Reliability Aspects of Uniformly Parameterised Cooperations

Peter Ochsenschläger and Roland Rieke
Fraunhofer Institute for Secure Information Technology, SIT
Darmstadt, Germany
Email: peter-ochsenschlaeger@t-online.de, roland.rieke@sit.fraunhofer.de

***Abstract*—In this paper, we examine reliability aspects of systems, which are characterised by the composition of a set of identical components. These components interact in a uniform manner, described by the schedules of the partners. Such kind of interaction is typical for scalable complex systems with cloud or grid structure. We call these systems "uniformly parameterised cooperations". We consider reliability of such systems in a possibilistic sense. This is formalised by always-eventually properties, a special class of liveness properties using a modified satisfaction relation, which expresses possibilities. As a main result, a finite state verification framework for uniformly parameterised reliability properties is given. The keys to this framework are structuring cooperations into phases and defining closed behaviours of systems. In order to verify reliability properties of such uniformly parameterised cooperations, we use finite state semi-algorithms that are independent of the concrete parameter setting.**

Keywords-reliability aspects of scalable complex systems; liveness properties; uniformly parameterised reliability properties; finite state verification; possibilistic reliability.

I. INTRODUCTION

The transition from systems composed of many isolated, small-scale elements to large-scale, distributed and massively interconnected systems is a key challenge of modern information and communications technologies. These systems need to be dependable, which means they need to remain secure, robust and efficient [1]. Examples for highly scalable systems comprise (i) grid computing architectures; and (ii) cloud computing platforms. In grid computing, large scale allocation issues relying on centralised controls present challenges that threaten to overwhelm existing centralised management approaches [1]. Cloud computing introduced the concept, to make software available as a service. This concept can only be successful, if certain obstacles such as reliability issues are solved [2]. In order to be able to model functional requirements of dependable systems best satisfying both fault-tolerance and security attributes, three distinct classes of (system specification) properties need to be considered, namely *safety*, *liveness*, and *information flow* [3]. Concrete reliability problems related to liveness properties range from replica selection to consistency of cloud storage (which allows multiple clients to access stored data concurrently in a consistent fashion) [4]. Most existing replica selection schemes rely on either central coordination (which has reliability, security, and scalability limitations) or distributed heuristics (which may lead to instability) [4]. Another important issue is, that clients of cloud services do not operate continuously, so clients should not depend on other clients for liveness of their operations [5].

In this paper, we consider systems that interact in a way that is typical for scalable complex systems. These systems, which we call *uniformly parameterised cooperations*, are characterised by (i) the composition of a set of identical components (copies of a two-sided cooperation); and (ii) the fact that these components interact in a uniform manner (described by the schedules of the partners). As an example of such uniformly parameterised systems of cooperations, e-commerce protocols can be considered. In these protocols, the two cooperation partners have to perform a certain kind of financial transactions. Such a protocol should work for several partners in the same manner, and the mechanism (schedule) to determine how one partner may be involved in several cooperations is the same for each partner. So, the cooperation is parameterised by the partners and the parameterisation should be uniform with respect to the partners.

Reliability is an important concept related to dependability, which ensures *continuity of correct service* [6]. In this paper, we consider reliability in a possibilistic sense, which means that correct services can be provided according to a certain pattern of behaviour again and again. These possibilities of providing correct services are expressed by a special class of liveness properties using a modified satisfaction relation. We call these properties *always-eventually properties*.

As a main result of the work presented, a finite state verification framework for *uniformly parameterised reliability properties* is given. The keys to this framework are structuring cooperations into phases and defining closed behaviours of systems. In this framework, *completion of phases strategies* and corresponding *success conditions* can be formalised [7], which produce finite state semi-algorithms that are independent of the concrete parameter setting. These algorithms are used to verify reliability properties of uniformly parameterised cooperations under certain regularity restrictions.

The paper is structured as follows. Section II gives an overview of the related work. In Section III, uniform parameterisations of two-sided cooperations in terms of formal

 ISBN: 978-1-61208-184-7

language theory is formalised. Section IV introduces the concept of uniformly parameterised reliability properties. The concept of structuring cooperations into phases given in Section V enables completion of phases strategies, which are described in Section VI. Consistent with this, corresponding success conditions can be formalised [2], which produce finite state semi-algorithms to verify reliability properties of uniformly parameterised cooperations. Finally, the paper ends with conclusions and an outlook in Section VII.

II. Related Work

System properties: A formal definition of safety and liveness properties is proposed in [8]. In [9], we defined a satisfaction relation, called *approximate satisfaction*, which expresses a possibilistic view on liveness and is equivalent to the satisfaction relation in [8] for safety properties. In this paper, we extended this concept (cf. Section IV) and defined *uniformly parameterised reliability properties*, which fit to the parameterised structure of the systems, which we consider here. Besides these safety and liveness properties so called "hyperproperties" [10] are of interest because they give formalisations for non-interference and non-inference.

Verification approaches for parameterised systems: An extension to the *Mur*φ verifier to verify systems with replicated identical components through a new data type called RepetitiveID is presented in [11]. A typical application area of this tool are cache coherence protocols. The aim of [12] is an abstraction method through symmetry, which works also when using variables holding references to other processes. This is not possible in Murφ. In [13], a methodology for constructing abstractions and refining them by analysing counter-examples is presented. The method combines abstraction, model-checking and deductive verification. However, this approach does not consider liveness properties. In [14], a technique for automatic verification of parameterised systems based on process algebra *CCS* [15] and the logic modal *mu-calculus* [16] is presented. This technique views processes as property transformers and is based on computing the limit of a sequence of mu-calculus formula generated by these transformers. The above-mentioned approaches demonstrate, that finite state methods combined with deductive methods can be applied to analyse parameterised systems. The approaches differ in varying amounts of user intervention and their range of application. A survey of approaches to combine model checking and theorem proving methods is given in [17].

Iterated shuffle products: In [18], it is shown that our definition of uniformly parameterised cooperations is strongly related to iterated shuffle products [19], if the cooperations are "structured into phases". The main concept for such a condition are shuffle automata [20] (multicounter automata [21]) whose computations, if they are deterministic, unambiguously describe how a cooperation partner is involved in several phases.

In [22], we have shown in particular that for self-similar parameterised systems $\mathscr{L}_{IK}$ the parameterised problem of verifying a *uniformly parameterised safety property* can be reduced to finite many fixed finite state problems.

Complementary to this, in the present paper, we define a uniformly parameterised reliability property based on this concept. The main result is a finite state verification framework for such *uniformly parameterised reliability properties*.

III. Parameterised Cooperations

The behaviour L of a discrete system can be formally described by the set of its possible sequences of actions. Therefore $L \subset \Sigma^*$ holds where Σ is the set of all actions of the system, and Σ^* (free monoid over Σ) is the set of all finite sequences of elements of Σ (words), including the empty sequence denoted by ε. $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$. Subsets of Σ^* are called formal languages [23]. Words can be composed: if u and v are words, then uv is also a word. This operation is called the *concatenation*; especially $\varepsilon u = u\varepsilon = u$. Concatenation of formal languages $U, V \subset \Sigma^*$ are defined by $UV := \{uv \in \Sigma^* | u \in U \text{ and } v \in V\}$. A word u is called a *prefix* of a word v if there is a word x such that $v = ux$. The set of all prefixes of a word u is denoted by $\text{pre}(u)$; $\varepsilon \in \text{pre}(u)$ holds for every word u. The set of possible continuations of a word $u \in L$ is formalised by the *left quotient* $u^{-1}(L) := \{x \in \Sigma^* | ux \in L\}$.

Infinite words over Σ are called ω-words [24]. The set of all infinite words over Σ is denoted Σ^ω. An ω-language L over Σ is a subset of Σ^ω. For $u \in \Sigma^*$ and $v \in \Sigma^\omega$ the *left concatenation* $uv \in \Sigma^\omega$ is defined. It is also defined for $U \subset \Sigma^*$ and $V \subset \Sigma^\omega$ by $UV := \{uv \in \Sigma^\omega | u \in U \text{ and } v \in V\}$.

For an ω-word w the prefix set is given by the formal language $\text{pre}(w)$, which contains every finite prefix of w. The prefix set of an ω-language $L \subset \Sigma^\omega$ is accordingly given by $\text{pre}(L) = \{u \in \Sigma^* |$ it exist $v \in \Sigma^\omega$ with $uv \in L\}$. For $M \subset \Sigma^*$ the *ω-power* $M^\omega \subset \Sigma^\omega$ is the set of all "infinite concatenations" of arbitrary elements of M. More formal definitions of theses ω-notions are given in the appendix.

Formal languages, which describe system behaviour, have the characteristic that $\text{pre}(u) \subset L$ holds for every word $u \in L$. Such languages are called *prefix closed*. System behaviour is thus described by prefix closed formal languages.

Different formal models of the same system are partially ordered with respect to different levels of abstraction. Formally, abstractions are described by so called alphabetic language homomorphisms. These are mappings $h^* : \Sigma^* \longrightarrow \Sigma'^*$ with $h^*(xy) = h^*(x)h^*(y)$, $h^*(\varepsilon) = \varepsilon$ and $h^*(\Sigma) \subset \Sigma' \cup \{\varepsilon\}$. So, they are uniquely defined by corresponding mappings $h : \Sigma \longrightarrow \Sigma' \cup \{\varepsilon\}$. In the following, we denote both the mapping h and the homomorphism h^* by h. Inverse homomorphism are denoted by h^{-1}. Let L be a language over the alphabet Σ'. Then $h^{-1}(L)$ is the set of words $w \in \Sigma^*$ such that $h(w) \in L$. In this paper, we consider a lot of alphabetic language homomorphisms. So, for simplicity, we

 ISBN: 978-1-61208-184-7

tacitly assume that a mapping between free monoids is an alphabetic language homomorphism if nothing contrary is stated.

To describe a two-sided cooperation, let $\Sigma = \Phi \cup \Gamma$ where Φ is the set of actions of cooperation partner F and Γ is the set of actions of cooperation partner G and $\Phi \cap \Gamma = \emptyset$. Now a prefix closed language $L \subset (\Phi \cup \Gamma)^*$ formally defines a two-sided cooperation.

Example 1. *Let* $\Phi = \{\mathrm{f_s}, \mathrm{f_r}\}$ *and* $\Gamma = \{\mathrm{g_r}, \mathrm{g_i}, \mathrm{g_s}\}$ *and hence* $\Sigma = \{\mathrm{f_s}, \mathrm{f_r}, \mathrm{g_r}, \mathrm{g_i}, \mathrm{g_s}\}$. *An example for a cooperation* $L \subset \Sigma^*$ *is now given by the automaton in Figure 1. It describes a simple handshake between F (client) and G (server), where a client may perform the actions* $\mathrm{f_s}$ *(send a request),* $\mathrm{f_r}$ *(receive a result) and a server may perform the corresponding actions* $\mathrm{g_r}$ *(receive a request),* $\mathrm{g_i}$ *(internal action to compute the result) and* $\mathrm{g_s}$ *(send the result).*

In the following, we will denote initial states by a short incoming arrow and final states by double circles. In this automaton, all states are final states, since L is prefix closed.

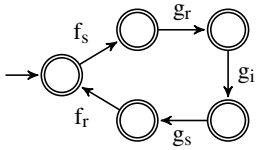


Figure 1. Automaton for 1-1-cooperation L

For parameter sets I, K and $(i,k) \in I \times K$ let Σ_{ik} denote pairwise disjoint copies of Σ. The elements of Σ_{ik} are denoted by a_{ik} and $\Sigma_{IK} := \bigcup_{(i,k)\in I\times K} \Sigma_{ik}$. The index ik describes the bijection $a \leftrightarrow a_{ik}$ for $a \in \Sigma$ and $a_{ik} \in \Sigma_{ik}$. Now $\mathscr{L}_{IK} \subset \Sigma_{IK}^*$ (prefix-closed) describes a *parameterised system*. To avoid pathological cases, we generally assume parameter and index sets to be non empty.

For a cooperation between one partner of type F with two partners of type G in Example 1 let

$$\Phi_{\{1\}\{1,2\}} = \{\mathrm{f_{s11}}, \mathrm{f_{r11}}, \mathrm{f_{s12}}, \mathrm{f_{r12}}\},$$
$$\Gamma_{\{1\}\{1,2\}} = \{\mathrm{g_{r11}}, \mathrm{g_{i11}}, \mathrm{g_{s11}}, \mathrm{g_{r12}}, \mathrm{g_{i12}}, \mathrm{g_{s12}}\} \text{ and}$$
$$\Sigma_{\{1\}\{1,2\}} = \Phi_{\{1\}\{1,2\}} \cup \Gamma_{\{1\}\{1,2\}}.$$

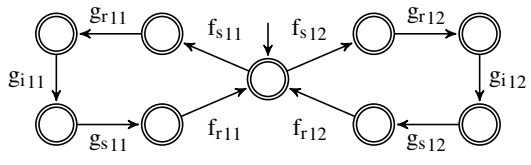


Figure 2. Automaton for 1-2-cooperation $\mathscr{L}_{\{1\}\{1,2\}}$

A 1-2-cooperation, where each pair of partners cooperates restricted by L and each partner has to finish the handshake it just is involved in before entering a new one, is now given (by reachability analysis) by the automaton in Figure 2 for $\mathscr{L}_{\{1\}\{1,2\}}$. It shows that one after another client 1 runs a handshake either with server 1 or with server 2. Figure 3 in contrast depicts an automaton for a 2-1-cooperation $\mathscr{L}_{\{1,2\}\{1\}}$ with the same overall number of partners involved but two of type F and one partner of type G. Figure 3 is more complex than Figure 2 because client 1 and client 2 may start a handshake independently of each other, but server 1 handles these handshakes one after another. A 5-3-cooperation with the same simple behaviour of partners already requires 194.677 states and 1.031.835 state transitions (computed by the SH verification tool [25]).

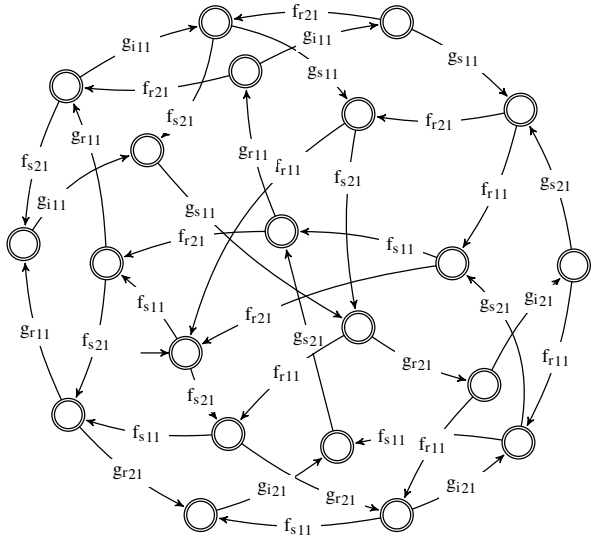


Figure 3. Automaton for the 2-1-cooperation $\mathscr{L}_{\{1,2\}\{1\}}$

For $(i,k) \in I \times K$, let $\pi_{ik}^{IK} : \Sigma_{IK}^* \to \Sigma^*$ with

$$\pi_{ik}^{IK}(a_{rs}) = \begin{cases} a \mid & a_{rs} \in \Sigma_{ik} \\ \varepsilon \mid & a_{rs} \in \Sigma_{IK} \setminus \Sigma_{ik} \end{cases}.$$

For *uniformly parameterised systems* $\mathscr{L}_{IK}$ we generally want to have

$$\mathscr{L}_{IK} \subset \bigcap_{(i,k)\in I\times K} ((\pi_{ik}^{IK})^{-1}(L))$$

because from an abstraction point of view, where only the actions of a specific Σ_{ik} are considered, the complex system $\mathscr{L}_{IK}$ is restricted by L.

In addition to this inclusion, $\mathscr{L}_{IK}$ is defined by *local schedules* that determine how each "version of a partner" can participate in different cooperations. More precisely, let $SF \subset \Phi^*$, $SG \subset \Gamma^*$ be prefix closed. For $(i,k) \in I \times$

K, let $\varphi_i^{IK} : \Sigma_{IK}^* \to \Phi^*$ and $\gamma_k^{IK} : \Sigma_{IK}^* \to \Gamma^*$ with

$$\varphi_i^{IK}(a_{rs}) = \begin{cases} a \mid & a_{rs} \in \Phi_{\{i\}K} \\ \varepsilon \mid & a_{rs} \in \Sigma_{IK} \setminus \Phi_{\{i\}K} \end{cases} \text{ and}$$

$$\gamma_k^{IK}(a_{rs}) = \begin{cases} a \mid & a_{rs} \in \Gamma_{I\{k\}} \\ \varepsilon \mid & a_{rs} \in \Sigma_{IK} \setminus \Gamma_{I\{k\}} \end{cases},$$

where Φ_{IK} and Γ_{IK} are defined correspondingly to Σ_{IK}.

Definition 1 (uniformly parameterised cooperation)**.** *Let I, K be finite parameter sets, then*

$$\mathscr{L}_{IK} := \bigcap_{(i,k)\in I\times K} (\pi_{ik}^{IK})^{-1}(L) \cap \bigcap_{i\in I} (\varphi_i^{IK})^{-1}(SF) \cap \bigcap_{k\in K} (\gamma_k^{IK})^{-1}(SG)$$

denotes a uniformly parameterised cooperation*.*

By this definition,

$$\mathscr{L}_{\{1\}\{1\}} = (\pi_{11}^{\{1\}\{1\}})^{-1}(L) \cap (\varphi_1^{\{1\}\{1\}})^{-1}(SF) \cap (\gamma_1^{\{1\}\{1\}})^{-1}(SG).$$

Because we want $\mathscr{L}_{\{1\}\{1\}}$ being isomorphic to L by the isomorphism $\pi_{11}^{\{1\}\{1\}} : \Sigma_{\{1\}\{1\}}^* \to \Sigma^*$, we additionally need

$$(\pi_{11}^{\{1\}\{1\}})^{-1}(L) \subset (\varphi_1^{\{1\}\{1\}})^{-1}(SF) \text{ and}$$
$$(\pi_{11}^{\{1\}\{1\}})^{-1}(L) \subset (\gamma_1^{\{1\}\{1\}})^{-1}(SG).$$

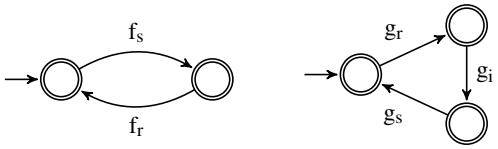
This is equivalent to $\pi_\Phi(L) \subset SF$ and $\pi_\Gamma(L) \subset SG$, where $\pi_\Phi : \Sigma^* \to \Phi^*$ and $\pi_\Gamma : \Sigma^* \to \Gamma^*$ are defined by

$$\pi_\Phi(a) = \begin{cases} a \mid & a \in \Phi \\ \varepsilon \mid & a \in \Gamma \end{cases} \text{ and } \pi_\Gamma(a) = \begin{cases} a \mid & a \in \Gamma \\ \varepsilon \mid & a \in \Phi \end{cases}.$$

So, we complete Def. 1 by the additional conditions

$$\pi_\Phi(L) \subset SF \text{ and } \pi_\Gamma(L) \subset SG.$$

Schedules SF and SG that fit to the cooperations given in Example 1 are depicted in Figs. 4(a) and 4(b). Here, we have $\pi_\Phi(L) = SF$ and $\pi_\Gamma(L) = SG$.



(a) Schedule SF (b) Schedule SG

Figure 4. Automata $\mathbb{SF}$ and $\mathbb{SG}$ for the schedules SF and SG

The system $\mathscr{L}_{IK}$ of cooperations is a typical example of a *complex system*. It consists of several identical components (copies of the two-sided cooperation L), which interact in a uniform manner (described by the schedules SF and SG and by the homomorphisms φ_i^{IK} and γ_k^{IK}).

Remark 1. *It is easy to see that $\mathscr{L}_{IK}$ is isomorphic to $\mathscr{L}_{I'K'}$ if I is isomorphic to I' and K is isomorphic to K'. More precisely, let $\iota_{I'}^{I} : I \to I'$ and $\iota_{K'}^{K} : K \to K'$ be bijections and let $\iota_{I'K'}^{IK} : \Sigma_{IK}^* \to \Sigma_{I'K'}^*$ be defined by*

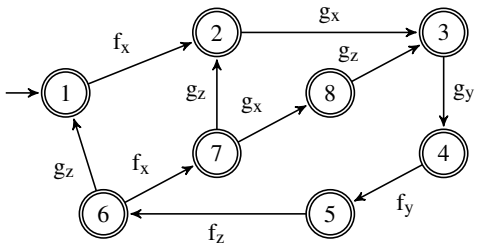
$$\iota_{I'K'}^{IK}(a_{ik}) := a_{\iota_{I'}^{I}(i)\iota_{K'}^{K}(k)} \text{ for } a_{ik} \in \Sigma_{IK}.$$

Hence, $\iota_{I'K'}^{IK}$ is a isomorphism and $\iota_{I'K'}^{IK}(\mathscr{L}_{IK}) = \mathscr{L}_{I'K'}$. The set of all these isomorphisms $\iota_{I'K'}^{IK}$ defined by corresponding bijections $\iota_{I'}^{I}$ and $\iota_{K'}^{K}$ is denoted by $\mathscr{I}_{I'K'}^{IK}$.

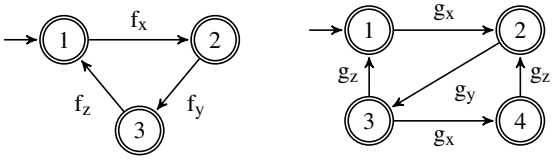
To illustrate the concepts of this paper, we consider the following example.

Example 2. *We consider a system of servers, each of them managing a resource, and clients, which want to use these resources. We assume that as a means to enforce a given* privacy policy *a server has to manage its resource in such a way that no client may access this resource while it is in use by another client (*privacy requirement*). This may be required to ensure* anonymity *in such a way that clients and their actions on a resource cannot be linked by an observer.*

We formalise this system at an abstract level, where a client may perform the actions f_x (send a request), f_y (receive a permission) and f_z (send a free-message), and a server may perform the corresponding actions g_x (receive a request), g_y (send a permission) and g_z (receive a free-message). The possible sequences of actions of a client resp. of a server are given by the automaton $\mathbb{SF}$ resp. $\mathbb{SG}$. The automaton $\mathbb{L}$ describes the 1-1-cooperation of one client and one server (see Figure 5). These automata define the client-server system $\mathscr{L}_{IK}$.



(a) 1-1-cooperation L



(b) Schedule SF (c) Schedule SG

Figure 5. Automata $\mathbb{L}$, $\mathbb{SF}$ and $\mathbb{SG}$ for Example 2

IV. A CLASS OF LIVENESS PROPERTIES

Usually, behaviour properties of systems are divided into two classes: *safety* and *liveness* properties [8]. Intuitively

a safety property stipulates that "something bad does not happen" and a liveness property stipulates that "something good eventually happens". In [8], both classes, as well as system behaviour, are formalised in terms of ω-languages, because especially for liveness properties infinite sequences of actions have to be considered.

Definition 2 (linear satisfaction)**.** *According to [8], a property E of a system is a subset of Σ^ω. If $S \subset \Sigma^\omega$ represents the behaviour of a system, then S* linearly satisfies *E iff $S \subset E$.*

In [8], it is furthermore shown that each property E is the intersection of a safety and a liveness property.

Safety properties $E_s \subset \Sigma^\omega$ are of the form $E_s = \Sigma^\omega \setminus F\Sigma^\omega$ with $F \subset \Sigma^*$, where F is the set of "bad things".

Liveness properties $E_l \subset \Sigma^\omega$ are characterised by $\mathrm{pre}(E_l) = \Sigma^*$. A typical example of a liveness property is

$$E_l = (\Sigma^* M)^\omega \text{ with } \emptyset \neq M \subset \Sigma^+. \tag{1}$$

This E_l formalises that "always eventually a finite action sequence $m \in M$ happens".

We describe system behaviour by prefix closed languages $B \subset \Sigma^*$. So, in order to apply the framework of [8], we have to transform B into an ω-language. This can be done by the *limit* $\lim(B)$ [24]. For prefix closed languages $B \subset \Sigma^*$, their limit is defined by

$$\lim(B) := \{w \in \Sigma^\omega | \mathrm{pre}(w) \subset B\}.$$

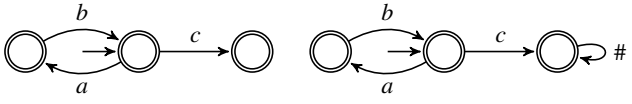
If B contains maximal words u (deadlocks), then these u are not captured by $\lim(B)$. Formally the set $\max(B)$ of all maximal words of B is defined by

$$\max(B) := \{u \in B | \text{ if } v \in B \text{ with } u \in \mathrm{pre}(v), \text{ then } v = u\}.$$

Now, using a dummy action #, B can be unambiguously described by

$$\hat{B} := B \cup \max(B)\#^* \subset \hat{\Sigma}^*,$$

where $\# \notin \Sigma$ and $\hat{\Sigma} := \Sigma \cup \{\#\}$. By this definition, in $\hat{B}$ the maximal words of B are continued by arbitrary many #'s. So, $\hat{B}$ does not contain maximal words.



(a) Automaton for B (b) Automaton for $\hat{B}$

Figure 6. Automata for B and $\hat{B}$

Let for example B be given by the automaton in Figure 6(a), then $\hat{B}$ is given by the automaton in Figure 6(b).

By this construction, we now can assume that system behaviour is formalised by prefix closed languages $\hat{B} \subset \Sigma^*\#^* \subset \hat{\Sigma}^*$ without maximal words, and the corresponding infinite system behaviour $S \subset \Sigma^\omega$ is given by $S := \lim(\hat{B})$.

For such an S and safety properties $E = \hat{\Sigma}^\omega \setminus F\hat{\Sigma}^\omega$ with $F \subset \hat{\Sigma}^*$ it holds

$$S \subset E \text{ iff } S \cap F\hat{\Sigma}^\omega = \emptyset \text{ iff } \mathrm{pre}(S) \cap F = \emptyset \text{ iff } \hat{B} \cap F = \emptyset.$$

If $F \subset \Sigma^*$, then $\hat{B} \cap F = \emptyset$ iff $B \cap F = \emptyset$. Therefore,

$$S \subset E \text{ iff } B \cap F = \emptyset \text{ for } F \subset \Sigma^*. \tag{2}$$

So, by (2) our approach in [22] is equivalent to the ω-notation of safety properties described by $F \subset \Sigma^*$.

Linear satisfaction (cf. Def. 2) is too strong for systems in our focus with respect to liveness properties, because $S = \lim(\hat{B})$ can contain "unfair" infinite behaviours, which are not elements of E.

Let for example $I \supset \{1,2\}$ and $K \supset \{1\}$, then $\lim(\widehat{\mathscr{L}_{IK}}) \cap \Sigma^\omega_{\{1\}\{1\}} \neq \emptyset$, which means that infinite action sequences exist, where only the partners with index 1 cooperate. So, if a property specification involves actions of a partner with index 2, as for instance $E = \Sigma^*_{IK}\Sigma_{\{2\}\{1\}}\Sigma^\omega_{IK}$, then this property is not linearly satisfied because $\lim(\widehat{\mathscr{L}_{IK}}) \not\subset E$.

Instead of neglecting such unfair infinite behaviours, we use a weaker satisfaction relation, called *approximate satisfaction*, which implicitly expresses some kind of fairness.

Definition 3 (approximate satisfaction)**.** *A system $S \subset \hat{\Sigma}^\omega$* approximately satisfies *a property $E \subset \hat{\Sigma}^\omega$ iff each finite behaviour (finite prefix of an element of S) can be continued to an infinite behaviour, which belongs to E. More formally, $pre(S) \subset pre(S \cap E)$.*

In [9], it is shown, that for safety properties linear satisfaction and approximate satisfaction are equivalent.

With respect to approximate satisfaction, liveness properties stipulate that "something good" eventually is possible.

Many practical liveness properties are of the form (1). Let us consider a prefix closed language $B \subset \Sigma^*$ and a formal language $\emptyset \neq M \subset \Sigma^+$. By definition 3 $\lim(\hat{B})$ approximately satisfies $(\hat{\Sigma}^* M)^\omega$ iff each $u \in B$ is prefix of some $v \in B$ with

$$v^{-1}(B) \cap M \neq \emptyset. \tag{3}$$

If B and M are regular sets, then (3) can be checked by usual automata algorithms [23] without referring to $\lim(\hat{B}) \cap (\hat{\Sigma}^* M)^\omega$.

Let us now consider the prefix closed language $L \subset \Sigma^*$ of example 2 and the "phase" $P \subset \Sigma^+$ given by the automaton $\mathbb{P}$ in Figure 7.

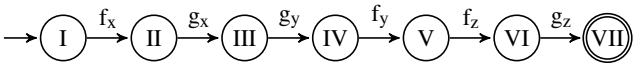


Figure 7. Automaton $\mathbb{P}$

$\lim(\hat{L})$ approximately satisfies the liveness property $(\hat{\Sigma}^* P)^\omega \subset \hat{\Sigma}^*$, because the automaton $\mathbb{L}$ in Figure 5(a) is strongly connected and $P \subset L$. (4)

(4) states that in the 1-1-cooperation $\lim(\hat{L})$ always eventually a "complete run through the phase P" is possible. This is a typical reliability property.

Properties of the form $(\hat{\Sigma}^* M)^\omega$ with $\emptyset \neq M \subset \Sigma^+$ we call *always-eventually properties*.

Let now $\emptyset \neq \mathring{M} \subset \Sigma^+_{\mathring{I}\mathring{K}}$ with fixed finite index sets $\mathring{I}$ and $\mathring{K}$. Then

$$(\hat{\Sigma}^*_{IK} \mathring{M})^\omega$$

is an always-eventually property for each finite index sets $I \supset \mathring{I}$ and $K \supset \mathring{K}$. Using bijections on $\mathring{I}$ and $\mathring{K}$ this can easily be generalised to each finite index sets I and K with $|I| \geq |\mathring{I}|$ and $|K| \geq |\mathring{K}|$, where $|I|$ denotes the cardinality of the set I. More precisely, let $\mathscr{I}^{\mathring{I}\mathring{K}}_{I'K'}$ be the set of all isomorphisms $\iota^{\mathring{I}\mathring{K}}_{I'K'} : \Sigma^*_{\mathring{I}\mathring{K}} \to \Sigma^*_{I'K'}$ generated by bijections $\iota^{\mathring{I}}_{I'} : \mathring{I} \to I'$ and $\iota^{\mathring{K}}_{K'} : \mathring{K} \to K'$ in such a way that

$$\iota^{\mathring{I}\mathring{K}}_{I'K'}(a_{ik}) := a_{\iota^{\mathring{I}}_{I'}(i)\iota^{\mathring{K}}_{K'}(k)}$$

for $a_{ik} \in \Sigma_{\mathring{I}\mathring{K}}$. Then

$$(\hat{\Sigma}^*_{IK} \iota^{\mathring{I}\mathring{K}}_{I'K'}(\mathring{M}))^\omega$$

is an always-eventually property for each $I \supset I'$, $K \supset K'$ and $\iota^{\mathring{I}\mathring{K}}_{I'K'} \in \mathscr{I}^{\mathring{I}\mathring{K}}_{I'K'}$. For finite index sets $\mathring{I}$, I, $\mathring{K}$ and K let

$$\mathscr{I}[(\mathring{I},\mathring{K}),(I,K)] := \bigcup_{I' \subset I, K' \subset K} \mathscr{I}^{\mathring{I}\mathring{K}}_{I'K'}.$$

Note that $\mathscr{I}[(\mathring{I},\mathring{K}),(I,K)] = \emptyset$ if $|\mathring{I}| > |I|$ or $|\mathring{K}| > |K|$.

Definition 4 (uniformly parameterised reliability property)**.** *Let $\mathring{I}$, I, $\mathring{K}$ and K be finite index sets with $|\mathring{I}| \leq |I|$ and $|\mathring{K}| \leq |K|$. If $\emptyset \neq \mathring{M} \subset \Sigma^+_{\mathring{I}\mathring{K}}$, then the family*

$$\mathscr{A}^{\mathring{M}}_{IK} := [(\hat{\Sigma}^*_{IK} \iota^{\mathring{I}\mathring{K}}_{I'K'}(\mathring{M}))^\omega]_{\iota^{\mathring{I}\mathring{K}}_{I'K'} \in \mathscr{I}[(\mathring{I},\mathring{K}),(I,K)]}.$$

is a strong uniformly parameterised always-eventually property *(uniformly parameterised reliability property).*

We say that $lim(\widehat{\mathscr{L}_{IK}})$ approximately satisfies *such a family $\mathscr{A}^{\mathring{M}}_{IK}$ iff $lim(\widehat{\mathscr{L}_{IK}})$ approximately satisfies each of the properties $(\hat{\Sigma}^*_{IK} \iota^{\mathring{I}\mathring{K}}_{I'K'}(\mathring{M}))^\omega$ for $\iota^{\mathring{I}\mathring{K}}_{I'K'} \in \mathscr{I}[(\mathring{I},\mathring{K}),(I,K)]$.*

Remark 2. *We use the adjective* strong*, because in [7] uniform parameterisations of general properties are defined, which, in case of always-eventually properties, are weaker than definition 4.*

Let us return to example 2 and let

$$\mathring{P} := (\pi^{\{1\}\{1\}}_{11})^{-1} P \subset \Sigma^+_{\{1\}\{1\}} \text{ and}$$
$$\mathring{E} := (\widehat{\Sigma_{\{1\}\{1\}}}^* \mathring{P})^\omega \subset \widehat{\Sigma_{\{1\}\{1\}}}^\omega. \tag{5}$$

Because $\pi^{\{1\}\{1\}}_{11} : \Sigma^*_{\{1\}\{1\}} \to \Sigma^*$ is an isomorphism, by (4) $\lim(\widehat{\mathscr{L}_{\{1\}\{1\}}})$ approximately satisfies $\mathring{E}$.

Now by definition 4 $\lim(\widehat{\mathscr{L}_{IK}})$ approximately satisfies $\mathscr{A}^{\mathring{P}}_{IK}$ iff in $\lim(\widehat{\mathscr{L}_{IK}})$ for each pair of clients and servers always eventually a complete run through a phase P is possible.

V. Cooperations based on phases

The schedule SG of example 2 shows that a server may cooperate with two clients partly in an interleaving manner. To formally capture such behaviour, cooperations are structured into phases [18]. This formalism is based on iterated shuffle products and leads to sufficient conditions for liveness properties (cf. Section VI).

Shuffling two words means arbitrarily inserting one word into the other word, like shuffling two decks of cards. In [21], this is formalised as follows:

A word $w \in \Sigma^*$ is called a *shuffle* of words $w_1, \ldots, w_m \in \Sigma^*$ if the positions of w can be coloured using m colors so that the positions with color $i \in \{1, \ldots, m\}$, when read from left to right, form the word w_i. *Shuffle of a set* $P \subset \Sigma^*$, is $\{w :$ w is a shuffle of some $w_1, \ldots, w_m \in P$, for some $m \in \mathbb{N}\}$.

However, we now provide an alternative formalisation, which is more adequate to the considerations in this paper.

Definition 5 (iterated shuffle product)**.** *Let $t \in \mathbb{N}$, and for each t let Σ_t be a copy of Σ. Let all Σ_t be pairwise disjoint. The index t describes the bijection $a \leftrightarrow a_t$ for $a \in \Sigma$ and $a_t \in \Sigma_t$ (which is equivalent to a colouring with color t in the formalism of [21]). Let*

$$\Sigma_{\mathbb{N}} := \bigcup_{t \in \mathbb{N}} \Sigma_t, \text{ and for each } t \in \mathbb{N}$$

let the homomorphisms $\tau^{\mathbb{N}}_t$ and $\Theta^{\mathbb{N}}$ be defined by

$$\tau^{\mathbb{N}}_t : \Sigma^*_{\mathbb{N}} \to \Sigma^* \text{ with } \tau^{\mathbb{N}}_t(a_s) = \left\{ \begin{array}{l|l} a & a_s \in \Sigma_t \\ \varepsilon & a_s \in \Sigma_{\mathbb{N}} \setminus \Sigma_t \end{array} \right. \text{ and}$$

$$\Theta^{\mathbb{N}} : \Sigma^*_{\mathbb{N}} \to \Sigma^* \text{ with } \Theta^{\mathbb{N}}(a_t) := a \text{ for } a_t \in \Sigma_t \text{ and } t \in \mathbb{N}.$$

The iterated shuffle product $P^{\sqcup\!\sqcup}$ *of P is now defined by*

$$P^{\sqcup\!\sqcup} := \Theta^{\mathbb{N}}[\bigcap_{t \in \mathbb{N}} (\tau^{\mathbb{N}}_t)^{-1}(P \cup \{\varepsilon\})] \text{ for } P \subset \Sigma^*.$$

It is easy to see that this is equivalent to the definition from [21] above. Let for example $P = \{ab\}$. Now, according to [21], the word $w = aabb$ is a shuffle of two words w_1, $w_2 \in P$ because two colors, namely 1 and 2, can be used to colour the word $aabb$ so that $w_1 = w_2 = ab \in P$. According to definition 5, $aabb \in P^{\sqcup\!\sqcup}$ because $aabb = \Theta^{\mathbb{N}}(a_1 a_2 b_2 b_1)$ and $\tau^{\mathbb{N}}_1(a_1 a_2 b_2 b_1) = \tau^{\mathbb{N}}_2(a_1 a_2 b_2 b_1) = ab \in P$ and $\tau^{\mathbb{N}}_t(a_1 a_2 b_2 b_1) = \varepsilon$ for $t \in \mathbb{N} \setminus \{1,2\}$.

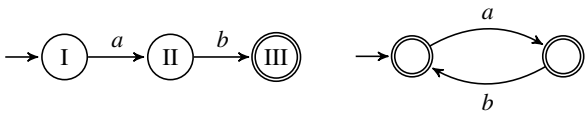
Following the ideas in [18], we structure cooperations into phases.

Definition 6 (based on a phase)**.** *A prefix closed language $B \subset \Sigma^*$ is based on a phase $P \subset \Sigma^*$, iff $B = \mathrm{pre}(P^{\sqcup\!\sqcup} \cap B)$.*

If B is based on P, then $B \subset \mathrm{pre}(P^{\sqcup\!\sqcup}) = (\mathrm{pre}(P))^{\sqcup\!\sqcup}$ and $B = \mathrm{pre}(P)^{\sqcup\!\sqcup} \cap B$.

Let for example $P = \{ab\}$ be given by the Automaton $\mathbb{P}$ in Figure 8(a) and B be given by the automaton $\mathbb{B}$ in Figure 8(b). Then $P^{\sqcup\!\sqcup} \cap B = \{ab\}^*$. This implies that B is based on P.

 ISBN: 978-1-61208-184-7



(a) Automaton $\mathbb{P}$ for P= {ab} (b) Automaton $\mathbb{B}$ for B

Figure 8. Automata $\mathbb{P}$ and $\mathbb{B}$

Generally, each B is based on infinitely many phases. If B is based on P, then B is based on P' for each $P' \supset P$. Each $B \subset \Sigma^*$ is based on Σ because $\Sigma^{\sqcup\!\sqcup} = \Sigma^*$. Figure 9 shows how we use phases to structure cooperations. The appropriate phases for our purposes as well as *closed behaviours* (words, in which all phases are completed) will be discussed in Section VI.

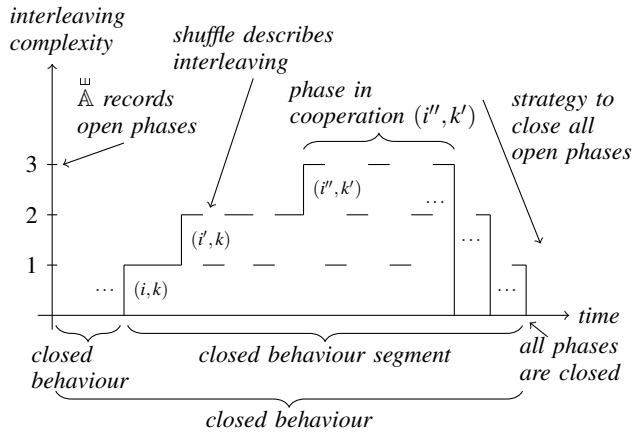


Figure 9. Phases and closed behaviours

We will now provide an automaton representation $\tilde{\mathbb{A}}$ for $P^{\sqcup\!\sqcup}$, which will illustrate "how a language B is based on a phase P". Let $P \subset \Sigma^*$ and $\mathbb{A} = (\Sigma, Q, \Delta, q_0, F)$ with $\Delta \subset Q \times \Sigma \times Q$, $q_0 \in Q$ and $F \subset Q$ be an (not necessarily finite) automaton that accepts P. To exclude pathological cases we assume $\varepsilon \notin P \neq \emptyset$. A consequence of this is in particular that $q_0 \notin F$. Let $\mathbb{N}_0^Q$ denote the set of all functions from Q in $\mathbb{N}_0$. For the construction of $\tilde{\mathbb{A}}$ the set $\mathbb{N}_0^Q$ plays a central role. In $\mathbb{N}_0^Q$ we distinguish the following functions:

$$0 \in \mathbb{N}_0^Q \text{ with } 0(x) = 0 \text{ for each } x \in Q,$$

and for $q \in Q$ the function

$$1_q \in \mathbb{N}_0^Q \text{ with } 1_q(x) = \begin{cases} 1 \mid & x = q \\ 0 \mid & x \in Q \setminus \{q\} \end{cases}.$$

As usual for numerical functions, a partial order as well as addition and partial subtraction are defined.

For $f, g \in \mathbb{N}_0^Q$ let $f \geqslant g$ iff $f(x) \geqslant g(x)$ for each $x \in Q$, $f + g \in \mathbb{N}_0^Q$ with $(f+g)(x) := f(x) + g(x)$ for each $x \in Q$, and for $f \geqslant g$, $f - g \in \mathbb{N}_0^Q$ with $(f-g)(x) := f(x) - g(x)$ for each $x \in Q$.

The key idea of $\tilde{\mathbb{A}}$ is, to record in the functions of $\mathbb{N}_0^Q$ how many open phases are in each state $q \in Q$ respectively. Its state transition relation $\tilde{\Delta}$ is composed of four subsets whose elements describe (a) the entry into a new phase, (b) the transition within an open phase, (c) the completion of an open phase, (d) the entry into a new phase with simultaneous completion of this phase. With these definitions we now define the *shuffle automaton* $\tilde{\mathbb{A}}$.

Definition 7 (shuffle automaton)**.**
The shuffle automaton $\tilde{\mathbb{A}} = (\Sigma, \mathbb{N}_0^Q, \tilde{\Delta}, 0, \{0\})$ *w.r.t.* $\mathbb{A}$ *is an automaton with infinite state set* $\mathbb{N}_0^Q$*, the initial state* 0*, which is the only final state and*

$$\begin{aligned}\tilde{\Delta} := &\{(f,a,f+1_p) \in \mathbb{N}_0^Q \times \Sigma \times \mathbb{N}_0^Q \mid \\ &\quad (q_0,a,p) \in \Delta \text{ and it exists } (p,x,y) \in \Delta\} \cup \\ &\{(f,a,f+1_p-1_q) \in \mathbb{N}_0^Q \times \Sigma \times \mathbb{N}_0^Q \mid \\ &\quad f \geqslant 1_q, (q,a,p) \in \Delta \text{ and it exists } (p,x,y) \in \Delta\} \cup \\ &\{(f,a,f-1_q) \in \mathbb{N}_0^Q \times \Sigma \times \mathbb{N}_0^Q \mid \\ &\quad f \geqslant 1_q, (q,a,p) \in \Delta \text{ and } p \in F\} \cup \\ &\{(f,a,f) \in \mathbb{N}_0^Q \times \Sigma \times \mathbb{N}_0^Q \mid (q_0,a,p) \in \Delta \text{ and } p \in F\}.\end{aligned}$$

Accepting of a word $w \in \Sigma^*$ *is defined as usual [23].*

Generally $\tilde{\mathbb{A}}$ is a non-deterministic automaton with an infinite state set. In the literature, such automata are called multicounter automata [21] and it is known that they accept the iterated shuffle products [26]. For our purposes, deterministic computations of these automata are very important. To analyse these aspects more deeply we use our own notation and proof of the main theorems. In [18], it is shown that $\tilde{\mathbb{A}}$ accepts $P^{\sqcup\!\sqcup}$.

Let for example $P = \{ab\}$ (cf. Figure 8(a)). Then the states $f : Q \to \mathbb{N}_0$ of the automaton $\tilde{\mathbb{P}}$ are described by the sets $\{(q,n) \in Q \times \mathbb{N}_0 | f(q) = n \neq 0\}$.

$$\emptyset \xrightarrow{a} \{(\mathrm{II},1)\} \xrightarrow{a} \{(\mathrm{II},2)\} \xrightarrow{b} \{(\mathrm{II},1)\} \xrightarrow{b} \emptyset$$

is the only computation of $aabb \in P^{\sqcup\!\sqcup}$ in $\tilde{\mathbb{P}}$; it is an accepting computation.

Example 3. *Let L be defined by the automaton $\mathbb{L}$ in Figure 5(a) and $P \subset \Sigma^+$ be defined by the automaton $\mathbb{P}$ in Figure 7, then $L \cap P^{\sqcup\!\sqcup}$ is accepted by the product automaton of $\mathbb{L}$ and $\tilde{\mathbb{P}}$ that is given in Figure 10.*

This automaton is strongly connected and isomorphic to $\mathbb{L}$ (without considering final states), which proves that L is based on phase P. The states $(7, \{(\mathrm{VI},1),(\mathrm{II},1)\})$ and $(8, \{(\mathrm{VI},1),(\mathrm{III},1)\})$ show that L is "in this states involved in two phases".

Note that this product automaton, as well as the product automaton in Figure 11(b) and 12(b), is finite and deterministic.

 ISBN: 978-1-61208-184-7

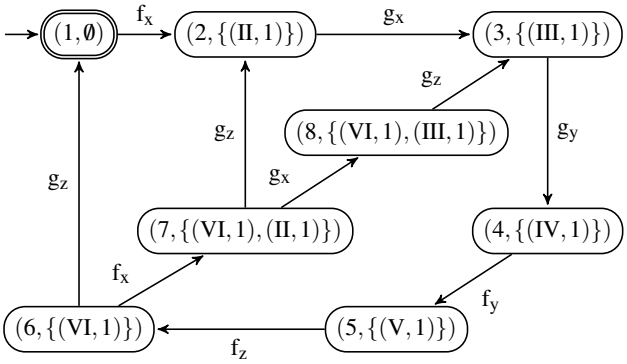


Figure 10. Product automaton of $\mathbb{L}$ and $\overset{\sqcup\!\sqcup}{\mathbb{P}}$

VI. Sufficient Conditions for a Class of Liveness Properties

The following definition is the key to sufficient conditions for strong uniformly parameterised always-eventually properties.

Definition 8 (set of closed behaviours)**.** *Let $B, M \subset \Sigma^*$. M is a* set of closed behaviours *of B, iff $x^{-1}(B) = B$ for each $x \in B \cap M$.*

In Figure 10, the initial state $(1,\emptyset)$ is the only final state of that strongly connected product automaton, so $P^{\sqcup\!\sqcup}$ is a set of closed behaviours of L.

Now, we get a sufficient condition for uniformly parameterised always-eventually properties.

Theorem 1. *Let I, K, $\mathring{I}$ and $\mathring{K}$ be finite index sets with $|\mathring{I}| \leq |I|$ and $|\mathring{K}| \leq |K|$. Let $\mathscr{L}_{IK}$ be a uniformly parameterised system of cooperations and let $\mathscr{C}_{IK} \subset \Sigma_{IK}^*$ be a set of closed behaviours of $\mathscr{L}_{IK}$, such that $\mathscr{L}_{IK} = pre(\mathscr{L}_{IK} \cap \mathscr{C}_{IK})$.*

If $lim(\widehat{\mathscr{L}_{\mathring{I}\mathring{K}}})$ approximately satisfies $(\widehat{\Sigma_{\mathring{I}\mathring{K}}^{*}}\, \mathring{M})^{\omega}$, with $\mathring{M} \subset \Sigma_{\mathring{I}\mathring{K}}^{+}$, then

$$lim(\widehat{\mathscr{L}_{IK}}) \text{ approximately satisfies } \mathscr{A}_{IK}^{\mathring{M}}.$$

For the proof of Theorem 1 see the appendix. The following theorem gives a set of closed behaviours of $\mathscr{L}_{IK}$.

Theorem 2. *Let $P^{\sqcup\!\sqcup}$ be a set of closed behaviours of L and let $\pi_\Phi(P^{\sqcup\!\sqcup})$ resp. $\pi_\Gamma(P^{\sqcup\!\sqcup})$ be a set of closed behaviours of SF resp. SG, then*

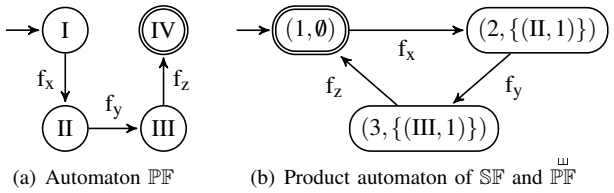
$$\mathscr{C}_{IK} := \bigcap_{(i,k)\in I\times K} (\pi_{ik}^{IK})^{-1}(P^{\sqcup\!\sqcup})$$

is a set of closed behaviours of $\mathscr{L}_{IK}$.

Theorem 2 is proven in [7]. We now show that $\pi_\Phi(P^{\sqcup\!\sqcup})$ is a set of closed behaviours of SF, which is given in Figure 5(b). The automaton $\mathbb{PF}$ in Figure 11(a) is the minimal automaton of $\pi_\Phi(P) \subset \Phi^+$.

By Theorem 3, which is given in the appendix

$$SF \cap \pi_\Phi(P^{\sqcup\!\sqcup}) = SF \cap (\pi_\Phi(P))^{\sqcup\!\sqcup}.$$

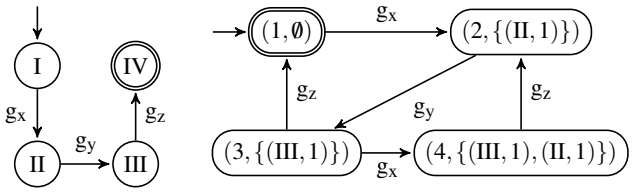


(a) Automaton $\mathbb{PF}$ (b) Product automaton of $\mathbb{SF}$ and $\overset{\sqcup\!\sqcup}{\mathbb{PF}}$

Figure 11. Automaton $\mathbb{PF}$ and product automaton of $\mathbb{SF}$ and $\overset{\sqcup\!\sqcup}{\mathbb{PF}}$

So, $SF \cap \pi_\Phi(P^{\sqcup\!\sqcup})$ is accepted by the product automaton of $\mathbb{SF}$ and $\overset{\sqcup\!\sqcup}{\mathbb{PF}}$ that is depicted in Figure 11(b). By the same argument as for the product automaton of $\mathbb{L}$ and $\overset{\sqcup\!\sqcup}{\mathbb{P}}$ SF is based on $\pi_\Phi(P)$, and $\pi_\Phi(P^{\sqcup\!\sqcup})$ is a set of closed behaviours of SF.

Likewise, the automaton $\mathbb{PG}$ in Figure 12(a) is the minimal automaton of $\pi_\Gamma(P) \subset \Gamma^+$, $SG \cap \pi_\Gamma(P^{\sqcup\!\sqcup})$ is accepted by the product automaton of $\mathbb{SG}$ and $\overset{\sqcup\!\sqcup}{\mathbb{PG}}$ in Figure 12(b), SG is based on $\pi_\Gamma(P)$, and $\pi_\Gamma(P^{\sqcup\!\sqcup})$ is a set of closed behaviours of SG.



(a) Automaton $\mathbb{PG}$ (b) Product automaton of $\mathbb{SG}$ and $\overset{\sqcup\!\sqcup}{\mathbb{PG}}$

Figure 12. Automaton $\mathbb{PG}$ and product automaton of $\mathbb{SG}$ and $\overset{\sqcup\!\sqcup}{\mathbb{PG}}$

So, by Figure 10, 11(b) and 12(b) all assumptions of Theorem 2 are fulfilled. (6)

Now to apply Theorem 1 together with Theorem 2 it remains to find conditions such that each $u \in \mathscr{L}_{IK}$ is prefix of some $v \in \mathscr{L}_{IK} \cap \mathscr{C}_{IK}$. This set of closed behaviours $\mathscr{C}_{IK}$ consists of all words $w \in \Sigma_{IK}^*$, in which all phases are completed.

Considering example 2, we have shown that each phase is initiated by an F-action (Figure 7), each F-partner is involved in at most one phase (Figure 11(b)), and, each G-partner is involved in at most two phases (Figure 12(b)).

Now to construct for each $u \in \mathscr{L}_{IK}$ a $v \in \mathscr{L}_{IK} \cap \mathscr{C}_{IK}$ with $u \in \text{pre}(v)$ one may imagine that the following strategy could work.

1) For each G-partner involved in two phases, complete one of this phases.
2) For each G-partner involved in one phases, complete this phase.
3) Complete the phases, where only an F-partner is involved in.

If L is based on P, SF based on $\pi_\Phi(P)$ and SG based on $\pi_\Gamma(P)$, then by Theorem 3 the assumptions of Theorem 2 imply $L = \text{pre}(L \cap P^{\sqcup\!\sqcup})$, $SF = \text{pre}(SF \cap \pi_\Phi(P^{\sqcup\!\sqcup}))$ and $SG = \text{pre}(SG \cap \pi_\Gamma(P^{\sqcup\!\sqcup}))$.

This is in [7] the starting point of a more general form of such a "completion (of phases) strategy", where also "success conditions" for that strategy are given. It is shown, that under certain regularity restrictions these conditions can be verified by semi-algorithms based on finite state methods. These restrictions are:

The product automata as in Figure 10, 11(b) and 12(b) must be finite and deterministic. (7)

We only get semi-algorithms but no algorithms, because the product automata are constructed step by step and this procedure does not terminate if the corresponding product automaton is not finite.

Using (7), (3) and the Theorems 1 and 2, the approximate satisfaction of uniformly parameterised always-eventually properties can be *verified by semi-algorithms based on finite state methods*. This verification method only depends on L, SF, SG, P and $\mathring{M}$ and *doesn't refer to the general index sets I and K*.

In [7], it is shown that the success conditions are fulfilled in example 2. So, by (4), Theorem 1, Theorem 2 and (6) in example 2 $\lim(\widehat{\mathscr{L}_{IK}})$ approximately satisfies $\mathscr{A}_{IK}^{\mathring{P}}$ for each finite index sets I and K, where $\mathring{P}$ is defined in (5).

VII. Conclusions and Future Work

The main result of this paper is a finite state verification framework for *uniformly parameterised reliability properties*. The uniformly parameterisation of reliability properties exactly fits to the scalability and reliability issues of complex systems and systems of systems, which are characterised by the composition of a set of identical components, interacting in a uniform manner described by the schedules of the partners.

In this framework, the concept of structuring cooperations into phases enables completion of phases strategies. Consistent with this, corresponding success conditions can be formalised [7], which produce finite state semi-algorithms (independent of the concrete parameter setting) to verify reliability properties of uniformly parameterised cooperations. The next step should be to integrate these semi-algorithms in our SH verification tool [25].

Furthermore, we plan a generalisation of the presented approach to systems whose global behaviour is composed of behavioural patterns. The aim is, to eventually derive a set of construction principles for reliable parameterised systems.

Another future work perspective is the application of the approach presented in this paper to the Security Modeling Framework (SeMF) [27]. In SeMF, beside system behaviour, also local views of agents and agents knowledge about system behaviour are considered.

Acknowledgement

Roland Rieke developed the work presented here in the context of the project MASSIF (ID 257475) being co-funded by the European Commission within FP7.

Appendix

A. Basic Notations

The set of all infinite words over Σ is defined by

$$\Sigma^\omega = \{(a_i)_{i\in\mathbb{N}} | a_i \in \Sigma \text{ for each } i \in \mathbb{N}\},$$

where $\mathbb{N}$ denotes the set of natural numbers. On Σ^ω a *left concatenation* with words from Σ^* is defined. Let $u = b_1 \dots b_k \in \Sigma^*$ with $k \geq 0$ and $b_j \in \Sigma$ for $1 \leq j \leq k$ and $w = (a_i)_{i\in\mathbb{N}} \in \Sigma^\omega$ with $a_i \in \Sigma$ for all $i \in \mathbb{N}$, then $uw = (x_j)_{j\in\mathbb{N}} \in \Sigma^\omega$ with $x_j = b_j$ for $1 \leq j \leq k$ and $x_j = a_{j-k}$ for $k < j$. For $w \in \Sigma^\omega$ the prefix set $\text{pre}(w) \subset \Sigma^*$ is defined by $\text{pre}(w) = \{u \in \Sigma^* |$ it exists $v \in \Sigma^\omega$ with $uv = w\}$. For $L \subset \Sigma^*$ the *ω-language* $L^\omega \subset \Sigma^\omega$ is defined by $L^\omega = \{(a_i)_{i\in\mathbb{N}} \in \Sigma^\omega |$ it exists a strict monotonically increasing function $f : \mathbb{N} \to \mathbb{N}$ with $a_1 \dots a_{f(1)} \in L$ and $a_{f(i)+1} \dots a_{f(i+1)} \in L$ for each $i \in \mathbb{N}\}$. $f : \mathbb{N} \to \mathbb{N}$ is called *strict monotonically increasing* if $f(i) < f(i+1)$ for each $i \in \mathbb{N}$.

B. Proof of Theorem 1

To prove Theorem 1 the following lemma is needed.

Lemma 1.

$$\mathscr{L}_{IK} \supset \mathscr{L}_{I'K'} \text{ for } I' \times K' \subset I \times K.$$

For the proof of Lemma 1 see [18] (proof of Theorem 1).

Proof: Proof of Theorem 1.

If $\lim(\widehat{\mathscr{L}_{\mathring{I}\mathring{K}}})$ approximately satisfies $(\widehat{\Sigma_{\mathring{I}\mathring{K}}}^* \mathring{M})^\omega$, then by (3) (with $u = \varepsilon$) there exists $v \in \mathscr{L}_{\mathring{I}\mathring{K}}$ with $v^{-1}(\mathscr{L}_{\mathring{I}\mathring{K}}) \cap \mathring{M} \neq \emptyset$. As $\iota_{I'K'}^{\mathring{I}\mathring{K}}$ is an isomorphism

$$\iota_{I'K'}^{\mathring{I}\mathring{K}}(\mathscr{L}_{\mathring{I}\mathring{K}}) = \mathscr{L}_{I'K'} \text{ and}$$
$$(\iota_{I'K'}^{\mathring{I}\mathring{K}}(v))^{-1}(\mathscr{L}_{I'K'}) \cap \iota_{I'K'}^{\mathring{I}\mathring{K}}(\mathring{M}) \neq \emptyset. \quad (8)$$

As $\mathscr{C}_{IK}$ is a set of closed behaviours of $\mathscr{L}_{IK}$ and each $u \in \mathscr{L}_{IK}$ is prefix of some $v \in \mathscr{L}_{IK} \cap \mathscr{C}_{IK}$, there exists $x \in u^{-1}(\mathscr{L}_{IK})$ with $(ux)^{-1}(\mathscr{L}_{IK}) = \mathscr{L}_{IK}$.

By Lemma 1 $\mathscr{L}_{IK} \supset \mathscr{L}_{I'K'}$ for each $I' \subset I$ and $K' \subset K$, so $(ux)^{-1}(\mathscr{L}_{IK}) \supset \mathscr{L}_{I'K'}$.

Now (8) implies

$$(\iota_{I'K'}^{\mathring{I}\mathring{K}}(v))^{-1}((ux)^{-1}(\mathscr{L}_{IK})) \cap \iota_{I'K'}^{\mathring{I}\mathring{K}}(\mathring{M}) \neq \emptyset. \quad (9)$$

As $(\iota_{I'K'}^{\mathring{I}\mathring{K}}(v))^{-1}((ux)^{-1}(\mathscr{L}_{IK})) = (ux\iota_{I'K'}^{\mathring{I}\mathring{K}}(v))^{-1}(\mathscr{L}_{IK})$, (9) and (3) complete the proof of Theorem 1. ■

C. Homomorphism Theorem for $P^{\sqcup\!\sqcup}$

Theorem 3 (homomorphism theorem for $P^{\sqcup\!\sqcup}$)**.**
Let $\mu : \Sigma^* \to \Sigma'^*$ be an alphabetic homomorphism, then holds

$$\mu(P^{\sqcup\!\sqcup}) = (\mu(P))^{\sqcup\!\sqcup}.$$

For the proof of Theorem 3 see [7] (proof of Theorem 6).

REFERENCES

[1] S. Bullock and D. Cliff, "Complexity and emergent behaviour in ICT systems," Hewlett-Packard Labs, Tech. Rep. HP-2004-187, 2004.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.

[3] Z. Benenson, F. Freiling, T. Holz, D. Kesdogan, and L. Penso, "Safety, liveness, and information flow: Dependability revisited," in *Proceedings of the 4th ARCS International Workshop on Information Security Applications*, pp. 56–65.

[4] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "Donar: decentralized server selection for cloud services," in *Proceedings of the ACM SIGCOMM 2010 conference on SIGCOMM*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 231–242.

[5] A. Shraer, C. Cachin, A. Cidon, I. Keidar, Y. Michalevsky, and D. Shaket, "Venus: verification for untrusted cloud storage," in *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, ser. CCSW '10. New York, NY, USA: ACM, 2010, pp. 19–30.

[6] A. Avizienis, J.-C. Laprie, B. Randell, and C. E. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Trans. Dependable Sec. Comput.*, vol. 1, no. 1, pp. 11–33, 2004.

[7] P. Ochsenschläger and R. Rieke, "Behaviour Properties of Uniformly Parameterised Cooperations," Fraunhofer SIT, Tech. Rep. SIT-TR-2010/2, 2010.

[8] B. Alpern and F. B. Schneider, "Defining Liveness," *Information Processing Letters*, vol. 21, no. 4, pp. 181–185, October 1985.

[9] U. Nitsche and P. Ochsenschläger, "Approximately Satisfied Properties of Systems and Simple Language Homomorphisms," *Information Processing Letters*, vol. 60, pp. 201–206, 1996.

[10] M. R. Clarkson and F. B. Schneider, "Hyperproperties," *Computer Security Foundations Symposium, IEEE*, vol. 0, pp. 51–65, 2008.

[11] C. N. Ip and D. L. Dill, "Verifying Systems with Replicated Components in Murφ," *Formal Methods in System Design*, vol. 14, no. 3, pp. 273–310, 1999.

[12] F. Derepas and P. Gastin, "Model Checking Systems of Replicated Processes with SPIN," in *Proceedings of the 8th International SPIN Workshop on Model Checking Software (SPIN'01)*, ser. Lecture Notes in Computer Science, M. B. Dwyer, Ed., vol. 2057. Toronto, Canada: Springer, May 2001, pp. 235–251.

[13] Y. Lakhnech, S. Bensalem, S. Berezin, and S. Owre, "Incremental Verification by Abstraction." in *TACAS*, ser. Lecture Notes in Computer Science, T. Margaria and W. Yi, Eds., vol. 2031. Springer, 2001, pp. 98–112.

[14] S. Basu and C. R. Ramakrishnan, "Compositional analysis for verification of parameterized systems," *Theor. Comput. Sci.*, vol. 354, no. 2, pp. 211–229, 2006.

[15] R. Milner, *Communication and Concurrency*, ser. International Series in Computer Science. NY: Prentice Hall, 1989.

[16] J. C. Bradfield and C. Stirling, "Modal Logics and Mu-Calculi: An Introduction," in *Handbook of Process Algebra*, J. A. Bergstra, A. Ponse, and S. A. Smolka, Eds. Elsevier Science, 2001, ch. 1.4.

[17] T. E. Uribe, "Combinations of Model Checking and Theorem Proving," in *FroCoS '00: Proceedings of the Third International Workshop on Frontiers of Combining Systems*. London, UK: Springer-Verlag, 2000, pp. 151–170.

[18] P. Ochsenschläger and R. Rieke, "Uniform Parameterisation of Phase Based Cooperations," Fraunhofer SIT, Tech. Rep. SIT-TR-2010/1, 2010.

[19] M. Jantzen, "Extending Regular Expressions with Iterated Shuffle," *Theor. Comput. Sci.*, vol. 38, pp. 223–247, 1985.

[20] J. Jedrzejowicz and A. Szepietowski, "Shuffle languages are in P," *Theor. Comput. Sci.*, vol. 250, no. 1-2, pp. 31–53, 2001.

[21] H. Björklund and M. Bojanczyk, "Shuffle Expressions and Words with Nested Data," in *Mathematical Foundations of Computer Science 2007*, 2007, pp. 750–761.

[22] P. Ochsenschläger and R. Rieke, "Security Properties of Self-similar Uniformly Parameterised Systems of Cooperations," in *Proceedings of the 19th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP)*. IEEE Computer Society, February 2011.

[23] J. Sakarovitch, *Elements of Automata Theory*. Cambridge University Press, 2009.

[24] D. Perrin and J.-E. Pin, *Infinite Words*. Elsevier, 2004, vol. Pure and Applied Mathematics Vol 141.

[25] P. Ochsenschläger, J. Repp, and R. Rieke, "The SH-Verification Tool," in *Proc. 13th International FLorida Artificial Intelligence Research Society Conference (FLAIRS-2000)*. Orlando, FL, USA: AAAI Press, May 2000, pp. 18–22.

[26] J. Jedrzejowicz, "Structural Properties of Shuffle Automata," *Grammars*, vol. 2, no. 1, pp. 35–51, 1999.

[27] A. Fuchs, S. Gürgens, and C. Rudolph, "Towards a Generic Process for Security Pattern Integration," in *Trust, Privacy and Security in Digital Business, 6th International Conference, TrustBus 2009, Linz, Austria, September 3–4, 2009, Proceedings*. Springer, 2009.

 ISBN: 978-1-61208-184-7