\mathcal{O}$ instead of $\mathcal{O}_M^{\Sigma,A}$ when Σ, A and M are clear from the context.

Note that the use of the universe of states as domain makes the definition of the observation function independent from a particular state machine representing a behavioural level B. Note also that we do not require the observation function to be injective. This means that some different states can

give the same values to the observables. In this case, the difference among the states is not visible to S through the observation, but it is internal to B.

To complete the machinery for checking whether a set of constraints is satisfied or not, we define the satisfaction relation in the natural way.

Definition 2 (Satisfaction relation)**.** *Let $\mathcal{O}_M^{\Sigma,A}$ be an observation function. A state $q \in \mathcal{Q}$* satisfies *a formula $\psi \in \Psi(\Sigma, A)$, written $q \models \psi$, iff $\mathcal{O}_M^{\Sigma,A}(q) \neq \perp$ and ψ is true, according to the standard semantics of many-sorted first order logic, with respect to the structure M and by substituting in ψ every occurrence of the free sorted variable $a_i : D_{j_i}$ with $\mathcal{O}_M^{\Sigma,A}(q)(a_i : D_{j_i})$.*

We also define an evaluation function $[[\cdot]] : \Psi(\Sigma, A) \to 2^{\mathcal{Q}}$ mapping a formula $\psi \in \Psi(\Sigma, A)$ to the set of states $[[\psi]] = \{q \in \mathcal{Q} \mid q \models \psi\}$, i.e. those satisfying ψ.

A set of constraints is formally expressed by a formula $\psi \in \Psi(\Sigma, A)$ that is the conjunction of all the formulas representing each constraint in the set. The set of constraints is satisfied if and only if the corresponding formula is true in the fixed structure M and observation $\mathcal{O}_M^{\Sigma,A}$.

Example 1: Let us consider a set of observables and associated sorts:

$$A = \{velocity : \mathbb{R},\ congestion : \mathbb{B}\}$$

Consider also a signature $\Sigma = \{\mathbb{R}, \mathbb{B}, ==, >, <, 0, 5\}$ where $\mathbb{R}$ and $\mathbb{B}$ are the sorts indicating the domains of real numbers and boolean, respectively; $==$ is the equality predicate interpreted as the identity relation in each domain; $>$ and $<$ are the usual greater-than and less-than predicates over $\mathbb{R}$; and the constants 0 and 5 are the real numbers 0 and 5. A possible formula ψ in the language $\Psi(\Sigma, A)$ is

$$congestion \implies velocity < 5 \ \wedge\ \neg congestion \implies velocity > 0$$

whose satisfaction depends on the particular values of the variables, which will be different in different states.
In the context of Autonomous Transport Vehicles (see Section 3), this formula can be thought to represent a set of two constraints, one imposing that "in case of congestion, the velocity of the vehicle must be lower than 5" and the other that "in normal traffic conditions, the velocity must be greater than 0".

2.2. Coupling S and B

Let us now formally define the behavioural level B and the structural level S separately. Afterwards, the $S[B]$ model is defined as the combination of the two.

Definition 3 (Behavioural level)**.** *The behavioural level of a system is a tuple* $B = (Q, q_0, \rightarrow_B)$*, where*

- $Q \subseteq \mathcal{Q}$ *is a* finite *set of states and* $q_0 \in Q$ *is the initial state; and*
- $\rightarrow_B \subseteq Q \times Q$ *is the transition relation.*

Definition 4 (Structural Level)**.** *The structural level of a system is a tuple* $S = (R, r_0, \mathcal{O}_M^{\Sigma,A}, \rightarrow_S, L)$*, where*

- R *is a finite set of states and* $r_0 \in R$ *is the initial state;*
- $\mathcal{O}_M^{\Sigma,A}$ *is an observation function on a signature* Σ*, a set of observables* A *and a structure* M*;*
- $\rightarrow_S \subseteq R \times \Psi(\Sigma, A) \times R$ *is a finite transition relation, labelled with a formula called* invariant*; and*
- $L : R \rightarrow \Psi(\Sigma, A)$ *is a function labelling each state with a formula representing* a set of constraints*.*

Let us now give an intuition of the adaptation semantics. Let the current S state be r_i and suppose $r_i \xrightarrow{\psi}_S r_j$ for some r_j. Assume that the behaviour is in a steady state (i.e. not adapting) q_i and therefore $q_i \models L(r_i)$. If the B state can move, but all B transitions $q_i \rightarrow_B q_j$ are such that $q_j \not\models L(r_i)$, then the system can start adapting to the target S state r_j. In this phase, the B level is no more constrained, but during adaptation the invariant ψ must be met. Adaptation ends when the behaviour reaches a state q_k such that $q_k \models L(r_j)$.

We want to remark that the model supports the non-deterministic choice between adaptations, i.e. the system can adapt to every target state r_j reachable with a transition $r_i \xrightarrow{\psi}_S r_j$ from the current r_i state. The non-determinism can be both external - that is different target states can be reached by satisfying different invariants - and internal - that is different target states can be reached satisfying the same invariant condition.

Definition 5 ($S[B]$ system)**.**
An $S[B]$ system *is the combination of a behavioural level* $B = (Q, q_0, \rightarrow_B)$ *and a structural level* $S = (R, r_0, \mathcal{O}_M^{\Sigma,A}, \rightarrow_S, L)$ *such that for all* $q \in Q$, $\mathcal{O}_M^{\Sigma,A}(q) \neq \perp$*. Moreover, in any* $S[B]$ *system the initial* B *state must satisfy the constraints of the initial* S *state, i.e.* $q_0 \models L(r_0)$*.*

3. Case Study: Adaptive Motion Controller of Autonomous Transport Vehicles

In this section, we illustrate the features of our approach by means of an example adapted from [29]: a model of a motion controller of *Autonomous Transport Vehicles (ATVs)* in a smart airport.

ATVs are responsible for the transport of passengers between stopovers like passenger entrances, check-in desks, departure gates, and plane parking positions. In the airport there are two types of roads that the ATVs can use: main roads and secondary roads, the latter ones used during traffic peaks. ATVs can travel at different speed in any road, preferably not at the maximum speed on secondary roads, as they are narrower than the main ones. For simplicity, we model only the subcomponent of ATVs that controls the vehicle speed and the switching between main and secondary roads. The $S[B]$ approach will be used to specify and implement the adaptation features of this controller in case of traffic congestion or blockages.

Behavioural Level

The behavioural level describes all the capabilities of the ATV controller, i.e. all the actions that the system is able to do in an unconstrained scenario. We suppose that each ATV controller has the possibility to perceive the current situation of traffic congestion or blockage by using a sensor or by communicating with a global monitoring systems of the airport. The ATV controller may, based on this perception, decide which action to execute.

The usual way of abstractly specifying a behaviour of this kind is a two-phase cycle: in the first phase there is the *perception* of the global environment, only relatively to the current congestion situation; in the second phase, a local *action* can be executed, corresponding to change either the vehicle velocity or the road to drive.

To formalise the behavioural level B we consider the following set of observable variables and associated sorts:

- $r : \{M \text{ (main)}, S \text{ (secondary)}\}$, the current road;

- $v : \{0\ (\text{slow}), 1\ (\text{medium}), 2\ (\text{high})\}$, the current velocity of the vehicle;
- $c : \{\textit{true}\ (\text{congestion}), \textit{false}\ (\text{no congestion})\}$, a boolean variable indicating the current knowledge of the ATV controller about the traffic congestion;
- $p : \{\textit{true}\ (\text{perception}), \textit{false}\ (\text{no perception})\}$, a boolean variable indicating whether or not the ATV controller is currently perceiving the congestion situation, i.e. it is in the first phase of its cycle; and
- $a : \{\textit{true}\ (\text{action}), \textit{false}\ (\text{no action})\}$, a boolean variable indicating whether or not the ATV controller is currently executing an action, i.e. it is in the second phase of its cycle.

Hereafter, each state q of the B level will be identified by the values of the observables, $q = (r, v, c, p, a)$. We will denote the boolean value *true* with the integer 1 and the boolean value *false* with the integer 0.

Figure 2 contains a schema representing the portion of the state machine, corresponding to the B level, starting from a generic state $(r, v, c, 0, 0)$, i.e. a state in which the controller is at the beginning of one iteration of its cycle. The two outgoing transitions non-deterministically model the perception of the same congestion situation known at the last perception (state $(r, v, c, 1, 0)$) or of the new (negated) situation (state$(r, v, \neg c, 1, 0)$). In any case, the system proceeds to the action phase (states $(r, v, c, 0, 1)$ or $(r, v, \neg c, 0, 1)$) where it can decide to change road or to change velocity ending up in an updated state, ready to start another perception-action iteration.

Structural Level

The structural level can be used to give an abstract, constraint-based, specification of the possible ways (the various states in S) in which the system can function together with the admissible adaptations among them (the transitions in S). In this case study we show how to implement, as an $S[B]$ system, a policy of the smart airport that requires the adaptation of the behaviours of the vehicles during their functioning. We assume that some of the ATVs in the airport are equipped with adaptability capabilities in order to implement, as an example, the following policy: "*whenever there is congestion, the adaptive ATV will have to switch as soon as possible to a secondary road and to limit the velocity to the maximum value 1; on the contrary, whenever there is not congestion, the adaptive ATV will have to use a main road at any velocity*".

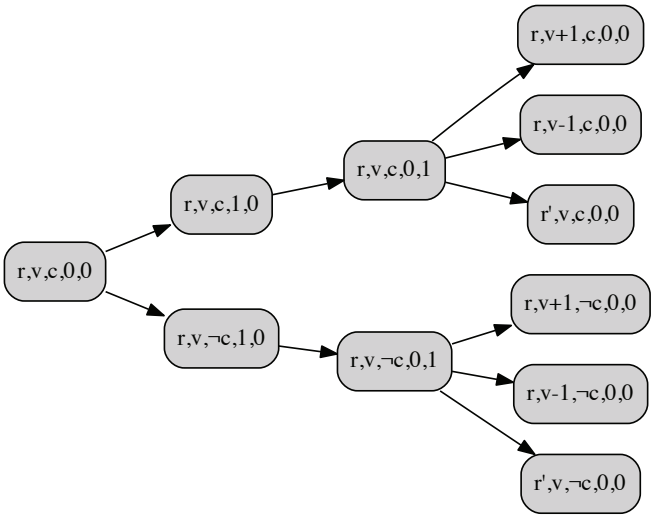


Figure 2: A schema of states and transitions starting from a generic state $(r, v, c, 0, 0)$. We let $r' \neq r$. Moreover, the transitions towards the states $(r, v+1, c, 0, 0)$ or $(r, v+1, \neg c, 0, 0)$ are not present whenever $v = 2$, while the transitions towards the states $(r, v-1, c, 0, 0)$ or $(r, v+1, \neg c, 0, 0)$ are not present whenever $v = 0$.

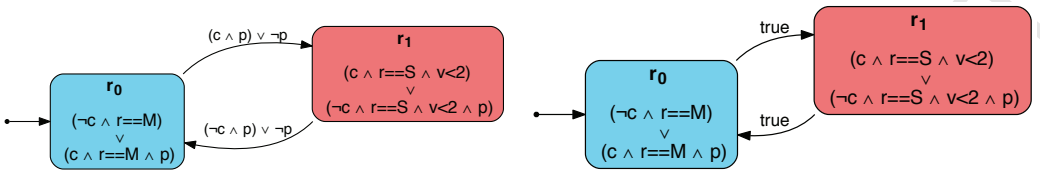
This policy clearly identifies two modes of operations of the adaptive ATVs, which are activated by different environmental conditions:

- a normal mode of operation, occurring when there is no traffic congestion and such that the main road is driven ($r == M$); and
- a fallback mode, occurring when congestion occurs; in this case the ATV has to be in the secondary road ($r == S$) and it cannot drive at the maximum velocity (v is either 0 or 1).

These two modes are modelled by two different S states, as shown in Figure 3(a). The constraint characterizing the normal mode (state r_0) is the disjunction of the description of the mode ($\neg c \wedge r == M$) with the possibility for the ATV controller to perceive the occurrence of a congestion when in the mode ($c \wedge r == M \wedge p$). Note that without the latter disjunctive term it would not be possible for the system to perceive the different value of the variable c because that would violate the constraint describing the mode ($\neg c \wedge r == M$), putting the perception state out of the normal mode. This according to the semantics of $S[B]$ (see Section 4). The fallback mode (state r_1) is also described, for the same reasons, by the disjunction of the mode description ($c \wedge r == S \wedge v < 2$) and the possibility for the system to perceive $\neg c$ when in the mode ($\neg c \wedge r == S \wedge v < 2 \wedge p$).

The invariant conditions on the transitions between r_0 and r_1 are needed to avoid the perception of a value of c different from that of the mode to which the system is adapting to. For instance, when adapting from r_0 to r_1 the target steady state is one in which there is congestion. During the adaptation we need to ensure that, whenever there is perception, only the value c can be perceived, ruling out the other possibility (due to non-determinism) of perceiving $\neg c$. This is expressed by the invariant $(p \wedge c) \vee \neg p$ that, in fact, corresponds to forcing the $S[B]$ system to ignore possible changes in the environment during adaptation. Without these invariants (see Figure 3(b)), the system could fall in a livelock during the adaptation phase, as it is shown in Section 5.1.

Figure 4 shows the full B level where the blue box encloses the states satisfying the constraints of r_0, and the red box those satisfying the constraints of r_1.



(a) S_0 (b) S_1

Figure 3: Two possible structural levels, S_0 (a) and S_1 (b), for the motion controller example. They model the adaptation logic between two modes of operation, r_0 (normal, blue) and r_1 (fallback, red).

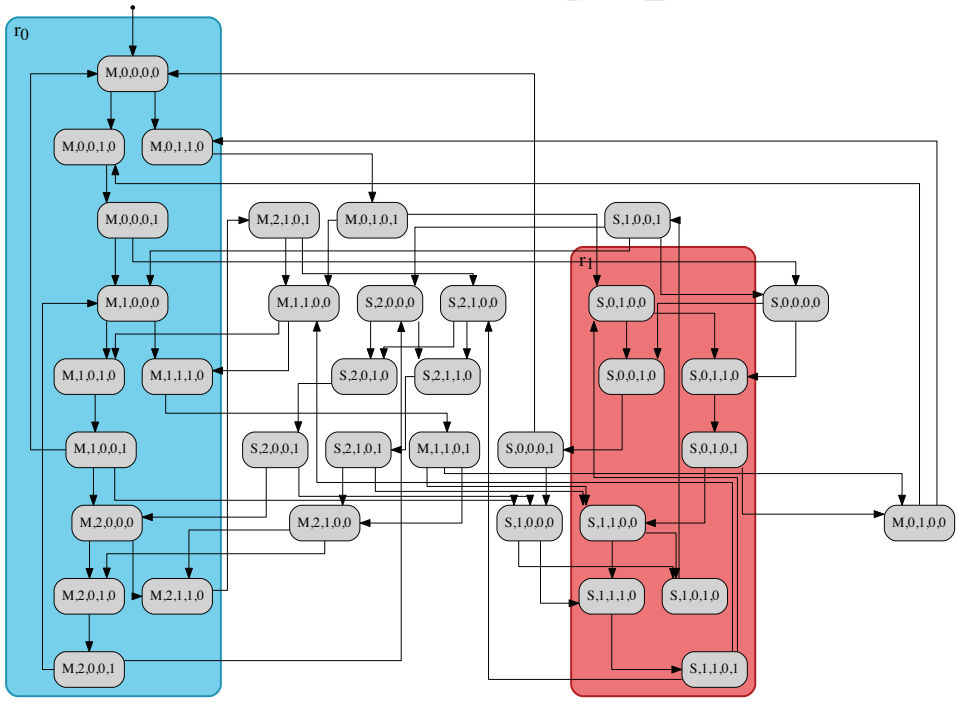


Figure 4: The behavioural state machine for the motion control example. Each state is labelled with the different evaluation of the observable variables (r, v, c, p, a), i.e. (road, velocity, congestion, perception, action). Coloured areas are used to represent the states of the S levels of Figure 3(a) and 3(b), which identify stable regions in the B level. These are r_0 (normal mode, blue box) and r_1 (fallback/congestion mode, red box).

4. Operational Semantics of the Flat $S[B]$ system

We give the operational semantics of an $S[B]$ system as a transition system resulting from the flattening of the behavioural and the structural levels. We obtain a Labelled Transition System (LTS) over states of the form (q, r, ρ), where:

- $q \in Q$ and $r \in R$ are the active B state and S state, respectively; and
- ρ keeps the target S state that must be reached during adaptation and the invariant that must be fulfilled during this phase. Therefore ρ is either empty (no adaptation is occurring), or a singleton $\{(\psi, r')\}$, with $\psi \in \Psi(\Sigma, A)$ a formula and $r' \in R$ an S state.

Definition 6 (Flat $S[B]$*system*)**.** *Consider an $S[B]$ system. The corresponding flat $S[B]$ system is the LTS $\mathcal{F}(S[B]) = (F, f_0, \xrightarrow{r} \cup \xrightarrow{r,\psi,r'})$ where*

- $F = Q \times R \times (\{(\psi, r') \mid \exists r \in R.\ r \xrightarrow{\psi}_S r'\} \cup \{\emptyset\})$ *is the set of states;*
- $f_0 = (q_0, r_0, \emptyset)$ *is the initial state;*
- $\xrightarrow{r} \subseteq F \times F$*, with $r \in R$, is a family of transition relations between non-adapting states, i.e., both satisfying $L(r)$;*
- $\xrightarrow{r,\psi,r'} \subseteq F \times F$*, with $r, r' \in R$ and $\psi \in \Psi(\Sigma, A)$, is a family of transition relations between adapting states, where the adaptation is determined by the S transition $r \xrightarrow{\psi}_S r'$; and*
- *the pairs in $\xrightarrow{r}$ and in $\xrightarrow{r,\psi,r'}$ are all and only those derivable using the rules in Table 1.*

Let us discuss the rules listed in Table 1 characterising the flattened transitional semantics:

- Rule Steady describes the steady (i.e. non-adapting) behaviour of the system. If the system is not adapting and a B state q can perform a transition to a q' that satisfies the current constraints $L(r)$, then the flat system can perform a non-adapting transition $\xrightarrow{r}$ of the form $(q, r, \emptyset) \xrightarrow{r} (q', r, \emptyset)$.

$$\textsc{Steady}\quad \frac{q \models L(r) \quad q \to_B q' \quad q' \models L(r)}{(q, r, \emptyset) \xrightarrow{r} (q', r, \emptyset)}$$

$$\textsc{AdaptStart}\quad \frac{\forall q''.(q \to_B q'' \implies q'' \not\models L(r)) \quad q \models L(r) \quad q \to_B q' \quad r \xrightarrow{\psi}_S r' \quad q' \not\models L(r') \quad q' \models \psi}{(q, r, \emptyset) \xrightarrow{r,\psi,r'} (q', r, \{(\psi, r')\})}$$

$$\textsc{Adapt}\quad \frac{\forall q''.(q \to_B q'' \implies q'' \not\models L(r')) \quad q \models \psi \quad q \not\models L(r') \quad q \to_B q' \quad q' \models \psi}{(q, r, \{(\psi, r')\}) \xrightarrow{r,\psi,r'} (q', r, \{(\psi, r')\})}$$

$$\textsc{AdaptEnd}\quad \frac{q \models \psi \quad q \not\models L(r') \quad q \to_B q' \quad q' \models L(r')}{(q, r, \{(\psi, r')\}) \xrightarrow{r,\psi,r'} (q', r', \emptyset)}$$

$$\textsc{AdaptStartEnd}\quad \frac{\forall q''.(q \to_B q'' \implies q'' \not\models L(r)) \quad q \models L(r) \quad q \to_B q' \quad r \xrightarrow{\psi}_S r' \quad q' \models L(r')}{(q, r, \emptyset) \xrightarrow{r,\psi,r'} (q', r', \emptyset)}$$

Table 1: Operational semantics of the flat $S[B]$ system

- Rule AdaptStart regulates the starting of an adaptation phase. Adaptation occurs when all of the next B states do not satisfy the current S state constraints - i.e. $\forall q''.(q \rightarrow_B q'' \implies q'' \not\models L(r))$ - and the B machine is not itself deadlocked ($q \rightarrow_B q'$). In this case, for each S transition $r \xrightarrow{\psi}_S r'$ an adaptation towards the target state r', under the invariant ψ, can start. The flat system performs an adapting transition $\xrightarrow{r,\psi,r'}$ of the form $(q, r, \emptyset) \xrightarrow{r,\psi,r'} (q', r, \{(\psi, r')\})$.

- Rule Adapt can be used only during an adaptation phase. It handles the case in which, after the current transition, the system keeps adapting because a steady configuration cannot be reached ($\forall q''.(q \rightarrow_B q'' \implies q'' \not\models L(r'))$). In this situation, since the system still must adapt ($q \not\models L(r')$), if the B machine is not deadlocked and the invariant can still be satisfied ($q \rightarrow_B q'$ and $q' \models \psi$), the rule allows a transition of the form $(q, r, \{(\psi, r')\}) \xrightarrow{r,\psi,r'} (q', r, \{(\psi, r')\})$. Note that during adaptation the behaviour is not regulated by the S states constraints. Note also that the semantics does not assure that a state where the target S state constraints hold is eventually reached. Two different formulations of such adaptability requirements are given in Section 5.

- Also rule AdaptEnd can only be applied during an adaptation phase and it handles the case in which, after the current transition, the adaptation must end because a steady configuration has been reached ($q' \models L(r')$). It allows a transition $\xrightarrow{r,\psi,r'}$ from an adapting state $(q, r, \{(\psi, r')\})$ to the steady (non-adapting) state $(q', r', \emptyset)$.

- Rule AdaptStartEnd handles the special case in which an adaptation phase must start from a steady situation - $\forall q''.(q \rightarrow_B q'' \implies q'' \not\models L(r)$ - but then, after just one move of the B level, another steady region of the S level is reached ($q' \models L(r')$). In this case the invariant ψ associated to the S transition is ignored and the system goes directly into another steady state. Note that this rule is alternative to the rule AdaptStart in which the initial situation is the same, but the steady region is not reached after one B transition. The flat system performs an adapting transition $\xrightarrow{r,\psi,r'}$ of the form $(q, r, \emptyset) \xrightarrow{r,\psi,r'} (q', r,' \emptyset)$.

Therefore, a successful adaptation occurs when rules AdaptEnd or AdaptStartEnd can be applied, i.e. when a transition from an adapting to a steady state is fired.

Let us now state some properties of the given flat semantics. In the following, given any transition relation $\rightarrow$ and any state s, by $s \rightarrow$ and by $s \not\rightarrow$ we mean, as usual, that there exists a state s' such that $s \rightarrow s'$ and that there exists no state s' such that $s \rightarrow s'$, respectively. Moreover, by $\rightarrow^+$ we indicate a finite, non-empty, sequence of $\rightarrow$ steps; more formally, there exists $n \in \mathbb{N}, n > 0$ such that $s = s_0 \rightarrow s_1 \rightarrow \cdots s_{n-1} \rightarrow s_n$. Finally, by $\rightarrow^k$, $k \geq 0$, we indicate k consecutive steps of the relation $\rightarrow$: $s = s_0 \rightarrow s_1 \rightarrow \cdots s_{k-1} \rightarrow s_k$. If $k = 0$, then $s \rightarrow^0 s'$ is equivalent to say that there is the empty sequence of steps s, and thus $s' = s$. This is always possible, even if the relation $\rightarrow$ is not reflexive.

Proposition 1 (Properties of flat semantics)**.**
Let $\mathcal{F}(S[B]) = (F, f_0, \xrightarrow{r} \cup \xrightarrow{r,\psi,r'})$ be a flat $S[B]$ system. All the following statements hold:

(i) If a steady transition can be performed, then adaptation cannot start:
$\forall (q, r, \emptyset) \in F.\ (q, r, \emptyset) \xrightarrow{r} (q', r, \emptyset) \implies (q, r, \emptyset) \xrightarrow{r,\psi,r'} \not\rightarrow$

(ii) If adaptation can start, then no steady transition is possible:
$\forall (q, r, \emptyset) \in F.\ (q, r, \emptyset) \xrightarrow{r,\psi,r'} (q', r, \{(\psi, r')\}) \implies (q, r, \emptyset) \xrightarrow{r} \not\rightarrow$

(iii) During adaptation no steady transition is possible:
$\forall (q', r', \{(\psi, r')\}) \in F.\ (q', r', \{(\psi, r')\}) \xrightarrow{r} \not\rightarrow$

(iv) The non-adapting and the adapting transition relations are disjoint:
$\forall r, r' \in R, \forall \psi \in \Psi(\Sigma, A).\ \xrightarrow{r} \cap \xrightarrow{r,\psi,r'} = \emptyset$

(v) In case of a successful adaptation, the adaptation phase ends as soon as possible, i.e. as soon as the target steady state can be reached with a single transition.

(vi) Given any $q \in Q$ and $r \in R$, then every infinite *path π in $\mathcal{F}(S[B])$ starting in a state $(q, r, \emptyset)$ such that $q \models L(r)$ is of one of the following kinds:*

(1) *adaptation always succeeds; the path alternates between steady transitions and adaptation paths:*

$$\pi = (q = q_0, r = r_0, \emptyset)(\xrightarrow{r_0})^{m_0}(\xrightarrow{r_0,\psi_0,r_1})^{n_0} \cdots$$
$$\cdots (q_i, r_i, \emptyset)(\xrightarrow{r_i})^{m_i}(\xrightarrow{r_i,\psi_i,r_{i+1}})^{n_i}(q_{i+1}, r_{i+1}, \emptyset) \cdots$$

such that for each $i \geq 0$*, either* $m_i = 1 \wedge n_i = 0$ *(steady transition) or* $m_i = 0 \wedge n_i > 0$ *(adaptation path);*

(2) *adaptation lasts forever; the path has a prefix in which steady transitions and adaptation paths alternate, but then one adaptation path never stops:*

$$\begin{aligned}\pi = (q = q_0, r = r_0, \emptyset)&(\xrightarrow{r_0})^{m_0}(\xrightarrow{r_0,\psi_0,r_1})^{n_0} \cdots \\ \cdots (q_i, r_i, \emptyset)&(\xrightarrow{r_i})^{m_i}(\xrightarrow{r_i,\psi_i,r_{i+1}})^{n_i}(q_{i+1}, r_{i+1}, \emptyset) \cdots \\ \cdots (q_k, r_k, \emptyset) &\xrightarrow{r_k,\psi_k,r_{k+1}} (q_{k+1}, r_k, \{\psi_k, r_{k+1}\}) \\ &\xrightarrow{r_k,\psi_k,r_{k+1}} (q_{k+2}, r_k, \{\psi_k, r_{k+1}\}) \\ &\xrightarrow{r_k,\psi_k,r_{k+1}} \cdots\end{aligned}$$

where $k \geq 0$ *and for each* i*,* $0 \leq i < k$*, either* $m_i = 1 \wedge n_i = 0$ *or* $m_i = 0 \wedge n_i > 0$*.*

(vii) Let $\pi \in \mathcal{F}(S[B])$ *be an* infinite *path starting in a state* $(q, r, \emptyset)$ *such that* $q \models L(r)$*. Then, in every position* i *of the path such that* $\pi[i] = (q_i, r_i, \emptyset)$*, it holds* $q_i \models L(r_i)$*.*

Proof. See Appendix A.1 □

4.1. Termination

In an $S[B]$ system, deadlocks cannot be compatible with adaptability. Indeed, we see adaptability as the property for which a system *continuously* operates under stable, allowed modes (steady states), by possibly performing adaptation paths across modes.

In the flat semantics deadlocks occurring at adapting states, e.g. when the adaptation invariant cannot be met, are clearly conflicting with the concept of adaptability. Instead, deadlocks at steady states are more subtle to interpret, since they may occur under two different conditions:

- from the current state, any transition lead to a state violating the current constraints, and, at the same time, adaptation cannot start because none of the next B states meet any of the adaptation invariants and any of the target constraints. In other words, the flat semantics terminates even if the B level can proceed. Evidently, this violates adaptability.

- the B level cannot progress at all. We consider this situation as a *bad deadlock* state in the behavioural model. Conversely, every B state indicating a *good termination* should have the chance to progress and therefore must be modelled, as usual in this case, with an idling self-loop.

We capture the requirement for which the flat $S[B]$ must not terminate through the $\textsc{Progress}(q, r)$ predicate:

$$\textsc{Progress}(q, r) \iff (q, r, \emptyset) \xrightarrow{r} \vee (q, r, \emptyset) \xrightarrow{r, \psi, r'}$$

4.2. Flat Semantics of the Motion Control Example

The flat semantics of the two systems $S_0[B]$ and $S_1[B]$ implementing the ATV motion controller case study are depicted in Figure 5 and 6, respectively.

Notably, the same behavioural level B possesses different adaptation capabilities depending on the structure S that is considered. Indeed, in $\mathcal{F}(S_0[B])$ every adaptation path leads to a target S state. On the other hand, in $\mathcal{F}(S_1[B])$ always there exists an adaptation path leading to a target stable region, but it may contain cycles of adapting states, thus leading to possibly infinite adaptation paths.

In other words, the behavioural level B is able to successfully adapt under the structural level S_0, for *all possible adaptation paths*. Recalling the definitions introduced in Section 1, $S_0[B]$ is *strong adaptable*. Conversely, B is able to successfully adapt under S_1 only for *some adaptation paths*, i.e. the finite ones. Therefore, $S_1[B]$ is *weak adaptable*. These two different kinds of adaptability are formalized in Section 5.

5. Adaptability Properties

The transitional semantics introduced in Section 4 does not guarantee that an adaptation phase can always start or that, once started, it always ends up in a state satisfying the constraints of the target S state. In this

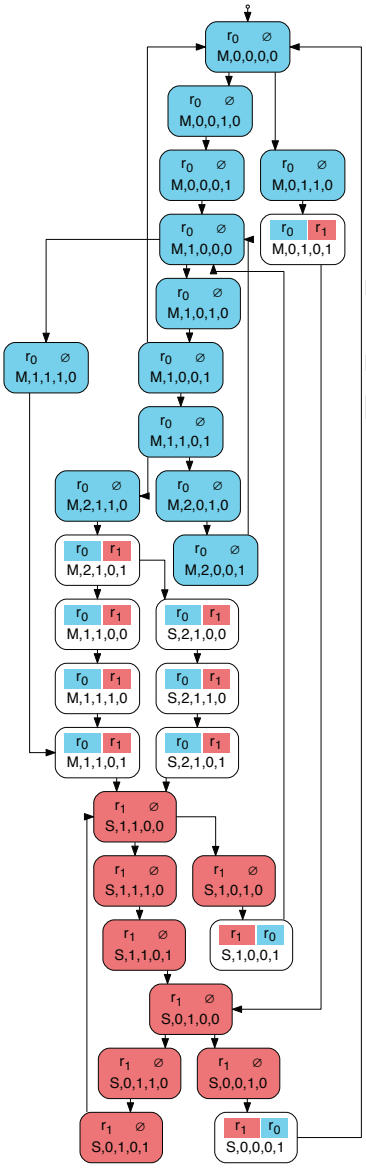


Figure 5: The flat semantics of the system $S_0[B]$. Here, every adaptation path leads to a target S-state.

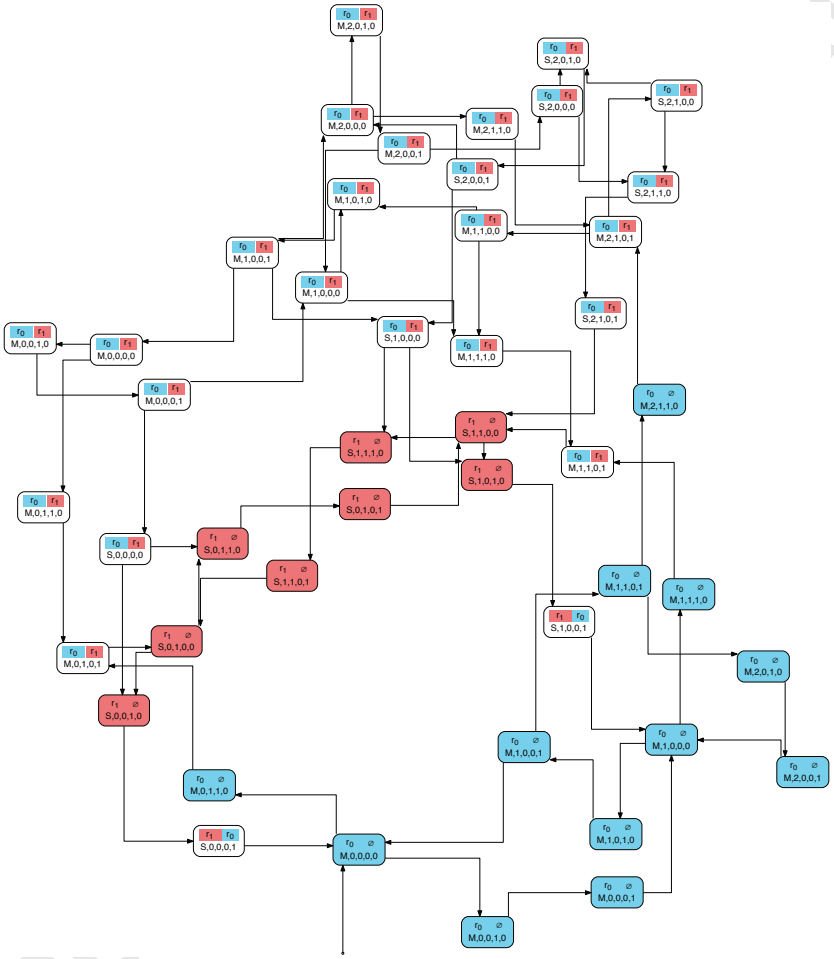


Figure 6: The flat semantics of the system $S_1[B]$. In this case, there are adaptation paths leading to a target stable region, but also infinite adaptation paths due to cycles (see e.g. the adaptation path from r_0 to r_1 $(M,2,1,0,1) \rightarrow (S,2,1,0,0) \rightarrow (S,2,0,1,0) \rightarrow (S,2,0,0,1) \rightarrow (M,2,0,0,0) \rightarrow (M,2,1,1,0) \rightarrow (M,2,1,0,1)$).

section we want to give some formal tools to analyse a given system w.r.t. these kind of properties. As a first step, we characterise two adaptability notions by means of two relations over the set of B states and the set of S states, namely a *weak adaptation relation* $\mathcal{R}_w$ and a *strong adaptation relation* $\mathcal{R}_s$. Then, we characterise the same adaptability notions logically and we prove that they can be model checked by using proper formulae of a temporal logic.

Informally, a state q of B is *weak adaptable* to a state r of S if it satisfies the constraints imposed by r and some of its successors are either weak adaptable to the same r or there is an adaptation phase of the flat system that, from q, reaches a state q' that is weak adaptable to another S state r'. In other words, we require that states satisfying $L(r)$ are in relation with r and that from "border" states, that is to say those that can start an adaptation phase for leaving r, there is always at least one way to safely reach another steady situation in another S state r'. As explained in Sect. 4.1, the PROGRESS predicate is used to rule out steady configurations in a state of bad termination. We formally define this relation using a co-inductive style as it is usually done, for instance, for bisimulation relations.

Definition 7 (Weak adaptation)**.** *Given an $S[B]$ system, a binary relation $\mathcal{R} \subseteq Q \times R$ is a* weak adaptation *if and only if whenever $q\ \mathcal{R}\ r$ we have:*

(i) $q \models L(r)$ *and* PROGRESS(q, r), and

(ii) if $(q, r, \emptyset) \xrightarrow{r}$ *then there exists* $q' \in Q$ *such that* $(q, r, \emptyset) \xrightarrow{r} (q', r, \emptyset)$ *and* $q'\ \mathcal{R}\ r$, and

(iii) if $(q, r, \emptyset) \xrightarrow{r,\psi',r''}$ *for some* $\psi' \in \Psi(\Sigma, A)$ *and* $r'' \in R$ *then there exist* $q' \in Q$, $\psi \in \Psi(\Sigma, A)$ *and* $r' \in R$ *such that* $(q, r, \emptyset) \xrightarrow{r,\psi,r'}{}^{+} (q', r', \emptyset)$ *and* $q'\ \mathcal{R}\ r'$.

We say that a state $q \in Q$ is weak adaptable *to a state $r \in R$, written $q|_w r$, if and only if there is a weak adaptation relation $\mathcal{R}$ such that (q, r) is in $\mathcal{R}$.*

At the level of the whole system, we say that $S[B]$ is weak adaptable *if the initial B state q_0 is weak adaptable to the initial S state r_0.*

Proposition 2 (Union of Weak Adaptation Relations)**.** *Given an $S[B]$ system, if $\mathcal{R}_1$ and $\mathcal{R}_2$ are weak adaptation relations, then $\mathcal{R}_1 \cup \mathcal{R}_2$ is a weak adaptation relation.*

Proof. See Appendix A.2 □

Definition 8 (Weak adaptability)**.** *Given an $S[B]$ system, the union of all weak adaptation relations among the states Q and R of $S[B]$ is denoted by $\mathcal{R}_w$ and is the* weak adaptability *relation of $S[B]$.*

Lemma 1 (Propagation of Weak Adaptation Relation)**.** *Consider an $S[B]$ system and let q and r be such that $q|_w r$. Then there exists in $\mathcal{F}(S[B])$ an infinite path*

$$\pi = (q = q_0, r = r_0, \emptyset)(\xrightarrow{r_0})^{m_0}(\xrightarrow{r_0,\psi_0,r_1})^{n_0} \cdots$$
$$\cdots (q_i, r_i, \emptyset)(\xrightarrow{r_i})^{m_i}(\xrightarrow{r_i,\psi_i,r_{i+1}})^{n_i}(q_{i+1}, r_{i+1}, \emptyset) \cdots$$

such that $\forall i \geq 0 \,.\, q_i|_w r_i \wedge ((m_i = 1 \wedge n_i = 0) \vee (m_i = 0 \wedge n_i > 0))$.

Proof. See Appendix A.3 □

Weak adaptability guarantees that there is always at least one way for a certain state of an $S[B]$ system to adapt, that is to say to continue to evolve in a consistent way w.r.t. the structural constraints of the S level. A stronger property that could be useful to know about the adaptability of a system is what we call *strong adaptability*. A B state q is *strong adaptable* to an S state r if it satisfies the constraints imposed by r and all its successors q' are either strong adaptable to the same r or they are always the starting point of a successful adaptation phase towards states q'' that are strong adaptable to other S states. Again, bad deadlocks are excluded from the relations. This time we require that all the "border" states are safe doors to other steady situations, whatever path is taken from them.

Definition 9 (Strong adaptation)**.** *Given an $S[B]$ system, a binary relation $\mathcal{R} \subseteq Q \times R$ is a* strong adaptation *if and only if whenever $q\ \mathcal{R}\ r$ we have:*

(i) $q \models L(r)$ *and* $\textsc{Progress}(q, r)$, and

(ii) for all $q' \in Q$, if $(q, r, \emptyset) \xrightarrow{r} (q', r, \emptyset)$ then $q'\ \mathcal{R}\ r$, and

(iii) all adaptation paths starting from $(q, r, \emptyset)$ are finite and end up in a state $(q', r', \emptyset)$ such that $q'\ \mathcal{R}\ r'$.

We say that a state $q \in Q$ is strong adaptable *to a state $r \in R$, written $q|_s r$, if and only if there is a strong adaptation relation $\mathcal{R}$ such that (q, r) is in $\mathcal{R}$.*

At the level of the whole system, we say that B is strong adaptable *to S if the initial B state q_0 is strong adaptable to the initial S state r_0.*

Proposition 3 (Union of Strong Adaptation Relations)**.** *Given an $S[B]$ system, if $\mathcal{R}_1$ and $\mathcal{R}_2$ are strong adaptation relations, then $\mathcal{R}_1 \cup \mathcal{R}_2$ is a strong adaptation relation.*

Proof. As in the case of weak adaptation relations. □

Definition 10 (Strong adaptability)**.** *Given an $S[B]$ system, the union of all strong adaptation relations among the states Q and R of $S[B]$ is denoted by $\mathcal{R}_s$ and is the* strong adaptability *relation of $S[B]$.*

In the remainder of the paper we will alternatively say that an $S[B]$ system is weak (strong) adaptable, in the sense that B is weak (strong) adaptable to S. It is straightforward to see that strong adaptability implies weak adaptability, since the strong version of the relation requires that every adaptation path reaches a target S state, while the weak version just requires that at least one adaptation path reaches a target S state.

Proposition 4 (Strong Adaptation implies Weak Adaptation)**.** *Consider an $S[B]$ system and let q and r be such that $q|_s r$. Then, it holds $q|_w r$.*

Proof. See Appendix A.4 □

Given the flat semantics $\mathcal{F}(S[B]) = (F, f_0, \xrightarrow{r} \cup \xrightarrow{r,\psi,r'})$ of an $S[B]$ system, we will denote, in the following, the set of reachable states from a certain state $f \in F$ as the reflexive and transitive closure $Post^*(f)$ of the operator $Post(s) = \{s' \in F \mid (s, s') \in \xrightarrow{r} \cup \xrightarrow{r,\psi,r'}\}$.

Lemma 2 (Propagation of Strong Adaptation Relation)**.** *Consider an $S[B]$ system and let q and r be such that $q|_s r$. Then, every state $(q', r', \emptyset) \in Post^*((q, r, \emptyset))$ is such that $q'|_s r'$.*

Proof. See Appendix A.5 □

The following proposition gives a precise candidate relation for checking if a system is strong adaptable: such a candidate is determined by the steady states of the flat semantics that are reachable from the initial state.

Proposition 5 (Construction of Strong Adaptation Relation)**.** *Given an $S[B]$ system, let $\mathcal{F}(S[B]) = (F, f_0, \xrightarrow{r} \cup \xrightarrow{r,\psi,r'})$ be its flat semantics. Then $S[B]$ is strong adaptable if and only if $\mathcal{R} = \{(q, r) \in Q \times R \mid (q, r, \emptyset) \in Post^*(f_0)\}$ is a strong adaptation relation.*

Proof. See Appendix A.6 □

5.1. Adaptation Relations in the Motion Controller Example

In the following we will show that in the ATV motion controller case study $S_0[B]$ is strong adaptable (and thus also weak adaptable) and $S_1[B]$ is weak adaptable, but not strong adaptable.

In order to verify that $S_0[B]$ is strong adaptable, we need to prove that $q_0|_s r_0$, by finding a strong adaptation relation $\mathcal{R}$ s.t. $(q_0, r_0) \in \mathcal{R}$. Note that in $\mathcal{F}(S_0[B])$, every state is reachable from the initial state $(0, r_0, \emptyset)$. Therefore by Proposition 5, we consider the relation $\mathcal{R} = \{(q, r) \mid q \in Q, r \in R, (q, r, \emptyset) \in F\}$, where F is the set of flat states of $\mathcal{F}(S_0[B])$. It is easy to verify that $\forall (q, r) \in \mathcal{R}.\ q \models L(r)$; and that $\textsc{Progress}(q, r)$ holds for any of such states, because there are no deadlock states in the flat semantics. Therefore condition (i) of the definition of strong adaptation is always true and has not to be further checked. Clearly, $(q_0, r_0) \in \mathcal{R}$. Finally, it can be shown that requirements (ii) and (iii) of Def. 9 hold for each element of $\mathcal{R}$.

We show the proof for $((M, 0, 0, 0, 0), r_0)$ and $((M, 0, 1, 1, 0), r_0)$. The other pairs can be proved analogously.

- $((M, 0, 0, 0, 0), r_0)$
  - (ii) $((M, 0, 0, 0, 0), r_0, \emptyset) \xrightarrow{r_0} ((M, 0, 0, 1, 0), r_0, \emptyset)$ and $((M, 0, 0, 1, 0), r_0) \in \mathcal{R}$; $((M, 0, 0, 0, 0), r_0, \emptyset) \xrightarrow{r_0} ((M, 0, 1, 1, 0), r_0, \emptyset)$ and $((M, 0, 1, 1, 0), r_0) \in \mathcal{R}$
  - (iii) $((M, 0, 0, 0, 0), r_0, \emptyset) \xrightarrow{r, \psi, r'} \!\!\!\!\!\!/\ \ $
- $((M, 0, 1, 1, 0), r_0)$
  - (ii) $((M, 0, 1, 1, 0), r_0, \emptyset) \overset{r}{\not\to}$
  - (iii) there is only one adaptation path from $((M, 0, 1, 1, 0), r_0, \emptyset)$ leading to the flat state $((S, 0, 1, 0, 0), r_1, \emptyset)$ (through the B state $(M, 0, 1, 0, 1)$), and $((S, 0, 1, 0, 0), r_1) \in \mathcal{R}$.

On the other hand, we demonstrate that $S_1[B]$ is weak adaptable by finding a weak adaptation relation $\mathcal{R}$ s.t. $(q_0, r_0) \in \mathcal{R}$. We take as $\mathcal{R}$ the following relation:

$$\{((M, 0, 0, 0, 0), r_0), ((S, 0, 1, 0, 0), r_1), ((M, 0, 1, 1, 0), r_0), ((S, 0, 0, 1, 0), r_1)\}$$

Similarly to $S_0[B]$, $(q_0, r_0) \in \mathcal{R}$ and for all $(q, r) \in \mathcal{R}$, $q \models L(r)$ and $\textsc{Progress}(q, r)$ both holds. Thus, we need to check requirements (ii) and (iii) of Definition 7 to prove that $\mathcal{R}$ is a weak adaptation relation.

We observe that pairs $((M, 0, 0, 0, 0), r_0)$ and $((S, 0, 1, 0, 0), r_1)$ meet requirements (ii) and (iii), the latter being trivially verified since they do not admit adaptation transitions. Pairs $((M, 0, 1, 1, 0), r_0)$ and $((S, 0, 0, 1, 0), r_1)$ also comply with the weak adaptation definition, since they can both reach, in a finite number of adaptation steps, flat states that map to elements in $\mathcal{R}$, i.e. $((S, 0, 1, 0, 0), r_1)$ and $((M, 0, 0, 0, 0), r_0)$, respectively.

However, note that $((S, 0, 1, 0, 0), r_1)$ cannot be in any strong adaptation relation because, by the propagation property, $((M, 2, 1, 1, 0), r_0)$ would be in the relation and there are infinite adaptation paths starting from its corresponding flat state (see Figure 6 and its caption). We conclude that $\mathcal{R}$ is a weak and not strong adaptation relation.

6. Logical Characterisation of Adaptability Properties

In this section we formulate the adaptability properties introduced in Section 5 in terms of formulae of a temporal logic that can be model checked [2, 17].

To this purpose we briefly recall the well-known *Computation Tree Logic (CTL)* [14, 15], a branching-time logic whose semantics is defined in terms of paths along a Kripke structure [33]. Given a set AP of atomic propositions, a Kripke structure is a tuple $(T, t_0, \rightarrow_k, I)$ where T is a finite set of states, t_0 is the initial state, $\rightarrow_\kappa \subseteq T \times T$ is a left-total transition relation and $I \colon T \rightarrow 2^{AP}$ maps each state to the set of atomic propositions that are true in that state. Given a state $t \in T$, a path π starting from t has the form $\pi \colon t = t_0 \rightarrow_\kappa t_1 \rightarrow_\kappa t_2 \rightarrow_\kappa \cdots$, where for all $i = 1, 2, \ldots, (t_{i-1}, t_i) \in \rightarrow_\kappa$. Given a path π and an index $i > 0$, by $\pi[i]$ we denote the i-th state along the path π. The set of all paths starting from t is denoted by $Paths(t)$. Note that, since the transition relation is required to be left-total, all runs are infinite. To model a deadlocked or terminated state in a Kripke structure the modeller must put a self-cycle on that state.

The set of well-formed CTL formulae are given by the following grammar:

$$\phi ::= \quad true \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{AX}\phi \mid \mathbf{EX}\phi \mid \mathbf{A}[\phi \mathbf{U} \phi] \mid \mathbf{E}[\phi \mathbf{U} \phi]$$

where $p \in AP$ is an atomic proposition, logical operators are minimal $(\neg, \wedge)$ in order to generate all the usual ones, and temporal operators ($\mathbf{X}$ next,

U until) quantify along paths and must be preceded by the universal path quantifier **A** or by the existential path quantifier **E**.

Given a state t of the underlying Kripke structure, the satisfaction of a CTL formula ϕ in t, written $t \models_{\text{CTL}} \phi$, is defined inductively as follows.

$$
\begin{array}{lll}
t \models_{\text{CTL}} \mathit{true} & & \text{for all } t \\
t \models_{\text{CTL}} p & \text{iff} & p \in I(t) \\
t \models_{\text{CTL}} \neg\phi & \text{iff} & t \not\models_{\text{CTL}} \phi \\
t \models_{\text{CTL}} \phi_1 \wedge \phi_2 & \text{iff} & t \models_{\text{CTL}} \phi_1 \text{ and } t \models_{\text{CTL}} \phi_2 \\
t \models_{\text{CTL}} \mathbf{AX}\phi & \text{iff} & \forall \pi \in \mathit{Paths}(t).\pi[1] \models_{\text{CTL}} \phi \\
t \models_{\text{CTL}} \mathbf{EX}\phi & \text{iff} & \exists \pi \in \mathit{Paths}(t)\colon \pi[1] \models_{\text{CTL}} \phi \\
t \models_{\text{CTL}} \mathbf{A}[\phi_1 \mathbf{U} \phi_2] & \text{iff} & \forall \pi \in \mathit{Paths}(t).\exists j \geq 0\colon (\pi[j] \models_{\text{CTL}} \phi_2 \text{ and} \\
 & & \quad \forall 0 \leq i < j.\pi[i] \models_{\text{CTL}} \phi_1) \\
t \models_{\text{CTL}} \mathbf{E}[\phi_1 \mathbf{U} \phi_2] & \text{iff} & \exists \pi \in \mathit{Paths}(t)\colon \exists j \geq 0\colon (\pi[j] \models_{\text{CTL}} \phi_2 \text{ and} \\
 & & \quad \forall 0 \leq i < j.\pi[i] \models_{\text{CTL}} \phi_1)
\end{array}
$$

Other useful temporal operators like $\mathbf{EF}\phi$ (ϕ holds potentially), $\mathbf{AF}\phi$ (ϕ is inevitable), $\mathbf{EG}\phi$ (potentially always ϕ) and $\mathbf{AG}\phi$ (invariantly ϕ), are derived, as usual, as follows: $\mathbf{EF}\phi \equiv \mathbf{E}[\mathit{true}\mathbf{U}\phi]$, $\mathbf{AF}\phi \equiv \mathbf{A}[\mathit{true}\mathbf{U}\phi]$, $\mathbf{EG}\phi \equiv \neg\mathbf{AF}\neg\phi$ and $\mathbf{AG}\phi \equiv \neg\mathbf{EF}\neg\phi$.

In the following we provide a Kripke structure derived from the flat semantics $\mathcal{F}(S[B])$ and two CTL formulae characterising weak and strong adaptability.

Definition 11 (Associated Kripke structure)**.** *Consider an $S[B]$ system and its associated flat semantics $\mathcal{F}(S[B]) = (F, f_0, \xrightarrow{r} \cup \xrightarrow{r,\psi,r'})$. Its* associated Kripke structure *is defined as $\mathcal{K}(S[B]) = (T, t_0, \rightarrow_k, I)$ where $T = F$, $t_0 = f_0$, $\rightarrow_k = \xrightarrow{r} \cup \xrightarrow{r,\psi,r'} \cup (\rightarrow_{\mathit{self}} \triangleq \{(t,t) \mid t \not\xrightarrow{r} \wedge t \not\xrightarrow{r_s,\psi,r'}\})$, I is defined w.r.t. the set $AP = \{\mathit{adapting}, \mathit{steady}, \mathit{progress}\}$ of atomic propositions as follows. For all $t \in T$:*

(i) $\mathit{adapting} \in I(t) \iff t = (q, r, \rho) \wedge t \xrightarrow{r,\psi,r'}$

(ii) $\mathit{steady} \in I(t) \iff t = (q, r, \emptyset) \wedge (t \xrightarrow{r} \vee\, t \xrightarrow{r,\psi,r'})$

(iii) $\mathit{progress} \in I(t) \iff t \xrightarrow{r} \vee\, t \xrightarrow{r,\psi,r'}$

Note that the only structural difference between $\mathcal{F}(S[B])$ and $\mathcal{K}(S[B])$ are the self-loop transitions in $\rightarrow_{\mathit{self}}$ added in $\mathcal{K}(S[B])$. These are needed because

the transition relation of the Kripke structure must be left-total, but indeed they allow us to keep the information that the states t such that $t \rightarrow_{self} t$ where originally deadlocked in $\mathcal{F}(S[B])$ (see discussion in Section 4.1). Then, the atomic proposition *progress*, by its definition (identical to the one given for PROGRESS(q, r) at the end of Section 4.1), is *not* true in all and only those states of $\mathcal{K}(S[B])$ that were originally deadlocked or bad terminated in $\mathcal{F}(S[B])$. Moreover, we remark that in some states both *adapting* and *steady* propositions may hold at the same time. These are the already mentioned "border" states, i.e. those that are still in a steady situation, but will start adapting in the next transition. From these, the next state may be *steady* again (immediate adaptation) or only *adapting* (adaptation in more than one step).

The formulae that we will check on $\mathcal{K}(S[B])$ are the following:

- **Weak adaptation:** there is a path in which the progress condition continuously holds and, as soon as adaptation starts, there exists at least one path for which the system eventually ends the adaptation phase leading to a steady state.

$$\mathbf{EG}((adapting \implies \mathbf{EF}\ steady) \wedge progress) \tag{6.1}$$

- **Strong adaptation:** for all paths, the progress condition always holds and whenever the system is in an adapting state, from there all paths eventually ends the adaptation phase leading to a steady state.

$$\mathbf{AG}((adapting \implies \mathbf{AF}\ steady) \wedge progress) \tag{6.2}$$

We remark that the same formulae could be expressed in the Action-based Computation Tree Logic (ACTL) [21] without the need of defining the atomic propositions. However, we decided to use CTL because we recognise that it is one of the mostly known and used temporal logic for model checking,

Moreover, the expressive power given by CTL is adequate for the adaptability checking we introduced, in the sense that the same properties could not be expressed in the other mostly used logic, Linear Temporal Logic (LTL) [40]. In particular, the weak adaptability property requires the existential path quantification (**EG**, **EF**), which cannot be expressed in LTL, to render the invariability of the possibility of adaptation along *one* certain computation. Differently, the strong adaptability property could also be formulated in LTL as $\Box((adapting \Rightarrow \Diamond steady) \wedge progress)$.

Theorem 1 (Weak adaptability checking)**.**
Consider an $S[B]$ system. Given a B state q and an S state r such that $q \models L(r)$, then q is weak adaptable to r if and only if the weak adaptation CTL formula (equation 6.1) is true in $\mathcal{K}(S[B])$ at state $(q, r, \emptyset)$. Formally, given a state $q \in [[L(r)]]$

$$q \mid_w r \iff (q, r, \emptyset) \models_{\text{CTL}} \mathbf{EG}((adapting \implies \mathbf{EF}\ steady) \wedge progress)$$

Proof. See Appendix A.7 □

Corollary 1.
Consider an $S[B]$ system. Then, $S[B]$ is weak adaptable if and only if

$$t_0 \models_{\text{CTL}} \mathbf{EG}((adapting \implies \mathbf{EF}\ steady) \wedge progress)$$

where t_0 is the initial state of $\mathcal{K}(S[B])$.

Proof. The thesis follows easily from Definition 7 and from Theorem 1. □

Theorem 2 (Strong adaptability checking)**.**
Consider an $S[B]$ system. Given a B state q and an S state r such that $q \models L(r)$, then q is strong adaptable to r if and only if the strong adaptation CTL formula (equation 6.2) is true in $\mathcal{K}(S[B])$ at state $(q, r, \emptyset)$. Formally, given a state $q \in [[L(r)]]$

$$q \mid_w r \iff (q, r, \emptyset) \models_{\text{CTL}} \mathbf{AG}((adapting \implies \mathbf{AF}\ steady) \wedge progress)$$

Proof. See Appendix A.8 □

Corollary 2.
Consider an $S[B]$ system. Then, $S[B]$ is strong adaptable if and only if

$$t_0 \models_{\text{CTL}} \mathbf{AG}((adapting \implies \mathbf{AF}\ steady) \wedge progress)$$

where t_0 is the initial state of $\mathcal{K}(S[B])$.

Proof. As in the weak case, the thesis follows easily from Definition 9 and from Theorem 2. □

Note that since we assume that the behavioural and the structural state machines are finite state, then the CTL adaptability properties can be model checked. This means that the defined notions of weak and strong adaptability are decidable and that the problem of adaptability checking can be reduced to a classical CTL model checking problem.

6.1. State Space Dimension

CTL model checking has been widely investigated in the literature and relies on efficient tools like NuSMV [13]. The time computational complexity of the model checking problem for CTL is $O((n+m)\cdot|\psi|)$, where n is the number of states in the Kripke structure, m is the number of transitions and $|\psi|$ is the length of the formula, i.e. the number of operators in its parse tree.

The well-known problem in this area is the so-called state explosion problem, that is to say the high number of states (and thus transitions) that comes out from even relatively short descriptions of systems composed of concurrent interactive components. It is not in the scope of this work to discuss and refer the high research efforts that are currently going on in this area. A good starting point can be found in [2]. We will just give a brief estimation of the dimension of the state space given a certain $S[B]$ system, which is the dominant complexity factor, considering that our formulae for the adaptability checking have constant length 4. The computational complexity of adaptability checking is therefore $O(n+m)$.

Note that in our context the usual sources of state explosion (components, concurrency) are "hidden" inside the behavioural level B. This is because we want to work on the very basic model of computation of finite state machines and maintain a black-box view of the behavioural level from the structural level point of view. Thus, we take as the dominant dimension of the problem the "already exploded" number of states of the behavioural level. Then, we estimate what the definition of an $S[B]$ model adds up to this explosion.

Recall that the Kripke structure to model check is $\mathcal{K}(S[B])$, derived from the flat semantics $\mathcal{F}(S[B])$. Depending on the dimension of S, the flat semantics could possibly lead to a transition system larger than the state space of the behavioural model B since the states are formally tuples in $Q\times R\times(\{(\psi,r')\mid\exists r\in R.\ r\xrightarrow{\psi}_S r'\}\cup\{\emptyset\})$. The dimension n of the state space is $O(|Q|\cdot 2^{|Q|}\cdot 2^{|Q|})$, where $|\cdot|$ is set cardinality, based only on the number of states in the behavioural level and due to the higher order nature of S. However, considering the intended role of the structural level S, the number of S states is never exponential. This is because the different S states represent different modes of operations and, thus, usually stand for disjoint sets of B states. For this reason, a more realistic estimation of the state space dimension n should be expressed w.r.t. both B and S dimensions, yielding a number that is $O(|Q|\cdot|R|\cdot|\to_S|)$.

7. Discussion and Conclusion

In this work we presented a formal hierarchical model for self-adaptive systems, consisting of a behavioural level and a structural level. The B level is a state machine describing the behaviour of the system and the S level is a second-order state machine accounting for the constraints which the system has to comply with. S states identify stable regions that the B level may reach by performing adaptation paths.

The adaptation semantics of the system is given by a flattened transition system that implements the following adaptation model: adaptation starts whenever the current B state does not meet the constraints specified by the current S state. Then, adaptation towards a target S state r' ends successfully when the system ends up in a different B state q' such that q' satisfies the constraints in r'.

We tackled the adaptability checking problem by firstly characterizing two degrees of adaptability: *weak adaptability*, for verifying if the system is able to adapt successfully for some adaptation paths; and *strong adaptability*, for verifying if the system is able to adapt successfully for all possible adaptation paths. Then, we defined weak and strong adaptation as relations over the set of B states and the set of S states, so that adaptability is verified when an appropriate adaptation relation can be built. We also provided a logical formulation of weak and strong adaptability, in terms of CTL formulae. Finally, by proving that the logical characterization is formally equivalent to the relational one, we demonstrated that the adaptability checking problem can be reduced to a classical model checking problem. We derived the computational complexity of the problem and showed that the state space dimension is polynomial in the dimension of the original $S[B]$ system.

The approach has been elucidated through an example of self-adaptive software systems, that is the motion controller of an autonomous transport vehicle. We considered two structural levels with different adaptation invariants, S_0 and S_1. Keeping the behavioural model B fixed, we derived the flat semantics of $S_0[B]$ and $S_1[B]$ and we compared their adaptation capabilities, showing that the former is strong adaptable, while the latter is only weak adaptable.

We report that this work gives a formal computational characterization of self-adaptive systems, and a novel and well-grounded formulation of the concept of adaptability. We elaborate effective formal methods to investigate

and solve the problem of adaptability checking. Provided that $S[B]$ systems are based on a general and essential model of computation (state machines), our results are general too and can be easily applied to richer and more expressive models.

7.1. Multiple Levels and Modular Adaptability Checking

Although we have investigated the relationships between two fundamental levels, it is possible to show how our model can easily scale-up to an arbitrary number of levels, arising from the composition of multiple $S[B]$ systems. We give an intuition of how a higher order $S[B]$ can be defined. In these settings, a first-order $S[B]$ system is a "classical" system, as defined in Sect. 2. For $n > 1$, a n^{th}-order $S[B]$ system is an $S[B]$ system $S^n[B^n]$, where $S^n = (R^n, r_0^n, A^n, \mathcal{O}^n, \rightarrow_S^n, L^n)$ is the structural level; and $B^n = \|_{i \in I}\ \mathcal{F}(S^{n-1}[B^{n-1}]_i)$ is the behavioural level resulting from the application of a parallel composition operation '$\|$' to the flattened semantics of a family of $n-1^{th}$-order $S[B]$ systems, indexed by $i \in I$.

Further, due to the separation between the S and the B levels, modular techniques for adaptability checking could be exploited in our model. Let $S_1 = (R_1, r_{1_0}, A_1, \mathcal{O}_1, \rightarrow_{S_1}, L_1)$ and $S_2 = (R_2, r_{2_0}, A_2, \mathcal{O}_2, \rightarrow_{S_2}, L_2)$ be two S levels. For instance, we would be interested in showing if the adaptation capabilities of S_2 are preserved by S_1, in the case that S_1 refines S_2, or $S_1 \preceq S_2$. To our purposes we may assume that $S_1 \preceq S_2$ iff a suitable simulation relation $\mathcal{R} \in R_1 \times R_2$ exists.

The following result would come quite straightforwardly: if $S_1 \preceq S_2$, it can be shown that for every B level, if $S_2[B]$ is strong adaptable, then $S_1[B]$ is strong adaptable too, i.e. that refinements at the S level would preserve strong adaptability. On the other hand, refinements do not necessarily preserve weak adaptability when $S_2[B]$ is weak adaptable but not strong adaptable. Instead, we cannot make any assumption on the adaptability of S_2 based just on the adaptability of its refinement S_1.

Modular adaptability could be investigated also in the opposite case, i.e. making the S level vary and considering two behavioural levels B_1 and B_2 such that $B_1 \preceq B_2$. Intuitively, it can be demonstrated that in this case abstractions at the B level preserve weak adaptation, or alternatively that if $S[B_1]$ is weak adaptable, then $S[B_2]$ is weak adaptable too for each structural level S.

We leave the two topics briefly introduced above as future work, where also other features of the $S[B]$ model can be developed.

7.2. Related Work

Let us now characterise our approach according to the "taxonomy of self-adaptation", a quite general software-oriented classification proposed by Salehie and Tahvildari [43]. Specifically, the features of the taxonomy considered here are *adaptation type*, or *how* adaptation is realized; temporal issues, related to *when* the system needs to change and to be monitored to achieve adaptation; and *interaction* aspects.

Interaction. In the $S[B]$ model communication and interactions of the adaptive system with other systems are not explicitly taken into account. This is because we focus on studying the adaptation capabilities of a fundamental model of computation, to which more powerful and expressive models can typically be reduced. Indeed, we always consider the behavioural level B as the transitional semantics of a system constructed from several interacting components.

Temporal characteristics. Recalling the introductory description of the adaptation semantics, *adaptation starts as late as possible*, only when no other evolution is possible that fulfils the current constraints; and *adaptation ends as soon as possible*, i.e. as soon as a target state can be reached. This implies that $S[B]$ systems support a basic type of *proactive* (i.e. anticipatory) adaptation, which ensures that the system reaches a state where structural constraints or adaptation invariants are not met if and only if no other evolution is possible.

The choice to exclude adaptations starting from states that can progress normally, i.e. without violating the constraints, is motivated by the same definition of adaptation: a mutation in an individual that leads to a *higher* level of fitness. Indeed, as stated in [6], an adaptive system *"... seeks to configure its structure with the overall aim of adaptation to the environment trying to optimize its function (i.e., to maximize its fit) to meet its reason of existence"*. Our model provides just a qualitative characterization of the fitness of a B state $\bar{q}$ in a S state $\bar{r}$, given by the satisfaction value (true or false) of the constraints. Therefore, no adaptations can start from the state $\bar{q}$ if it can satisfy the constraints in $\bar{r}$ during its evolution, since this configuration corresponds to the highest possible fitness, and any adaptation would produce *equal* (in case of successful adaptation) or *lower* (unsuccessful adaptation) fitness values. Nevertheless according to this semantics, the system cannot prevent an unsuccessful adaptation to occur by starting adaptation earlier, i.e. at a state admitting successors that are in the current steady region.

Adaptation type. This feature concerns aspects related to the implementation of adaptation mechanisms. According to the taxonomy above, our approach falls into the definitions of *model-based adaptation*, i.e. based on a model of the system and of the environment; and of *close adaptation*, in the sense that the system has only a fixed number of adaptive actions, due to the fact that we focus on models with finite and fixed state space. On the contrary, *open-adaptive* systems support the runtime addition of adaptation actions. Salehie and Tahvildari also distinguish between *external* and *internal* adaptation. $S[B]$ systems belong to the former type, where the adaptation logic (S level) and the application logic (B level) are separated. In internal adaptation, conversely, adaptation mechanisms are mixed at the application level.

Taking a broader view that generalizes from software systems, Sagasti [42] distinguishes between two different adaptive behaviours: the system adapts by modifying itself (*Darwinian adaptation*); or it adapts by modifying its environment (*Singerian adaptation*). In this work, we clearly focus on the former type of adaptation. Following this line, the adaptation type of $S[B]$ systems can be further classified as a *top-down* and *behavioural* adaptation. Top-down, because the S level imposes high-level functions (e.g. constraints, rules and policies) on the lower B level, which adapts itself whenever it cannot fulfil the current constraints. Bottom-up adaptation represents the opposite direction, occurring for instance when new higher-level patterns emerge from the lower level. On the other hand, behavioural adaptation is related to functional changes, like changing the program code or following different trajectories in the state space. In literature, it is generally opposed to structural adaptation, which is related to architectural reconfiguration, e.g. addition, migration and removal of components. Note that structural and behavioural adaptation must not be confused with the structural and the behavioural level of an $S[B]$ system.

Behavioural Adaptation. Zhang et al. give a general state-based model of self-adaptive programs, where the adaptation process is seen as a transition between different non-adaptive regions in the state space of the program [46]. In order to verify the correctness of adaptation they define a logic called A-LTL (an adapt-operator extension to LTL) and model-checking algorithms [47] for verifying adaptation requirements. Similarly, in $S[B]$ systems adaptation can be seen as a transition in the S level between two steady regions (the S states), which corresponds to performing a path at the B level. However,

in our model the steady-state regions are represented in a more declarative way using constraints associated to the states of the S level. Adaptation of the B level is not necessarily instantaneous and during this phase the system is left unconstrained but an invariant condition that is required to be met during adaptation. Differently to [46], the invariants are specific for every adaptation transition making this process controllable in a finer way.

PobSAM [30, 31] is another formal model for self-adaptive systems, where actors expressed in Rebeca are governed by managers that enforce dynamic policies (described in an algebraic language) according to which actors adapt their behaviour. Different adaptation modes allow to handle events occurring during adaptation and ensuring that managers switch to a new configuration only once the system reaches a safe state. Similarly to our structural and behavioural levels, the structure of a PobSAM model is built on multiple levels: the level of managed actors, the levels of autonomous managers and a view level, which acts as a sort of observation function over the state variables of the actors. Further, a recently published extension of PobSAM called HPobSAM [29], enables the hierarchical refinement of managed components. On the other hand, $S[B]$ systems are based on the general formalism of state machines and enjoy the property that higher levels lead to higher-order structures: indeed, the S level can be interpreted as a second-order B level, since an S state identifies a set of stable B states and firing an S transition means performing an adaptation path, i.e. a sequence of B transitions.

In the position paper by Bruni et al. [10], adaptation is defined as the run-time modification of the control data of a system and this approach is instantiated into a formal model based on labelled transition systems. They consider a system S that is embedded in some environment $\mathcal{E}$ and that has to fulfil a goal ψ. When the environment and the goal are fixed, the system S is such that:

$$\mathcal{E}[S] \models \psi,$$

where $\mathcal{E}[S]$ can be alternatively expressed as the parallel composition $\mathcal{E} \| S$. However, S may operate under run-time modifications in the environment and goal. Thus, when the environment $\mathcal{E}$ changes into $\mathcal{E}'$ and the goal ψ into ψ', S adapts itself into S' such that:

$$\mathcal{E}'[S'] \models \psi'.$$

As shown in [22], the problem of finding such an S' can be formulated as an LTS control problem. Similarly, in our context we can see an $S[B]$ system

as a behavioural model B embedded in a structure S. Let $\bar{r}_S[\bar{q}_B]$ denote the current state of the $S[B]$ system. Then,

$$\langle S[B], \bar{r}_S[\bar{q}_B] \rangle \models \psi_{S[B]},$$

where $\psi_{S[B]} \equiv \bar{q}_B \in L(\bar{r}_S)$, i.e. the current B state must meet the constraints imposed by the current S state. Therefore, the structural level S not only acts as the operating environment for B, but also encodes the goal ψ, which requires that B has to move within the stable region identified by the constraints in the current S state. We can imagine that whenever ψ is no longer satisfied, adaptation produces a system $S'[B']$ from the current system $S[B]$, in a way that

$$\langle S'[B'], \bar{r}_{S'}[\bar{q}_{B'}] \rangle \models \psi_{S'[B']}.$$

In our settings, the control data component is not explicitly implemented and there are no transitions that can be directly controlled, but due to the top-down adaptation semantics, the S level provides some kind of control mechanism on B. Indeed, during the steady phase, S forbids any transition to a B state that violates the current constraints if there is at least one transition to a state satisfying them. Instead, when adaptation starts, we can think that S outputs to B a list of target S states, so directing the evolution of B towards a set of possible goals and excluding those transitions that lead to states violating the adaptation invariant.

The adaptation as control data modification view of [10] is implemented also in the formalism of Adaptable Interface Automata (AIAS) [9], which extends Interface Automata [20] with state-labelling atomic propositions, a subset of which - the control propositions - models the control data. Adaptation occurs in correspondence of transitions that change the control propositions, and actions labelling such transitions are called control actions. Similarly to our $S[B]$ systems, an adaptation phase is thus a sequence of adaptation transitions. Let $\mathcal{A}^C, \mathcal{A}^I, \mathcal{A}^O$ denote the set of control actions and the sets of input and output actions of the underlying interface automaton, respectively. On top of these actions, the authors provides different characterizations of an AIA P: P is adaptable when $\mathcal{A}^C \neq \emptyset$; P is controllable when $\mathcal{A}^C \cap \mathcal{A}^I \neq \emptyset$; and P is self-adaptive when $\mathcal{A}^C \cap \mathcal{A}^O \neq \emptyset$. Given an AIA P, adaptability properties are defined as those that are satisfied by P, but are not satisfied if control actions are removed, i.e. by the AIA $P_{|\mathcal{A} \setminus \mathcal{A}^C}$. In addition, the authors show how our notions of weak and strong adaptability can be encoded in their framework.

Theorem-proving techniques have also been used for assessing the correctness of adaptation: in [34] a proof lattice called transitional invariant lattice is built to verify that an adaptive program satisfies global invariants before and after adaptation. In particular it is proved that if it is possible to build that lattice, then adaptation is correct. Instead, in our model the notion of correctness of adaptation is formalized by means of the weak and strong adaptability relations, that can be alternatively expressed as CTL formulae, thus reducing the problem of adaptability checking to a classical model checking problem.

Furthermore, in [45], the authors define a spatial and chemical-inspired tuple-space model in the context of distributed pervasive services. They show that equipping the classical tuple-space model with reaction and diffusion rules makes possible to support features like adaptivity and competition among services, by implementing Lotka-Volterra-like rules. Similarly, $S[B]$ systems are inspired by complex natural systems, where the dichotomy between the behavioural level and the structural level may represent for instance the genotype and the phenotype level of an organism, respectively; or, by using the metaphor of multiscale systems, the B level can be used to model the system at the micro-scale (e.g. cellular scale), while the S level would represent the emergent macro-scale features (e.g. the tissue). However, our model is not quantitative and is not based on a coordination model, but on a simple and general model of computation (state machines). Moreover, adaptation is not the result of nature-inspired rules, but is rigorously defined from the B and the S level by operational semantics rules.

Structural Adaptation. Our model currently supports only behavioural adaptation, but several approaches have been recently defined also in the context of structural adaptation, most of them relying on dynamic software architectures. In [7], several formal techniques for specifying self-managing architectures, i.e. able to support autonomous run-time architectural changes, are surveyed and compared.

An important line of research focuses on the application of graph-based methods, initiated by Le Métayer's work [36] where architectural styles are captured by graph grammars and graph rewriting rules are used to enable architectural reconfiguration. Other relevant literature in this field includes the Architectural Design Rewriting (ADR) framework [11], in which an architectural style is described as an algebra over typed graphs (i.e. the architectures), whose operators correspond to term-rewriting rules. ADR supports the well-

formed compositions of architectures, the hierarchical specification of styles, style checking and style-preserving reconfiguration. A method for selecting which rules to apply for maintaining a particular architectural style against unexpected run-time reconfigurations is proposed in [41], where the authors extend ADR rules with pre- and post-conditions, representing invariants to be met by an architecture before and after the application of a rule. Tool support is discussed in [8], where the authors provide an in-depth comparison between the implementations of typed graph grammars in Alloy [27] and of ADR in Maude [18].

Moreover, in [5], an approach for verifying safety properties in structurally adapting multi-agent systems is presented. By modelling a system as a graph and its evolution by graph transformation rules, safety properties are verified by means of structural invariants, i.e. a set of forbidden graph patterns.

Hierarchical and Multi-level Methods. Besides $S[B]$ systems, multi-level approaches have been extensively used for the modelling of self-adaptive software systems. For instance, in [19] Corradini et al. identify and formally relate three different levels: the *requirement level*, dealing with high-level properties and goals; the *architectural level*, focusing on the component structure and interactions between components; and the *functional level*, accounting for the behaviour of a single component.

Furthermore, Kramer and Magee [32] define a three-level architecture for self-managed systems consisting of a *component control level* that implements the functional behaviour of the system by means of interconnected components; a *change management level* responsible for changing the lower component architecture according to the current status and objectives; and a *goal management level* that modifies the lower change management plans according to high-level goals.

Hierarchical finite state machines, in particular Statecharts [26] have also been employed to describe the multiple architectural levels in self-adaptive software systems (see e.g. [28, 44]).

Relevant applications of multi-level approaches include the work by Zhao et al. [48], where the authors present a two-level model for self-adaptive systems consisting of a functional behavioural level - accounting for the application logic and modelled as state machines - and an adaptation level, accounting for the adaptation logic and represented with a mode automaton [37]. In this case, each mode in the adaptation level is associated with different functional state machines and adaptation is seen as a change of mode. Adaptation

properties are described and checked by means of a mode-based extension of LTL called mLTL.

Another accepted fact is that higher levels in complex adaptive systems lead to higher-order structures. Here the higher S level is described by means of a second order state machine (i.e. a state machine over the power set of the B states). Similar notions have been formalized by Baas [1] with the *hyperstructures* framework for multi-level and higher-order dynamical systems; and by Ehresmann and Vanbremeersch with their *memory evolutive systems* [23], a model for hierarchical autonomous systems based on category theory.

There are several other works worth mentioning, but here we do not aim at presenting an exhaustive state-of-the-art in this widening research field. We address the interested reader to the surveys [12, 43] for a general introduction to the essential aspects and challenges in the modelling of self-adaptive software systems.

Acknowledgements

The authors want to thank the anonymous reviewers for their valuable suggestions and prof. Mario Rasetti for the continuous inspiration and the useful discussions about the topics of this work and its general context. This work was supported by the project "TOPDRIM: *Topology Driven Methods for Complex Systems*" funded by the European Commission (FP7 ICT FET Proactive - Grant Agreement N. 318121).

References

[1] N. Baas, Emergence, hierarchies, and hyperstructures, in: C. Langton (Ed.), Artificial Life III, volume 17, Addison Wesley, 1994, pp. 515–537.

[2] C. Baier, J.P. Katoen, Principles of Model Checking, The MIT Press, 2008.

[3] E. Bartocci, D. Cacciagrano, M. Di Berardini, E. Merelli, L. Tesei, Timed Operational Semantics and Well-Formedness of Shape Calculus, Scientific Annals of Computer Science 20 (2010) 33–52.

[4] E. Bartocci, F. Corradini, M. Di Berardini, E. Merelli, L. Tesei, Shape Calculus. A Spatial Mobile Calculus for 3D Shapes, Scientific Annals of Computer Science 20 (2010) 1–31.

[5] B. Becker, D. Beyer, H. Giese, F. Klein, D. Schilling, Symbolic invariant verification for systems with dynamic structural adaptation, in: Proceedings of the 28th international conference on Software engineering, ICSE '06, ACM, 2006, pp. 72–81.

[6] A. Bouchachia, N. Nedjah, Introduction to the special section on self-adaptive systems: Models and algorithms, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 7 (2012) 13.

[7] J. Bradbury, J. Cordy, J. Dingel, M. Wermelinger, A survey of self-management in dynamic software architecture specifications, in: Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems, ACM, pp. 28–33.

[8] R. Bruni, A. Bucchiarone, S. Gnesi, D. Hirsch, A.L. Lafuente, Graph-based design and analysis of dynamic software architectures, in: Concurrency, Graphs and Models, volume 6065 of *Lecture Notes in Computer Science*, Springer, 2008, pp. 37–56.

[9] R. Bruni, A. Corradini, F. Gadducci, A.L. Lafuente, A. Vandin, Adaptable transition systems, in: Recent Trends in Algebraic Development Techniques, volume 7841 of *Lecture Notes in Computer Science*, Springer, 2013, pp. 95–110.

[10] R. Bruni, A. Corradini, F. Gadducci, A. Lluch Lafuente, A. Vandin, A conceptual framework for adaptation, in: Fundamental Approaches to Software Engineering, volume 7212 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 240–254.

[11] R. Bruni, A. Lluch-Lafuente, U. Montanari, E. Tuosto, Style-based architectural reconfigurations, Bulletin of the European association for theoretical computer science 94 (2008) 161–180.

[12] B. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, et al., Software engineering for self-adaptive systems: A research roadmap, Software Engineering for Self-Adaptive Systems (2009) 1–26.

[13] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella, NuSMV 2: An opensource tool

for symbolic model checking, in: Computer Aided Verification, Springer, pp. 359–364.

[14] E. Clarke, E. Emerson, A. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications, ACM Transactions on Programming Languages and Systems (TOPLAS) 8 (1986) 244–263.

[15] E.M. Clarke, E.A. Emerson, Design and synthesis of synchronization skeletons using branching time temporal logic, in: Logic of Programs, number 131 in Lecture Notes in Computer Science, Springer-Verlag, 1981, pp. 52–71.

[16] E.M. Clarke, O. Grumberg, K.L. McMillan, X. Zhao, Efficient generation of counterexamples and witnesses in symbolic model checking, in: Proceedings of the 32nd annual ACM/IEEE Design Automation Conference, pp. 427–432.

[17] E.M. Clarke, O. Grumberg, D.A. Peled, Model Checking, The MIT press, 1999.

[18] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martı-Oliet, J. Meseguer, J.F. Quesada, Maude: Specification and programming in rewriting logic, Theoretical Computer Science 285 (2002) 187–243.

[19] F. Corradini, P. Inverardi, A. Wolf, On relating functional specifications to architectural specifications: a case study, Science of Computer Programming 59 (2006) 171–208.

[20] L. De Alfaro, T. Henzinger, Interface automata, ACM SIGSOFT Software Engineering Notes 26 (2001) 109–120.

[21] R. De Nicola, F. Vaandrager, Action versus state based logics for transition systems, Semantics of Systems of Concurrent Processes (1990) 407–419.

[22] N. D'Ippolito, V. Braberman, N. Piterman, S. Uchitel, The modal transition system control problem, in: FM 2012: Formal Methods, volume 7436 of *Lecture Notes in Computer Science*, 2012, pp. 155–170.

[23] A. Ehresmann, J. Vanbremeersch, Memory evolutive systems: hierarchy, emergence, cognition, volume 4, Elsevier Science, 2007.

[24] S. Feferman, Applications of many-sorted interpolation theorems, in: Proceedings of the Tarski Symposium (Proc. Sympos. Pure Math., Vol. XXV, Univ. of California, Berkeley, Calif., 1971), pp. 205–223.

[25] P. Feiler, R. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, L. Northrop, D. Schmidt, K. Sullivan, Ultra-large-scale systems: The software challenge of the future, Software Engineering Institute (2006).

[26] D. Harel, Statecharts: A visual formalism for complex systems, Science of computer programming 8 (1987) 231–274.

[27] D. Jackson, Software Abstractions: Logic, Language, and Analysis, The MIT Press, 2011.

[28] G. Karsai, A. Ledeczi, J. Sztipanovits, G. Peceli, G. Simon, T. Kovacshazy, An approach to self-adaptive software based on supervisory control, Self-adaptive software: applications (2003) 77–92.

[29] N. Khakpour, S. Jalili, M. Sirjani, U. Goltz, B. Abolhasanzadeh, HPob-SAM for modeling and analyzing IT ecosystems–through a case study, Journal of Systems and Software 85 (2012) 2770–2784.

[30] N. Khakpour, S. Jalili, C. Talcott, M. Sirjani, M. Mousavi, Formal modeling of evolving self-adaptive systems, Science of Computer Programming 78 (2011) 3–26.

[31] N. Khakpour, R. Khosravi, M. Sirjani, S. Jalili, Formal analysis of policy-based self-adaptive systems, in: Proceedings of the 2010 ACM Symposium on Applied Computing, ACM, 2010, pp. 2536–2543.

[32] J. Kramer, J. Magee, Self-managed systems: an architectural challenge, in: Future of Software Engineering, 2007. FOSE'07, IEEE, pp. 259–268.

[33] S. Kripke, Semantical considerations on modal logic, Acta Philosophica Fennica 16 (1963) 83–94.

[34] S. Kulkarni, K. Biyani, Correctness of component-based adaptation, Component-Based Software Engineering (2004) 48–58.

[35] R. Laddaga, Self-adaptive software, Technical Report 98-12, DARPA BAA, 1997.

[36] D. Le Métayer, Describing software architecture styles using graph grammars, Software Engineering, IEEE Transactions on 24 (1998) 521–533.

[37] F. Maraninchi, Y. Rémond, Mode-automata: a new domain-specific construct for the development of safe critical systems, Science of Computer Programming 46 (2003) 219–254.

[38] E. Merelli, N. Paoletti, L. Tesei, A multi-level model for self-adaptive systems, EPTCS 91 (2012) 112–126. *Proceedings of FOCLASA '12*.

[39] E. Merelli, M. Pettini, M. Rasetti, Topology driven modeling: the IS metaphor, Natural Computing DOI: 10.1007/s11047-014-9436-7 (2014).

[40] A. Pnueli, The temporal logic of programs, in: 18th IEEE Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 1977, pp. 46–67.

[41] K. Poyias, E. Tuosto, Enforcing architectural styles in presence of unexpected distributed reconfigurations, EPTCS 104 (2012) 67–82. *Proc. of ICE 2012*.

[42] F. Sagasti, A conceptual and taxonomic framework for the analysis of adaptive behavior, General systems 15 (1970) 151–160.

[43] M. Salehie, L. Tahvildari, Self-adaptive software: Landscape and research challenges, ACM Transactions on Autonomous and Adaptive Systems (TAAS) 4 (2009).

[44] M. Shin, Self-healing components in robust software architecture for concurrent and distributed systems, Science of Computer Programming 57 (2005) 27–44.

[45] M. Viroli, M. Casadei, S. Montagna, F. Zambonelli, Spatial coordination of pervasive services through chemical-inspired tuple spaces, ACM Transactions on Autonomous and Adaptive Systems 6 (2011).

[46] J. Zhang, B. Cheng, Model-based development of dynamically adaptive software, in: Proceedings of the 28th international conference on Software engineering, ACM, 2006, pp. 371–380.

[47] J. Zhang, H. Goldsby, B. Cheng, Modular verification of dynamically adaptive systems, in: Proceedings of the 8th ACM international conference on Aspect-oriented software development, ACM, 2009, pp. 161–172.

[48] Y. Zhao, D. Ma, J. Li, Z. Li, Model checking of adaptive programs with mode-extended linear temporal logic, in: Engineering of Autonomic and Autonomous Systems (EASe), 2011 8th IEEE International Conference and Workshops on, IEEE, pp. 40–48.

Appendix A. Proofs

Appendix A.1. Proposition 1 (Properties of flat semantics)
Proof.

(i + ii) Both the couples of rules Steady + AdaptStart and Steady + AdaptStartEnd ensure that there cannot exist a non-adapting state with both an outgoing non-adapting transition $\xrightarrow{r}$ and an outgoing adapting transition $\xrightarrow{r,\psi,r'}$. Indeed, the premises of the two rules, in both cases, are mutually exclusive by the fact that $(q \rightarrow_B q' \wedge q' \models L(r))$ is the negation of $\forall q''.(q \rightarrow_B q'' \implies q'' \not\models L(r))$.

(iii) Rules Adapt and AdaptEnd are the only ones producing an outgoing transition from an adapting state and none of them produces an r-labelled transition.

(iv) $(i) \wedge (ii) \wedge (iii) \implies (iv)$.

(v) Rule Adapt ensures that an adaptation transition is taken only if there are no other transitions that directly lead to the target S state. Indeed Adapt and AdaptEnd are mutually exclusive, thus avoiding adaptation steps to be taken when adaptation can end. This also holds for the successful adaptation paths (of length 1) obtained with the rule AdaptStartEnd, whose premises are not compatible with those of rule Adapt.

(vi) Let π be a generic infinite path of $\mathcal{F}(S[B])$ starting at a state $(q, r, \emptyset)$. Let $i \geq 0$ be a generic position in π, denoted $\pi[i]$, such that $\pi[i] = (q_i, r_i, \emptyset)$. Then, by properties (i)-(iv), there are only the following cases:

1. $(q_i, r_i, \emptyset) \overset{r_i}{\not\Rightarrow} \wedge (q_i, r_i, \emptyset) \overset{r_i,\psi,r'}{\Longrightarrow}$, i.e. the path stops at position $\pi[i] = (q_i, r_i, \emptyset)$; this contradicts the fact that π is infinite, thus this case never occurs along π;
2. $(q_i, r_i, \emptyset) \overset{r_i}{\rightarrow} (q_{i+1}, r_{i+1}, \emptyset)$; rule Steady was applied in the derivation of π;
3. $(q_i, r_i, \emptyset) \overset{r_i,\psi,r'}{\longrightarrow} (q_{i+1}, r_{i+1}, \emptyset)$; rule AdaptStartEnd was applied in the derivation of π;
4. $(q_i, r_i, \emptyset) \overset{r_i,\psi_i,r_{i+1}}{\longrightarrow} (q'_i, r_i, \{(\psi_i, r_{i+1})\})$; rule AdaptStart was applied in the derivation of π; at this point only the two mutually exclusive rules Adapt and AdaptEnd could be applied in the subsequent derivation of π, yielding only two possible sub-cases:
   (a) the derivation of π continues by zero or more applications of rule Adapt followed by one application of rule AdaptEnd, i.e.
   $(q_i, r_i, \emptyset)(\overset{r_i,\psi_i,r_{i+1}}{\longrightarrow})^{n_i}(q_{i+1}, r_{i+1}, \emptyset)$ for some $n_i > 0$;
   (b) the derivation of π continues with infinite applications of rule Adapt, i.e.

$$\begin{aligned}(q_i, r_i, \emptyset) &\overset{r_i,\psi_i,r_{i+1}}{\longrightarrow} (q'_i, r_i, \{\psi_i, r_{i+1}\}) \\ &\overset{r_i,\psi_i,r_{i+1}}{\longrightarrow} (q''_i, r_i, \{\psi_i, r_{i+1}\}) \\ &\overset{r_i,\psi_i,r_{i+1}}{\longrightarrow} \dots\end{aligned}$$

Thus, if for all steps $i \geq 0$ the case 4(b) does not occur then π is of kind **(1)**. On the contrary, if there exists $k \geq 0$ such that case 4(a) occurs at all steps $i < k$ and the case 4(b) occurs at step k, then π is of kind **(2)**.

(vii) By property *(vi)* it follows that the positions i in which $\pi[i] = (q_i, r_i, \emptyset)$ are those and only those in which the $S[B]$ system is in a steady state, i.e. it is either the first state f_0 in which by definition it holds $q_0 \models L(r_0)$, or it has been reached by using Rules Steady, AdaptEnd or StartAdaptEnd, which all explicitly check that $q_i \models L(r_i)$.

□

Appendix A.2. Proposition 2 (Union of Weak Adaptation Relations)

Proof. If $(q, r) \in \mathcal{R}_1 \cup \mathcal{R}_2$ then $(q, r) \in \mathcal{R}_1$ or $(q, r) \in \mathcal{R}_2$. Then it is possible to trivially verify all the conditions on (q, r) of Definition 7 using the same

proofs already available for $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively, by substituting, in these proofs, $\mathcal{R}_1$ and $\mathcal{R}_2$ with $\mathcal{R}_1 \cup \mathcal{R}_2$. □

Appendix A.3. Lemma 1 (Propagation of Weak Adaptation Relation)
Proof. If the path exists then, by property *(vi)* of Proposition 1, its form is of kind (1) or of kind (2). We show the existence of a path of kind (1), which equals to the one of the thesis. Moreover, we have to show that the weak adaptability propagates along the path, i.e. $\forall i \geq 0 \;.\; q_i|_w r_i$. We use induction to construct the infinite path of the right kind and to show the propagation. When $i = 0$, from $(q_0 = q, r_0 = r, \emptyset)$ we take the empty path and, by hypothesis, $q_0 = q|_w r_0 = r$. Moreover, the condition $\textsc{Progress}(q_0, r_0)$ guarantees that the path can continue. At the generic step $i > 0$ suppose by induction that $q_i|_w r_i$. Thus, there exists a weak adaptation relation $\mathcal{R}_i$ containing (q_i, r_i). In addition, the condition $\textsc{Progress}(q_i, r_i)$ guarantees that the path can continue. This implies the existence of transition(s) $(q_i, r_i, \emptyset)(\xrightarrow{r_i})^{m_i}(\xrightarrow{r_i,\psi_i,r_{i+1}})^{n_i}(q_{i+1}, r_{i+1}, \emptyset)$. There are two cases:

- $m_i = 1$. Thus $(q_i, r_i, \emptyset) \xrightarrow{r_i} (q', r_{i+1} = r_i, \emptyset)$ for some $q' \in B$. According to Definition 7 there is at least one B state q_{i+1} such that $\mathcal{R}_i$ contains $(q_{i+1}, r_{i+1} = r_i)$. Thus we can choose the transition with target $(q' = q_{i+1}, r_{i+1} = r_i, \emptyset)$ and have $q_{i+1}|_w r_{i+1}$.

- $m_i = 0$. Thus $(q_i, r_i, \emptyset) \xrightarrow{r_i,\psi'_i,r'_{i+1}}$ for some ψ'_i and r'_{i+1}. This is a transition leaving from a state q_i that is weak adaptive to r_i. Thus, by the definition of weak adaptation, there exists a state $(q_{i+1}, r_{i+1}, \emptyset)$ such that $(q_i, r_i, \emptyset)(\xrightarrow{r_i,\psi_i,r_{i+1}})^{n_i}(q_{i+1}, r_{i+1}, \emptyset)$ for some $n_i > 0$ and $(q_{i+1}, r_{i+1}) \in \mathcal{R}_i$, i.e., $q_{i+1}|_w r_{i+1}$. Note that condition *(iii)* of Definition 7 excludes the possibility that the adaptation path continues forever, which would yield an infinite path of kind (2).

□

Appendix A.4. Proposition 4 (Strong Adaptation implies Weak Adaptation)
Proof. Since $q|_s r$ then there exists a strong adaptation relation $\mathcal{R}$ such that $(q, r) \in \mathcal{R}$. We construct a weak adaptation relation $\mathcal{R}'$ containing (q, r), hence $q|_w r$. At the beginning we put (q, r) in $\mathcal{R}'$, then for some transition $(q, r, \emptyset) \xrightarrow{\tau} (q', r, \emptyset)$ we add (q, r'), which belongs to $\mathcal{R}$, also to $\mathcal{R}'$. If no $\xrightarrow{\tau}$ transitions are possible then, if $(q, r, \emptyset) \xrightarrow{r,\psi,r'}$, since q is strong adaptable

to r, we select one of the surely existing successor states $(q', r', \emptyset)$ such that $(q', r') \in \mathcal{R}$ and we add (q', r') to $\mathcal{R}'$. Then, we iterate this process for each new pair added in $\mathcal{R}'$. The process will terminate since the states are finite and the resulting $\mathcal{R}'$ will be, by construction, a weak adaptation relation. □

Appendix A.5. Lemma 2 (Propagation of Strong Adaptation Relation)

Proof. We construct inductively all possible finite paths from $(q, r, \emptyset)$ to any $(q', r', \emptyset) \in Post^*((q, r, \emptyset))$ showing that they are of the form:

$$\begin{aligned}&\pi = (q = q_0, r = r_0, \emptyset)(\xrightarrow{r_0})^{m_0}(\xrightarrow{r_0,\psi_0,r_1})^{n_0} \cdots \\ &\cdots (q_i, r_i, \emptyset)(\xrightarrow{r_i})^{m_i}(\xrightarrow{r_i,\psi_i,r_{i+1}})^{n_i}(q_{i+1}, r_{i+1}, \emptyset) \cdots \\ &\cdots (q_k = q', r_k = r', \emptyset)\end{aligned}$$

where $k \geq 0$ and for each i, $0 \leq i < k$, either $m_i = 1 \wedge n_i = 0$ or $m_i = 0 \wedge n_i > 0$. By the same induction we also prove that for all $i \leq k$, $q_i|_s r_i$. If $k = 0$, then $q_0 = q$ and $r_0 = r$, thus the thesis is trivially true. When $k > 0$ we assume by induction that we constructed, starting from $(q, r, \emptyset)$, all the possible finite paths, which are all of the form:

$$\begin{aligned}&\pi = (q = q_0, r = r_0, \emptyset)(\xrightarrow{r_0})^{m_0}(\xrightarrow{r_0,\psi_0,r_1})^{n_0} \cdots \\ &\cdots (q_i, r_i, \emptyset)(\xrightarrow{r_i})^{m_i}(\xrightarrow{r_i,\psi_i,r_{i+1}})^{n_i}(q_{i+1}, r_{i+1}, \emptyset) \cdots \\ &\cdots (q_{k-1}, r_{k-1}, \emptyset)\end{aligned}$$

where for each i, $0 \leq i < k$, either $m_i = 1 \wedge n_i = 0$ or $m_i = 0 \wedge n_i > 0$. Moreover we assume by induction that $q_i|_s r_i$ for all $i \leq k-1$. Since $q_{k-1}|_s r_{k-1}$, by condition (i) of Definition 9, $\textsc{Progress}(q_{k-1}, r_{k-1})$ holds. This implies that the paths of the form above cannot stop here. To continue the path from $(q_{k-1}, r_{k-1}, \emptyset)$ there are only the following possibilities:

1. $(q_{k-1}, r_{k-1}, \emptyset) \xrightarrow{r_{k-1}} (q_k, r_k, \emptyset)$; rule Steady can be applied, then $m_{k-1} = 1$ and $n_{k-1} = 0$; to prove that $q_k|_s r_k$ it is sufficient to take the same relation $\mathcal{R}$ that exists from the fact that $q_{k-1}|_s r_{k-1}$ because condition (ii) of Definition 9 implies that $q_k \mathcal{R} r_k$;
2. $(q_{k-1}, r_{k-1}, \emptyset) \xrightarrow{r_{k-1},\psi,r'} (q_k, r_k, \emptyset)$; rule AdaptStartEnd can be applied, then $m_{k-1} = 0$ and $n_{k-1} = 1$; also in this case, to prove that $q_k|_s r_k$ it is sufficient to take the same relation $\mathcal{R}$ that exists from the fact that $q_{k-1}|_s r_{k-1}$ because condition (iii) of Definition 9 implies that $q_k \mathcal{R} r_k$;

3. $(q_{k-1}, r_{k-1}, \emptyset) \xrightarrow{r_{k-1},\psi_{k-1},r_k} (q'_{k-1}, r_{k-1}, \{(\psi_{k-1}, r_k)\})$; rule AdaptStart can be applied; at this point only the two mutually exclusive rules Adapt and AdaptEnd can be applied in the subsequent derivation of the path, yielding only two possible sub-cases:
   (a) the derivation continues by zero or more applications of rule Adapt followed by one application of rule AdaptEnd, i.e. $(q_{k-1}, r_{k-1}, \emptyset)(\xrightarrow{r_{k-1},\psi_{k-1},r_k})^{n_{k-1}}(q_k, r_k, \emptyset)$ for some $n_{k-1} > 0$;
   (b) the derivation continues with infinite applications of rule Adapt, i.e.

$$\begin{aligned}(q_{k-1}, r_{k-1}, \emptyset) &\xrightarrow{r_{k-1},\psi_{k-1},r_k} (q'_{k-1}, r_{k-1}, \{\psi_{k-1}, r_k\}) \\ &\xrightarrow{r_{k-1},\psi_{k-1},r_k} (q''_{k-1}, r_{k-1}, \{\psi_{k-1}, r_k\}) \\ &\xrightarrow{r_{k-1},\psi_{k-1},r_k} \dots\end{aligned}$$

   Condition (*iii*) of Definition 9 implies that the case (b) cannot occur. Thus, in this case the path continues in any possible way of the form: $(q_{k-1}, r_{k-1}, \emptyset)(\xrightarrow{r_{k-1},\psi_{k-1},r_k})^{n_{k-1}}(q_k, r_k, \emptyset)$ with $m_{k-1} = 0$ and $n_{k-1} > 0$; also in this case, to prove that $q_k|_s r_k$ it is sufficient to take the same relation $\mathcal{R}$ that exists from the fact that $q_{k-1}|_s r_{k-1}$ because condition (*iii*) of Definition 9 implies that $q_k \; \mathcal{R} \; r_k$;

□

Appendix A.6. Proposition 5 (Construction of Strong Adaptation Relation)
Proof. If $S[B]$ is strong adaptable then $f_0 = (q_0, r_0, \emptyset)$ and $q_0|_s r_0$. By applying the propagation lemma of strong adaptation (Lemma 2), we get that all states $(q, r, \emptyset) \in Post^*(f_0)$ are such that $q|_s r$. Thus, we can use $\mathcal{R}_{q,r}$ to denote the strong adaptation relation, containing (q, r), that exists for each $(q, r, \emptyset) \in Post^*(f_0)$. Moreover, we naturally deduce that, for each such pair, $q \models L(r)$. Thus, if we take $\hat{\mathcal{R}} = \bigcup_{(q,r,\emptyset)\in Post^*(f_0)} \mathcal{R}_{q,r}$ we have, by definition of $\hat{\mathcal{R}}$ and of $\mathcal{R}$, $\mathcal{R} \subseteq \hat{\mathcal{R}}$. But $\mathcal{R}$ contains, by its definition, each possible pair (q, r) such that $(q, r, \emptyset)$ is reachable from f_0 and by the rule Steady of the operational semantics, $q \models L(r)$. Thus, it must also hold $\hat{\mathcal{R}} \subseteq \mathcal{R}$. Hence, $\mathcal{R} = \hat{\mathcal{R}}$. By Proposition 3, $\hat{\mathcal{R}} = \mathcal{R}$ is a strong adaptation relation.

For the converse, trivially, if $\mathcal{R} = \{(q, r) \in Q \times R \mid (q, r, \emptyset) \in Post^*(f_0)\}$ is a strong adaptation relation then $S[B]$ is strong adaptable, because, by definition, $(q_0, r_0) \in \mathcal{R}$.

□

Appendix A.7. Theorem 1 (Weak adaptability checking)

Proof. ($\Rightarrow$) Having $q \mid_w r$, by the propagation of weak adaptation (Lemma 1) we can construct an infinite path π in $\mathcal{F}(S[B])$, and thus in $\mathcal{K}(S[B])$, starting from $(q, r, \emptyset)$, which has the form specified in the Lemma and such that $q_i \mid_w r_i$ for all i. Such a path can be used to show that the given CTL formula is true. Since $q_i \mid_w r_i$, along π the *progress* proposition is true in all states. Moreover, whenever $m_i = 0$ and $n_i > 0$ in $(q_i, r_i, \emptyset)(\xrightarrow{r_i})^{m_i}(\xrightarrow{r_i,\psi_i,r_{i+1}})^{n_i}(q_{i+1}, r_{i+1}, \emptyset)$, the proposition *adapting* is true in state $(q_i, r_i, \emptyset)$. In this case we reach, by following the path, the state $(q_{i+1}, r_{i+1}, \emptyset)$ in which, since *progress* is true, then also *steady* is true. Note that *adapting* is true also in all the intermediate states between $(q_i, r_i, \emptyset)$ and $(q_{i+1}, r_{i+1}, \emptyset)$. Following the same path, from all these states the same target state $(q_{i+1}, r_{i+1}, \emptyset)$, in which *steady* is true, is reached. The cases in which $m_i = 1$ and $n_i = 0$ correspond to states in which *adapting* is false (by definition of *adapting* and by property (i) of Proposition 1), thus in this case the implication ($adapting \implies \mathbf{EF}\ steady$) is vacuously true.

($\Leftarrow$) If the formula is true at state $(q, r, \emptyset)$, by definition of the semantics of CTL we have that there exists an infinite path π in $\mathcal{K}(S[B])$ in which every state satisfies *progress* and the sub-formula ($adapting \implies \mathbf{EF}\ steady$). Such a path is a witness of the truth of the formula and can be calculated by a model checker usually in the form of a prefix followed by a cycle in which some reasonable[2] fairness constraints hold[3]. To show that $q \mid_w r$ we use such a path π to generate a weak adaptation relation $\mathcal{R}_\pi$ as follows:

$$\mathcal{R}_\pi = \{(q_i, r_i) \mid \exists i \geq 0 \colon \pi[i] = (q_i, r_i, \emptyset)\}$$

First, we note that $(q = q_0, r = r_0)$ is in $\mathcal{R}_\pi$ because $\pi[0] = (q = q_0, r = r_0, \emptyset)$. Then, we conclude the proof by showing, in the following, that $\mathcal{R}_\pi$ is indeed a weak adaptation relation. Let $i \geq 0$ and let $\pi[i] = (q_i, r_i, \emptyset)$. We check that for the generic pair $(q_i, r_i) \in \mathcal{R}_\pi$ all the conditions of the weak adaptability definition (Def. 7) hold:

[2]In our case a fairness condition that should be required for the weak adaptability checking can be expressed as follows:"if a state satisfying *steady* can be eventually reached infinitely many times, then it is eventually reached infinitely many times".

[3]For a detailed discussion on the fairness constraints in CTL and on the generation of witnesses we refer to [2, 16].

(i) $q_i \models L(r_i)$ holds by property (vii) of Proposition 1; $\textsc{Progress}(q_i, r_i)$ also holds because $\pi[i] = (q_i, r_i, \emptyset)$ is a state along an infinite path, thus it does not stop;

(ii) if in $\pi[i]$ we have that $(q_i, r_i, \emptyset) \xrightarrow{r}$ then we can take the transition $(q_i, r_i, \emptyset) \xrightarrow{r} (q_{i+1}, r_{i+1}, \emptyset)$ of π and then $(q_{i+1}, r_{i+1}) \in \mathcal{R}_\pi$;

(iii) if in $\pi[i]$ the $(q_i, r_i, \emptyset) \xrightarrow{r,\psi,r'}$ for some ψ and r', then $\pi[i] \models_{\text{CTL}} \textit{adapting}$ and $\pi[i] \models_{\text{CTL}} \textit{steady}$. Thus, the sub-formula ($\textit{adapting} \implies \mathbf{EF}\ \textit{steady}$) is immediately true in $\pi[i]$. However, we know that π continues and by property (v) of Proposition 1 we know that the semantics imposes that π adapts as soon as possible. Thus, there are two further sub-cases:

- the adaptation is immediate: $(q_i, r_i, \emptyset) \xrightarrow{r,\psi,r'} (q_{i+1}, r' = r_{i+1}, \emptyset)$ and thus $(q_{i+1}, r_{i+1}) \in \mathcal{R}_\pi$;
- the adaptation cannot be immediate, thus in π we have $(q_i, r_i, \emptyset) \xrightarrow{r,\psi,r'} (q', r_i, \{(\psi, r')\})$ for some $q' \in Q$, $\psi \in \Psi(\Sigma, A)$ and $r' \in R$. Again by the progress of π and by the definition of $\mathcal{K}(S[B])$, it holds that $(q', r_i, \{(\psi, r')\}) \models_{\text{CTL}} \textit{adapting}$ and that $(q', r_i, \{(\psi, r')\}) \not\models_{\text{CTL}} \textit{steady}$. Now, the sub-formula ($\textit{adapting} \implies \mathbf{EF}\ \textit{steady}$) is not immediately true and, by hypothesis, it holds in $(q', r_i, \{(\psi, r')\})$. Thus, there exists in $\mathcal{K}(S[B])$ a path starting from $(q', r_i, \{(\psi, r')\})$ and leading in j steps, $j > 0$, to a state in which *steady* holds, that is of the form $(q'', r', \emptyset)$. Among possibly others, the continuation of π until the next state of the form $(q'' = q_{i+1}, r' = r_{i+1}, \emptyset)$ is a finite path satisfying this condition. Indeed, if it were not, then π would either stop or infinitely continue along states that satisfy *adapting* but not *steady*. This contradicts the fact that π is a witness (under reasonable fairness conditions) of the truth of the original CTL formula. Thus, we have that $(q_i, r_i, \emptyset)(\xrightarrow{r_i,\psi_i,r_{i+1}})^{j+1}(q_{i+1}, r_{i+1}, \emptyset)$ (where $r_{i+1} = r'$ and $\psi_i = \psi$) and $(q_{i+1}, r_{i+1}) \in \mathcal{R}_\pi$.

□

Appendix A.8. Theorem 2 (Strong adaptability checking)

Proof. ($\Rightarrow$) Having $q \mid_s r$ we consider all paths π in $\mathcal{K}(S[B])$ starting from $(q, r, \emptyset)$, which have the form and the properties stated in the proof of Lemma 2. All such paths can be used to show that the given CTL formula is true by using the same argument used in the proof part ($\Rightarrow$) of Theorem 1, by turning the initial existential quantification into a universal one.

($\Leftarrow$) Also in this case the proof is similar to the proof part ($\Leftarrow$) of Theorem 1. However, the reasoning should be repeated not considering the witness of the truth of the formula, but a generic path π starting from $(q, r, \emptyset)$. Moreover, the strong adaptation relation $\mathcal{R}$ must be defined generalising on all paths:

$$\mathcal{R} = \{(q_i, r_i) \mid \exists \pi \in \mathit{Paths}((q, r, \emptyset)) \colon \exists i \geq 0 \colon \pi[i] = (q_i, r_i, \emptyset)\}$$

Then, the checking of the conditions of the strong adaptability definition (Def. 9) on every pair of $\mathcal{R}$ requires similar arguments to the proof of Theorem 1. □