Effective Verification of Systems with a Dynamic Number of Components

Pavlína VařekovÆ [*], Pavel Moravec [†], Ivana ČernÆ [*], Barbora Zimmerova [*]
Faculty of Informatics
Masaryk University
602 00 Brno, Czech Republic
{xvareko1, xmoravec, cerna, zimmerova}@fi.muni.cz

ABSTRACT

In the paper, we present a novel approach to verification of dynamic component-based systems, the systems that can have a changing number of components over their life-time. We focus our attention on systems with a stable part (called *provider*) and a number of dynamic components of one type (called *clients*) because dynamic systems can be often decomposed into segments like this. Our method for verification of such systems is based on determining a number k of dynamic components, such that if a system is proved correct for any number lower than k, it is consequently correct for an arbitrarily large number of dynamic components. The paper aims not only in proving the propositions that state this, it concentrates also on bounding the set of dynamic systems and verifiable properties in a way, that k is relatively small and thus practically interesting. In addition to this, we present an algorithm for computing k.

Categories and Subject Descriptors

D.2.4 [**Software Engineering**]: Software/Program Verification

General Terms

Component-based systems, software verification

Keywords

Component-based systems, dynamic number of components, finite-state systems, formal verification

1. INTRODUCTION

The difficulty of formal verification of *dynamic component-based systems*, the systems that can have changing number of components over their life-time, is one of the most discussed issues these days in CBSE. It simply follows from the fact that if the number of components in a dynamic system is not bounded (it can grow over all bounds), the system model is infinite in general.

Our approach to the dynamic systems verification is based on two simple observations. First, the dynamic components[1] use to respect a *common type*, including a common template of their behaviour. Second, dynamic systems are often *not infinite by nature*. If we interpreted a dynamic system as a collection of its instances, each with a fixed number of dynamic components of each type, the instances would be finite-state, or at least have natural finite-state models. However their number would be infinite, which is the core of the problem.

In this paper, we focus on dynamic systems with a stable part (called *provider*) and a number of dynamic components of one type (called *clients*) because dynamic systems can be often decomposed into parts with this structure. The crucial idea of our approach to verification of a set of properties on such systems can be presented as follows. As the dynamic components share a common behaviour, there exists a number $k \in \mathbb{N}_0$ such that: *if the properties hold on the system with m dynamic components for any $m \leq k$, they also hold on the system with more then k dynamic components.* In the paper, we define such k, prove this statement true for it, and design an algorithm for computing it.

In particular, the value k for a dynamic system and a set of properties can be estimated as a sum of two measures. The first one is a measure of complexity of a dynamic system reflecting the maximal number of clients that are regarded by the provider. The second one is a similar measure on properties that reflects the minimal number of clients necessary to exhibit a path violating studied properties. In the paper, we focus on LTL-like properties, and we are interested only in those properties whose violation involves a limited number of components.

As an underlying formalism for this work, we use the *Component-interaction automata* language [11]. However the basic idea of this contribution is very general and the reasoning presented here is also applicable to other formalisms that

[*]The authors have been supported by the grant No. 1ET400300504.
[†]The author has been partially supported by the Grant Agency of Czech Republic grant No. 102/05/H050.

Sixth International Workshop on Specification and Verification of Component-Based Systems (SAVCBS 2007), September 3-4, 2007, Cavtat near Dubrovnik, Croatia.

[1]The components that are dynamically added and removed at run-time; analogy to instances of a type.

model systems as finite-state LTS or regular-like expressions.

The paper is structured as follows. First, we briefly discuss related work in Section 2. Then, Section 3 outlines the basic definitions of the Component-interaction automata language, and Section 4 formally defines dynamic systems that are of our interest. Section 5 designates the set of properties that are appropriate for verification using our approach, and it proves several propositions that are crucial for verification of dynamic systems. Section 6 is dedicated to the algorithm for finding the value k discussed above, and the text is closed with a conclusion and discussion of future work in Section 7.

2. RELATED WORK

As far as we are concerned, the idea presented above has not been elaborated yet. However there are other approaches that end up with finite models of potentially infinite dynamic systems. In [1], the author presents his solution to creating a finite model of a component that may communicate to arbitrary number of clients. It is assumed that even if generally the number of dynamic components connected to the component can be arbitrary, during the assembly phase, concrete number of the components is known. The author does not consider dynamic creation and removal of components at run-time. Another approach is studied in [10] where systems are first modelled as infinite-state and then reduced to finite-state by adjusted verification technique. In [7], the authors also use the technique of state-space reduction, in this case for verification of Java programs.

An alternative to state-space reduction is the verification of infinite-state models. This is discussed for instance in [2]. However these techniques are very time consuming and often do not guarantee to finish.

3. COMPONENT-INTERACTION AUTOMATA LANGUAGE

Component-interaction automata [5] are a specification language for modelling of component interactions in hierarchical component-based software systems. They capture each component as a labelled transition system with structured labels and a hierarchy of component names. The basic definitions are briefly reminded in this section. For more details, see [11].

Definition 3.1. A *hierarchy of component names*[2] is a tuple $H = (H_1, \ldots, H_m)$ of one of the following forms, where S_H denotes the set of component names corresponding to H. The first case is that $H_1, \ldots, H_m$ are pairwise different natural numbers or strings over the Greek alphabet (denoted $\mathbb{A}$ within the text); then $S_H = \bigcup_{i=1}^{m} \{H_i\}$. The second case is that $H_1, \ldots, H_m$ are hierarchies of component names where $S_{H_1}, \ldots, S_{H_m}$ are pairwise disjoint; then $S_H = \bigcup_{i=1}^{m} S_{H_i}$.

A *component-interaction automaton* (or a *CI automaton* for short) is a 5-tuple $\mathcal{C} = (Q, Act, \delta, I, H)$ where Q is a finite set of states, Act is a finite set of *actions*, $\Sigma = ((S_H \cup \{-\}) \times Act \times (S_H \cup \{-\})) \setminus (\{-\} \times Act \times \{-\})$ is a set of *labels*, $\delta \subseteq Q \times \Sigma \times Q$ is a finite set of *labelled transitions*, $I \subseteq Q$ is a nonempty set of *initial states*, and H is a hierarchy of component names.

The labels are triplets of a form $(-,a,o')$,$(o,a,-)$, or (o,a,o') and accordingly are of the type *input*, *output*, or *internal* respectively.
- The input label $(-,a,o')$ represents that the component o' receives the action a as an input.
- The output label $(o,a,-)$ represents that the component o sends the action a as an output.
- The internal label (o,a,o') represents that the component o sends the action a as an output and synchronously the component o' receives the action a as an input.

Examples of CI automata and their hierarchies of component names are in Figure 2 and Figure 4.

Definition 3.2. A *path* of a CI automaton $\mathcal{C} = (Q, Act, \delta, I, H)$ is an alternating sequence of states and labels given by δ that is either infinite, or is finite in case that it ends with a deadlock state. $Path(\mathcal{C})$ is the set of all paths of the CI automaton $\mathcal{C}$, $Path^{Init}(\mathcal{C})$ is its subset containing all paths starting in an initial state and $Path^{Init}_{Inf}(\mathcal{C})$ is the set of all infinite paths from $Path^{Init}(\mathcal{C})$.

If $\pi = q_0, l_0, q_1, l_1, \ldots$ is a (finite or infinite) path of the CI automaton $\mathcal{C}$, then
- $\mathcal{L}(\pi, i)$ is the i-th label of π (starting from $i = 0$) if there is any; here $\mathcal{L}(\pi, i) = l_i$,
- $\pi(i)$ is the i-th state of π if it exists; here $\pi(i) = q_i$ and
- π^i is the i-th suffix of π; here $\pi^i = q_i, l_i, \ldots$.

Notation. *For a given CI automaton $\mathcal{C}$ we denote $\mathcal{L}_\mathcal{C}$ the set of all labels reachable from an initial state in $\mathcal{C}$, $\mathcal{L}_{int,\mathcal{C}}$ the set of all internal labels reachable in $\mathcal{C}$.*

Definition 3.3. We say that a set of component-interaction automata $\{(Q_i, Act_i, \delta_i, I_i, H_i)\}_{i\in\mathcal{I}}$ is *composable* if $\mathcal{I} \subseteq \mathbb{N}$ is finite and $(H_i)_{i\in\mathcal{I}}$ is a hierarchy of component names.

Let $\mathcal{S} = \{(Q_i, Act_i, \delta_i, I_i, H_i)\}_{i\in\mathcal{I},\mathcal{I}\subseteq\mathbb{N}}$ be a finite composable set of component-interaction automata. Then the *complete transition space* for $\mathcal{S}$, written $\Delta_\mathcal{S}$, is a set of transitions among product states from $\Pi_{i\in\mathcal{I}} Q_i$ such that each transition reflects that either one of the automata from $\mathcal{S}$ follows its original transition and the others wait, or two automata synchronise on complementary labels $(o,a,-)$ and $(-,a,o')$, where $o, o' \in \mathbb{N} \cup \mathbb{A}$, and it forms a new label (o,a,o'). For precise definition see the appendix.

Let $\mathcal{S} = \{(Q_i, Act_i, \delta_i, I_i, H_i)\}_{i\in\mathcal{I}}$ be a composable set of CI automata and $\mathcal{F} \supseteq \bigcup_{i\in\mathcal{I}} \mathcal{L}_{int,\mathcal{C}_i}$ be a set of (feasible) labels. Then $\otimes^\mathcal{F}$ is a *composition operator with respect to feasible labels* which for the set $\mathcal{S}$ determines the CI automaton $\otimes^\mathcal{F}\mathcal{S} = (\Pi_{i\in\mathcal{I}} Q_i, \cup_{i\in\mathcal{I}} Act_i, \delta, \Pi_{i\in\mathcal{I}} I_i, (H_i)_{i\in\mathcal{I}})$ such that $\delta = \{(q, x, q') \in \Delta_\mathcal{S} \mid x \in \mathcal{F}\}$.

We say that the automaton $\otimes^\mathcal{F}\{\mathcal{C}_i\}_{i\in\mathcal{I}}$ is *defined* iff $\{\mathcal{C}_i\}_{i\in\mathcal{I}}$ is a composable set of CI automata and $\mathcal{F} \supseteq \bigcup_{i\in\mathcal{I}} \mathcal{L}_{int,\mathcal{C}_i}$.

Definition 3.4. Let $\mathcal{L}$ be a set of labels, S a set of component names. Then $Comm(\mathcal{L}, S)$ is a set of the labels $\mathcal{L}$ together with the internal labels that follow from $\mathcal{L}$ after communication with components whose names are in the set S.

[2]As distinct to the original definition, the component names do not need to be natural numbers, they can also be strings over Greek alphabet. This technicality does not influence any other definitions.

$$Comm(\mathcal{L},S) = \mathcal{L} \cup \{_{(o,a,o')} \mid {}_{(o,a,-)} \in \mathcal{L} \wedge o' \in S\} \cup \{_{(o,a,o')} \mid {}_{(-,a,o')} \in \mathcal{L} \wedge o \in S\}$$

Example 3.1. For $\mathcal{L} = \{_{(-,act_1,\alpha)},{}_{(\beta,act_2,-)},{}_{(\alpha,act_3,\beta)}\}$ holds: $Comm(\mathcal{L},\{1,2\}) = \mathcal{L} \cup \{_{(1,act_1,\alpha)},{}_{(\beta,act_2,1)},{}_{(2,act_1,\alpha)},{}_{(\beta,act_2,2)}\}$.

3.1 The logic for specifying properties

In formal verification techniques, like *model checking* [6], the properties for verification are specified in temporal logics. In our approach, we use the logic CI-LTL [12]. CI-LTL is an extension of the action-based LTL [9], which is in addition able to express that a given action is enabled in a state of a path (one-step branching).

Definition 3.5. Let $\mathcal{L}$ be a set of labels of CI automata, then CI-LTL formulas over $\mathcal{L}$ are defined inductively:
1) If $l \in \mathcal{L}$, then $\mathcal{P}(l)$ and $\mathcal{E}(l)$ are formulas.
2) If ϕ and ψ are formulas, then $\phi \wedge \psi, \neg\,\phi, \mathcal{X}\,\phi$ and $\phi\ \mathcal{U}\ \psi$ are formulas.
3) Every formula can be obtained by a finite number of applications of previous two steps.

Let $\mathcal{C} = (Q, Act, \delta, I, H)$ be a CI automaton, then CI-LTL formulas are interpreted over the paths $\pi \in Path^{Init}_{Inf}(\mathcal{C})$ where the satisfaction relation $\models$ is defined inductively:

$$\begin{aligned}
\pi &\models \mathcal{E}(l) && \iff \exists q \in Q : \pi(0) \xrightarrow{l} q\\
\pi &\models \mathcal{P}(l) && \iff \mathcal{L}(\pi,0) = l\\
\pi &\models \varphi \wedge \psi && \iff \pi \models \varphi \text{ and } \pi \models \psi\\
\pi &\models \neg\,\varphi && \iff \pi \not\models \varphi\\
\pi &\models \mathcal{X}\,\varphi && \iff \pi^1 \models \varphi\\
\pi &\models \varphi\ \mathcal{U}\ \psi && \iff \exists j \in \mathbb{N}_0 : \pi^j \models \psi \text{ and}\\
& && \quad \forall k \in \mathbb{N}_0, k < j : \pi^k \models \phi
\end{aligned}$$

Other operators can be defined as shortcuts: $\varphi \vee \psi \equiv \neg\,(\neg\,\varphi \wedge \neg\,\psi)$, $\varphi \Rightarrow \psi \equiv \neg\,(\varphi \wedge \neg\,\psi)$, $\mathcal{F}\,\varphi \equiv \text{true}\ \mathcal{U}\ \varphi$, $\mathcal{G}\,\varphi \equiv \neg\,\mathcal{F}\,\neg\,\varphi$.

Notation. *Let φ be a CI-LTL formula then $\mathcal{L}_\varphi$ is a set of labels that occur in the formula.*

4. DYNAMIC SYSTEM MODEL

To simplify the explication of our approach, we narrow our attention to the dynamic systems consisting of a number of dynamic clients which are connected to one provider that represents the stable part of the system. We assume that the clients have the same behaviour, and that they do not communicate with each other (illustrated in Figure 1). This is natural in any system where the clients may be added and removed dynamically.

Additionally, we focus only on the systems that fulfil that: (1) the system has only one type of clients, and (2) each client is modelled as an automaton with the hierarchy of component names (i). Note that the previous conditions do not significantly restrict the number of systems we can deal with. Renaming and system partitioning usually helps to transform a system into the permissible one.

In the definition below, the description is reflected as follows. The first bullet states that the components constituting the provider are named with strings over Greek alphabet.

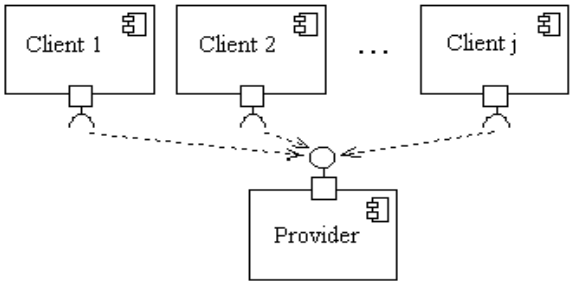


Figure 1: A dynamic system with j clients.

The second bullet reflects that all the clients have the same models up to the names of modelled components, where i-th client models a component with a numerical name i. The third, fourth and fifth bullet restrict the transitions allowed in the composition of the automata. The third bullet states that no two clients may communicate with each other, the fourth assures that all clients are handled equally by the provider, and the fifth care for the composition to be defined.

Definition 4.1. Let $\mathcal{C}_0 = (Q_0, Act_0, \delta_0, I_0, H_0)$ and $\mathcal{C}_i = (Q, Act, \delta_i, I, (i))$, where $i \in \mathbb{N}$, be CI automata and $\mathcal{F}$ be a set of labels. The tuple $(\mathcal{C}_0, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F})$ is a *CI model of a dynamic system* (or a *dynamic system model* for short) iff:
- $S_{H_0} \subseteq \mathbb{A}$,[3]
- for all $i \in \mathbb{N}$ the CI automaton, which results from $\mathcal{C}_i$ after renaming of all component names with $r : \{i\} \to \{1\}$, is equal to the CI automaton $\mathcal{C}_1$,
- $\mathcal{F} \cap \{_{(i,act,j)} \mid i,j \in \mathbb{N}, i \neq j\} = \emptyset$,
- each permutation p of the set $\mathbb{N} \cup \mathbb{A} \cup \{-\}$, which is the identity on the set $\mathbb{A} \cup \{-\}$ fulfils:
$$\mathcal{F} = \{_{(p(o_1),a,p(o_2))} \mid {}_{(o_1,a,o_2)} \in \mathcal{F}\},$$
- $\mathcal{F} \supseteq \mathcal{L}_{int,\mathcal{C}_0} \cup \bigcup_{i\in\mathbb{N}} \mathcal{L}_{int,\mathcal{C}_i}$.

Notation. *For a dynamic system model $\mathcal{D} = (\mathcal{C}_0, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F})$ the automaton $\mathcal{C}_0$ is called* provider, *CI automata $\mathcal{C}_1, \mathcal{C}_2, \ldots$ are called* clients.

For the rest of the paper let us fix that if $\mathcal{D}$ is a dynamic system, then it denotes the tuple:
$(\mathcal{C}_0 = (Q_0, Act_0, \delta_0, I_0, H_0), \{\mathcal{C}_i = (Q, Act, \delta_i, I, (i))\}_{i\in\mathbb{N}}, \mathcal{F})$.

Definition 4.2. Let $\mathcal{D}$ be a dynamic system and $n \in \mathbb{N}_0$, then $\mathcal{D}_n = \otimes^{\mathcal{F}}\{\mathcal{C}_i\}_{0\le i\le n}$ is the CI automaton modelling system $\mathcal{D}$ with n clients and $\mathcal{L}_{\mathcal{D}} = \bigcup_{i\in\mathbb{N}_0} \mathcal{L}_{\mathcal{D}_i}$ is the set of all labels reachable in any of the automata $\{\mathcal{D}_i\}_{i\in\mathbb{N}_0}$.

Example 4.1. Let us consider a simple example of a dynamic system model depicted in Figure 2 capturing a simple system consisting of a database and its clients.

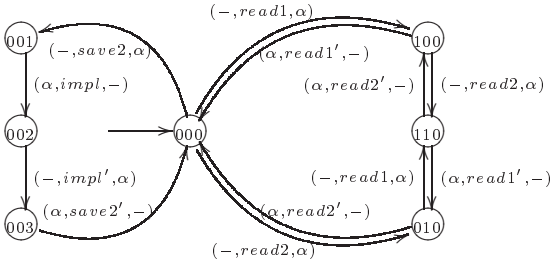
The database (modelled by the component $\mathcal{C}_0$) provides three types of services: $read1$, $read2$ and $save2$ which correspond to standard reading or saving of data in a database. Each of the services is modelled by a tuple of actions, the first of them, e.g. $_{(-,read1,\alpha)}$, represents a receiving of a request of the service and the second, e.g. $_{(\alpha,read1',-)}$, models the response, which indicates that the service was finished. Services $read1$ and $read2$ can be executed in parallel, but $save2$

[3] $\mathbb{A}$ is the set of strings over Greek alphabet

can not. Because of space reasons, services $read1$ and $read2$ are modelled without implementation details. Service $save2$ is modelled as a service that can be provided only after a connection of a component that implements the service.

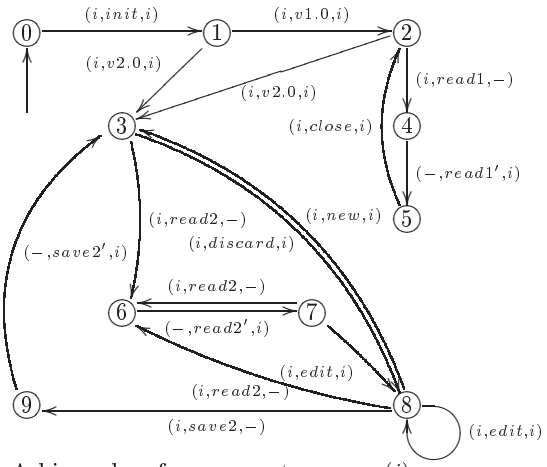
Clients (modelled by components $\mathcal{C}_i$, $i \in \mathbb{N}$) can use three previously mentioned types of services. They first initialise their system and after that they choose one of two versions of the software. If they choose version 1.0, they can only read data from the database or upgrade their software, else they can read data, edit them and save the changes.

- $\mathcal{C}_0$:



A hierarchy of component names: (α)

- $\mathcal{C}_i$, where $i \in \mathbb{N}$:



A hierarchy of component names: (i)

- $\mathcal{F} = \bigcup_{i\in\mathbb{N}} \mathcal{L}_{int,\mathcal{C}_i} \cup \{(\alpha,impl,-),(-,impl',\alpha)\} \cup \{(i,read2,\alpha),(\alpha,read2',i),(i,save2,\alpha),(\alpha,save2',i),(i,read1,\alpha),(\alpha,read1',i) \mid i \in \mathbb{N}\}$.

Figure 2: A CI model of dynamic system $\mathcal{DB} = (\mathcal{C}_0, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F})$.

5. VERIFICATION

In this section, we study verification of the properties whose violation involves at most a finite number of clients. We denote m the minimal number of clients that are necessary for the violation. For an arbitrary integer m and a dynamic system model $\mathcal{D}$ the set of such properties is denoted $Property(\mathcal{D}, m)$. A property for $m = 2$ can for example state that *whenever a dynamic component outputs a request for service a, no other dynamic component gets the response for a before the previous component does.* If the dynamic components behave the same and do not communicate with each other, it really suffices to observe only two of them to check this property.

In this section, we first discuss when we can say that a provider *regards at most n* clients. After definition of this term, we show that if a provider in $\mathcal{D}$ regards at most n clients, then given a set of properties $Property(\mathcal{D}, m)$ it holds that if we verify all of these on the dynamic system with $0, 1, \ldots, n+m$ clients, then the properties hold on the dynamic system *with any finite number of clients.*

5.1 Essential definitions and lemmas

For each dynamic system model $\mathcal{D}$ and each set of labels $X \subseteq \mathcal{L}_{\mathcal{C}_0}$ we define the value n from the previous paragraph. Then we prove some basic properties that the value n fulfils.

The sub-section starts with the definition of $\pi_{\Delta,X}$, which is in fact an abstraction of the path π pointing up important parts of the path given by the set of labels X. It is followed by the essential definition stating when we can say that a provider regards at most n clients.

Definition 5.1. Let $\mathcal{D}$ be a dynamic system model, $X \subseteq \mathcal{L}_{\mathcal{C}_0}$, $j \in \mathbb{N}$ and $\pi = q_0, l_0, q_1, l_1, q_2, \cdots \in Path^{Init}_{Inf}(\mathcal{D}_j)$. Then $\pi_{\Delta,X} = h(q_0,l_0,q_1), h(q_1,l_1,q_2), \ldots,$ such that h is defined as:

$$h\big((r_0,\ldots,r_j),(o,a,o'),(r'_0,\ldots,r'_j)\big) = \begin{cases} r_0,(f(o),a,f(o')),r'_0 : \\ \quad (o,a,o') \in Comm(X,\mathbb{N}) \\ \epsilon : \quad \text{otherwise,} \end{cases}$$

where ϵ is an empty string and $f(o) = \begin{cases} o, & o \notin \mathbb{N}, \\ *, & o \in \mathbb{N}. \end{cases}$

Example 5.1. For the dynamic system model $\mathcal{DB}$ described in Figure 2, $X = \{(-,read2,\alpha),(\alpha,read2',-)\}$, $j = 4$ and

$$\begin{aligned} \pi = (000,0,0,0) &\overset{(3,init,3)}{\rightarrow} (000,0,0,1) \overset{(3,v2.0,3)}{\rightarrow} \\ &\rightarrow (000,0,0,3) \overset{(3,read2,\alpha)}{\rightarrow} (010,0,0,6) \overset{(1,init,1)}{\rightarrow} \\ &\rightarrow (010,1,0,6) \overset{(1,v1.0,1)}{\rightarrow} (010,2,0,6) \overset{(1,read1,\alpha)}{\rightarrow} \\ &\rightarrow (011,4,0,6) \overset{(\alpha,read2',3)}{\rightarrow} (001,4,0,7) \cdots \end{aligned}$$

it holds: $\pi_{\Delta,X} = \boxed{000} \xrightarrow{(*,read2,\alpha)} \boxed{010}, \boxed{011} \xrightarrow{(\alpha,read2',*)} \boxed{001}, \cdots$

Definition 5.2. Let $\mathcal{D}$ be a dynamic system model, $X \subseteq \mathcal{L}_{\mathcal{C}_0}$ contains all observable labels, and $n \in \mathbb{N}$. We say that *the provider at any time regards at most n clients* iff for any $i \geq n$ it holds that:
$\{\pi_{\Delta,X} \mid \pi \in Path^{Init}_{Inf}(\mathcal{D}_i)\} = \{\pi_{\Delta,X} \mid \pi \in Path^{Init}_{Inf}(\mathcal{D}_n)\}$.

Notation. *Let $\mathcal{D}$ be a dynamic system model and $X \subseteq \mathcal{L}_{\mathcal{C}_0}$ again contains all observable labels. Then $|\mathcal{D}|_X$ is a minimal n such that the provider at any time regards at most n clients, if there is any. If there is no such n then $|\mathcal{D}|_X = \infty$.*

Example 5.2. For the dynamic system model $\mathcal{DB}$ described in Figure 2 and
$X = \{(-,save2,\alpha),(\alpha,impl,-),(-,impl',\alpha),(\alpha,save2',-)\}$
it holds that:

- $\{\pi_{\Delta,X} \mid \pi \in Path^{Init}_{Inf}(\mathcal{DB}_0)\} = \emptyset$,
- $\{\pi_{\Delta,X} \mid \pi \in Path^{Init}_{Inf}(\mathcal{DB}_1)\}$ contains exactly one infinite sequence σ of the shape:

  $\boxed{000} \xrightarrow{(*,save2,\alpha)} \boxed{001}, \boxed{001} \xrightarrow{(\alpha,impl,-)} \boxed{002}, \boxed{002} \xrightarrow{(-,impl',\alpha)} \boxed{003},$
  $\boxed{003} \xrightarrow{(\alpha,save2',*)} \boxed{000}, \boxed{000} \xrightarrow{(*,save2,\alpha)} \boxed{001}, \cdots$

  The set moreover contains all finite prefixes of σ ending with "$\boxed{003} \xrightarrow{(\alpha,save2',*)} \boxed{000}$" and the empty sequence ϵ.

- $\{\pi_{\Delta,X} \mid \pi \in Path^{Init}_{Inf}(\mathcal{D}B_2)\} =$
  $\{\pi_{\Delta,X} \mid \pi \in Path^{Init}_{Inf}(\mathcal{D}B_1)\}$.

Hence the database regards at most one client.

The following lemma shows, that if in a dynamic system model $\mathcal{D}$ we delete the transitions over any subset of observable labels $Comm(X,\mathbb{N})$, such that the result $\mathcal{D}'$ is a dynamic system model, then it fulfils $|\mathcal{D}'|_X \leq |\mathcal{D}|_X$.

Lemma 5.1. *Let $\mathcal{D}$ be a dynamic system model, $X \subseteq \mathcal{L}_{\mathcal{C}_0}$. If a dynamic system model $\mathcal{D}' = (\mathcal{C}_0, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F} \setminus S)$ fulfils $S \subseteq Comm(X,\mathbb{N})$, then $|\mathcal{D}'|_X \leq |\mathcal{D}|_X$.*

Proof. The statement follows immediately from the fact that all deleted transitions, which occur in automata $\{\mathcal{D}_i\}_{i\in\mathbb{N}_0}$ and do not occur in automata $\{\mathcal{D}'_i\}_{i\in\mathbb{N}_0}$, have labels in the set $Comm(X,\mathbb{N})$. □

The next lemma involves dynamic system models, in which providers are composite components such that one of their sub-components contains all actions that are used for communication with clients. This lemma can be used for finding an over-approximation of the value $|\mathcal{D}|_X$ for these systems. In addition to these, it has a corollary useful for verification of properties which we are interested in.

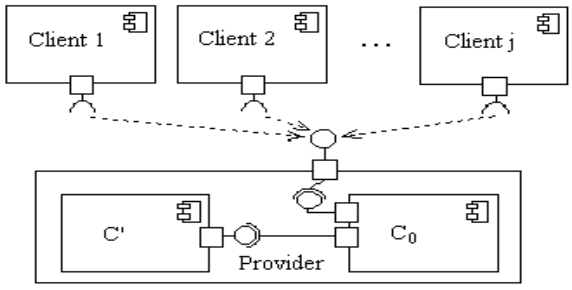


Figure 3: A dynamic system with composed component according to Lemma 5.2 and j clients.

Lemma 5.2. *Let $\mathcal{D}$ be a dynamic system model and $X \subseteq \mathcal{L}_{\mathcal{C}_0}$. Provided that $\mathcal{C}' = (Q', Act', \delta', I', H')$ is a CI automaton and $\mathcal{F}'$ is a set of labels such that $\otimes^{\mathcal{F}'}\{\mathcal{C}_0, \mathcal{C}'\}$ is defined, $S_{H'} \subseteq \mathbb{A}$, then $\mathcal{D}' = (\otimes^{\mathcal{F}'}\{\mathcal{C}_0, \mathcal{C}'\}, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F})$ is a dynamic system model. Further, let the following holds:*

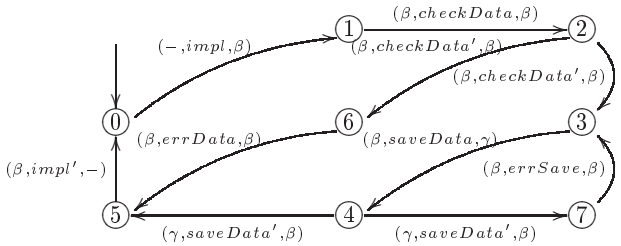
1) $(o',a,o) \in \mathcal{F}', o' \in S_{H'} \Rightarrow o \in \{-\} \cup S_{H_0} \cup S_{H'}$,
   $(o,a,o') \in \mathcal{F}', o' \in S_{H'} \Rightarrow o \in \{-\} \cup S_{H_0} \cup S_{H'}$,
2) $(o,a,o') \in \mathcal{F}', o' \in S_{H'}, o \in S_{H_0} \Rightarrow (o,a,-) \in X \cap \mathcal{F}$,
   $(o',a,o) \in \mathcal{F}', o' \in S_{H'}, o \in S_{H_0} \Rightarrow (-,a,o) \in X \cap \mathcal{F}$,
3) $\forall i \geq |\mathcal{D}|_X : Path^{Init}(\mathcal{D}_i) = Path^{Init}_{Inf}(\mathcal{D}_i)$.

Then for $X' \subseteq Comm(X, S_{H'}) \cup \mathcal{L}_{\mathcal{C}'}$ is $|\mathcal{D}'|_{X'} \leq |\mathcal{D}|_X$.

Proof. See appendix.

Example 5.3. This example illustrates a usage of previous two lemmas. Let us consider the dynamic system model $\mathcal{D}B$ from Figure 2. In Figure 4 there is a CI automaton $\mathcal{C}'$ which implements service *save*, thus the CI automaton $C_0^{impl} = \otimes^{\mathcal{F}'}\{\mathcal{C}_0, \mathcal{C}'\}$ models the database where all the services are implemented, and $\mathcal{D}B^{impl} = (C_0^{impl}, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F}''')$, where $\mathcal{F}'$, $\mathcal{F}'''$ are described in Figure 4, models a dynamic

- $\mathcal{C}'$:



A hierarchy of component names: (β, γ)

- $\mathcal{F}' = \mathcal{L}_{\mathcal{C}_0} \cup \mathcal{L}_{\mathcal{C}'} \cup \{_{(\alpha,impl,\beta),(\beta,impl',\alpha)}\}$
- $\mathcal{F}'' = \mathcal{F} \cup \mathcal{L}_{int,\mathcal{C}_0^{impl}}$
- $\mathcal{F}''' = \mathcal{F}'' \setminus \{_{(\alpha,impl,-),(-,impl',\alpha)}\}$.

Figure 4: A CI automaton $\mathcal{C}'$ and sets $\mathcal{F}'$, $\mathcal{F}''$, $\mathcal{F}'''$.

system whose provider is a database with all services implemented. Its clients are the same as in database $\mathcal{D}B$ and the clients can use the services of the provider similarly as in $\mathcal{D}B$. Lemma 5.2 says that for all $X \supseteq \{_{(\alpha,impl,-),(-,impl',\alpha)}\}$ and $X' \subseteq Comm(X,\mathbb{N})$:

$$|(\mathcal{C}_0^{impl}, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F}'')|_{X'} \leq |DB|_X,$$

where $\mathcal{F}''$ is described in Figure 4, and Lemma 5.1 then claims that for those X it in addition holds that:
$|DB^{impl}|_X = |(\mathcal{C}_0^{impl}, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F}''')|_{X'} \leq |(\mathcal{C}_0^{impl}, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F}'')|_{X'}$.

5.2 Properties $Property(\mathcal{D}, m)$

Above, for each dynamic system $\mathcal{D}$ and a set of observable labels X, we have defined the value $|\mathcal{D}|_X$. In this part, we show the employment of the value in verification of dynamic systems.

Assume a dynamic system $\mathcal{D}$. The properties that we aim to verify, can be specified with a sequence of formulas $\{\varphi_i\}_{i\in\mathbb{N}_0}$ over $\mathcal{L}_{\mathcal{D}}$ such that a property is satisfied iff for each $i \in \mathbb{N}_0$ it holds that $\varphi_i \models \mathcal{D}_i$. Note that not every sequence of formulas $\{\varphi_i\}_{i\in\mathbb{N}_0}$ represents a meaningful property of the system. Thus we concentrate on the formulas satisfying:
- the property makes no distinctions among clients,
- if the property is violated by a path in a system $\mathcal{D}_{i+j}$ where j components do not perform any steps, the property is violated by the same sequence of labels also in the system with i clients only.

Definition 5.3. Let $\mathcal{D}$ be a dynamic system model. A sequence $\{\varphi_i\}_{i\in\mathbb{N}}$ of formulas over the set of labels $\mathcal{L}_{\mathcal{D}}$ is *harmonised with the dynamic system* $\mathcal{D}$ iff for all $i \in \mathbb{N}$:
- formula φ_i and formula φ_i with permutated numerical names of components have the same set of models,
- let
$\pi^i = (q_0^0, q_0^1, \ldots, q_0^i), l_0, (q_1^0, q_1^1, \ldots, q_1^i), l_1, \ldots \in Path^{Init}_{Inf}(\mathcal{D}_i)$
and for $j \in \mathbb{N}$, $\pi^{i+j} =$
$(q_0^0, \ldots, q_0^i, q_0^{i+1}, \ldots, q_0^{i+j}), l_0, (q_1^0, \ldots, q_1^i, q_0^{i+1}, \ldots, q_0^{i+j}), \ldots$
in $Path^{Init}_{Inf}(\mathcal{D}_{i+j})$ then $\pi^{i+j} \not\models \varphi_{i+j}$ implies $\pi^i \not\models \varphi_i$.

As we have discussed before, we are not interested in all formula sequences implied by the definition above. We focus only on the formula sequences that represent properties whose violation involves only a finite number of observed components. Moreover, we consider only the properties that

are invariant under stuttering according to CI-LTL. The first restriction is requisite, but the second one is not. If we did not apply the second restriction, we would in many cases end up with $|\mathcal{D}|_X = \infty$.

Definition 5.4. Two paths π and σ are *stuttering equivalent* wrt. a set of labels $\mathcal{L}$ iff there are two finite or infinite sequences $0 = i_0^1 < i_1^1 < \cdots$ and $0 = i_0^2 < i_1^2 < \cdots$ such that for each index $j \geq 1$ and each $l \in \mathcal{L}$, the following holds:

- $\pi^{i_{j-1}^1} \models \mathcal{E}(l) \Leftrightarrow \pi^{i_{j-1}^1+1} \models \mathcal{E}(l) \Leftrightarrow \cdots \Leftrightarrow \pi^{i_j^1-1} \models \mathcal{E}(l)$
  $\Longleftrightarrow$
  $\sigma^{i_{j-1}^2} \models \mathcal{E}(l) \Leftrightarrow \sigma^{i_{j-1}^2+1} \models \mathcal{E}(l) \Leftrightarrow \cdots \Leftrightarrow \sigma^{i_j^2-1} \models \mathcal{E}(l)$
- $\pi^{i_j^1-1} \models \mathcal{P}(l) \Longleftrightarrow \sigma^{i_j^2-1} \models \mathcal{P}(l)$
- $\forall l \in \mathcal{L}, i^1 \in \mathbb{N}: \ \pi^{i^1} \models \mathcal{P}(l) \Rightarrow i^1 = i_j^1 - 1 \ (j \in \mathbb{N})$
- $\forall l \in \mathcal{L}, i^2 \in \mathbb{N}: \ \sigma^{i^2} \models \mathcal{P}(l) \Rightarrow i^2 = i_j^2 - 1 \ (j \in \mathbb{N})$.

Observe that if π and σ are stuttering equivalent paths, then for each $j \geq 0$, all the paths $\pi^{i_j^1}, \ldots, \pi^{i_{j+1}^1-1}$ and $\sigma^{i_j^2}, \ldots, \sigma^{i_{j+1}^2-1}$ are also pairwise stuttering equivalent.

Definition 5.5. A CI-LTL formula φ is called *invariant under stuttering* iff for each two paths π and σ stuttering equivalent wrt. $\mathcal{L}_\varphi$, the equivalence $\pi \models \varphi \Leftrightarrow \sigma \models \varphi$ holds.

Lemma 5.3. *Let φ be a CI-LTL formula which does not contain operator $\mathcal{X}$ and any occurrence of an atomic proposition $\mathcal{P}(l)$ in φ is of the form either $\phi_1 \mathcal{U}(\mathcal{P}(l) \wedge \phi_2)$ or $(\phi_1 \wedge \neg\mathcal{P}(l))\mathcal{U}\phi_2$ or $(\phi_1 \wedge \neg\mathcal{P}(l_1))\mathcal{U}(\mathcal{P}(l_2) \wedge \phi_2)$. Then φ is invariant under stuttering.*

Proof. See appendix.

Definition 5.6. Let $\mathcal{D}$ be a dynamic system model and $\{\varphi_i\}_{i\in\mathbb{N}}$ be a sequence of formulas harmonised with $\mathcal{D}$. Then we define $|\{\varphi_i\}_{i\in\mathbb{N}}|_\mathcal{D}$ as the minimal $m \in \mathbb{N}_0$ such that for each $j \in \mathbb{N}$ and each path $\pi \in Path_{Inf}^{Init}(\mathcal{D}_j)$ satisfying $\pi \not\models \varphi_j$ there is an invariant under stuttering subformula $\varphi_{j,\pi}$ of formula φ_j and there are names of components $i_1, \ldots, i_m \in \mathbb{N}$ that fulfil

- $\pi \not\models \varphi_{j,\pi}$,
- formula $\neg\varphi_{j,\pi} \Rightarrow \neg\varphi_j$ is satisfied on each automaton, whose set of labels is a superset of $\mathcal{L}_{\varphi_{j,\pi}} \cup \mathcal{L}_{\varphi_j}$,
- $\mathcal{L}_{\varphi_{j,\pi}} \subseteq \mathcal{L}_\mathcal{D}$ and $\mathcal{L}_{\varphi_{j,\pi}}$ does not contain the labels that involve names of dynamic components that are different from $i_1, \ldots, i_m$.

Definition 5.7. For a dynamic system model $\mathcal{D}$ and $m \in \mathbb{N}_0$, $Property(\mathcal{D}, m)$ is the set of all sequences harmonised with the dynamic system $\mathcal{D}$ such that

$\{\varphi_i\}_{i\in\mathbb{N}} \in Property(\mathcal{D}, m)$ iff $|\{\varphi_i\}_{i\in\mathbb{N}}|_\mathcal{D} \leq m$.

Example 5.4. In this example, the previous definition is illustrated by three properties of dynamic system model $\mathcal{DB}$ from Figure 2 described by a sequence of CI-LTL formulas.

1) Consider the sequence of formulas
$\varphi_0 = \varphi_1 = true$
$\varphi_2 = \phi(1,2)$
$\varphi_3 = \phi(1,2) \wedge \phi(1,3) \wedge \phi(2,3)$,
$\ldots$,
where
$\phi(i,j) = \neg\mathcal{F}\left(\mathcal{E}_{(i,save2',\alpha)} \wedge \mathcal{E}_{(j,save2',\alpha)}\right)$
capturing that *a state, in which the provider can respond that the action 'save' was done to two clients, is unreachable.* This sequence is in $Property(\mathcal{DB}, 2)$ because for arbitrary $n \in \mathbb{N}_0$ and $\pi \in Path_{Inf}^{Init}(\mathcal{DB}_n)$ satisfying $\pi \not\models \varphi_n$, there exists $i \in \mathbb{N}_0$ such that for different numbers $j_1, j_2 \in \mathbb{N}$ the actions $_{(j_1,save2',\alpha)}$, $_{(j_2,save2',\alpha)}$ are enabled in the state $\pi(i)$. Thus $\pi \not\models \phi(j_1, j_2) = \varphi_{n,\pi}$ and it is obvious that the formula $\varphi_{n,\pi}$ fulfils all conditions in Definition 5.6.

2) The set $Property(\mathcal{DB}, 0)$ contains all temporal properties described by a sequence of identical formulas $\{\varphi_i = \varphi\}_{i\in\mathbb{N}_0}$, where φ is invariant under stuttering and $\mathcal{L}_\varphi$ does not contain labels that involve any clients.
For example:
$\varphi = \mathcal{G}\left(\mathcal{P}_{(\alpha,impl,-)} \Rightarrow \mathcal{F}\,\mathcal{P}_{(-,impl',\alpha)}\right)$.
This sequence of formulas is in $Property(\mathcal{DB}, 0)$ because for an arbitrary $n \in \mathbb{N}_0$ and $\pi \in Path_{Inf}^{Init}(\mathcal{DB}_n)$ satisfying $\pi \not\models \varphi_n$ the formula $\varphi_{n,\pi} = \varphi_n$ fulfils all conditions in Definition 5.6.

3) The set $Property(\mathcal{DB}, 1)$ contains the sequence of formulas
$\varphi_0 = true$
$\varphi_1 = \phi(1)$,
$\varphi_2 = \phi(1) \wedge \phi(2)$,
$\varphi_3 = \phi(1) \wedge \phi(2) \wedge \phi(3)$,
$\ldots$,
where
$\phi(i) = \mathcal{G}\left(\mathcal{P}_{(i,edit,i)} \Rightarrow \mathcal{F}\,\mathcal{E}_{(i,save,\alpha)}\right)$,
capturing that *globally after some component edits an entry in the database, it will be eventually enabled to save the changes.*

Similarly as in the case 1) it can be shown that this sequence of formulas is in the set $Property(\mathcal{DB}, 1)$.

Finally, the following lemma claims that for verification of the properties from the set $Property(\mathcal{D}, 0)$ it suffices to verify the models $\mathcal{D}_0, \ldots, \mathcal{D}_{|\mathcal{D}|_X}$. The theorem below formulates an analogical result for the properties $Property(\mathcal{D}, m)$, $m \in \mathbb{N}$, stating that we only need to verify the models $\mathcal{D}_0, \ldots, \mathcal{D}_{|\mathcal{D}|_X+m}$. The idea of this theorem is based on the modification of the dynamic system $\mathcal{D}$ and the corresponding modification of verified temporal property. To these modified system and property, we can apply Lemma 5.2 and get the statement from the theorem. [4]

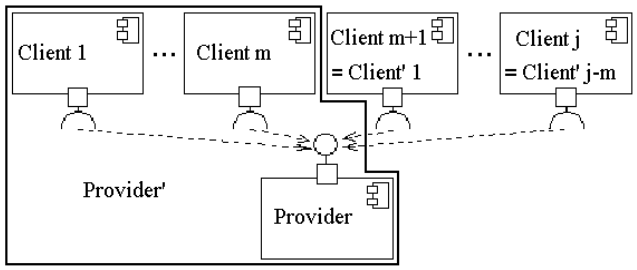


Figure 5: A dynamic system $\mathcal{D}$ with j clients and dynamic system $\mathcal{D}'$ with $j - m$ clients.

[4] The modified model $\mathcal{D}'$ is created from $\mathcal{D}$ by modification of its provider, modelling the provider of $\mathcal{D}$ composed with m clients. The remaining items of the dynamic system are identical to $\mathcal{D}$ (see Figure 5). The modification of property is more complex and it is described in detail in the proof of the theorem.

Definition 5.8. Let $\mathcal{D}$ be a dynamic system model and $\{\varphi_i\}_{i\in\mathbb{N}}$ then a set of labels X *contains all labels necessary for verification of* $\{\varphi_i\}_{i\in\mathbb{N}}$ *on* $\mathcal{D}$ iff

- $\mathcal{P}(l)$ is a subformula of φ_i for some $i \in \mathbb{N}_0 \Rightarrow$ $l \in Comm(X, \mathbb{N})$
- $\mathcal{E}(l)$ is a subformula of φ_i for some $i \in \mathbb{N}_0 \Rightarrow$ $\forall$ transition (q, l', q') in $\mathcal{D}_i$ iff $q \models \mathcal{E}(l)$ and $q' \not\models \mathcal{E}(l)$ or vice versa, then $l \in Comm(X, \mathbb{N})$.

Lemma 5.4. *Let* $\mathcal{D}$ *be a dynamic system model and* $\{\varphi_i\}_{i\in\mathbb{N}} \in Property(\mathcal{D}, 0)$ *and* X *contains all necessary labels for verification of it. Then for every* $j \in \mathbb{N}$ *it holds that:*

$$\mathcal{D}_{|\mathcal{D}|_X} \models \varphi_{|\mathcal{D}|_X} \;\Rightarrow\; \mathcal{D}_{|\mathcal{D}|_X+j} \models \varphi_{|\mathcal{D}|_X+j}.$$

Proof. See appendix.

Theorem 5.1. *Let* $\mathcal{D}$ *be a dynamic system model,* $\{\varphi_i\}_{i\in\mathbb{N}} \in Property(\mathcal{D}, m)$, X *contains all necessary labels for verification of it and*

- $\forall i \geq |\mathcal{D}|_X$: $Path^{Init}(\mathcal{D}_i) = Path^{Init}_{Inf}(\mathcal{D}_i)$,
- $\big((a,act,n) \in \mathcal{F}$ *for some* $n \in \mathbb{N}\big) \;\Rightarrow\; (a,act,-) \in X \cap \mathcal{F}$,
- $\big((n,act,a) \in \mathcal{F}$ *for some* $n \in \mathbb{N}\big) \;\Rightarrow\; (-,act,a) \in X \cap \mathcal{F}$.

Then for every $j \in \mathbb{N}$ *it holds that:*

$$\mathcal{D}_{|\mathcal{D}|_X+m} \models \varphi_{|\mathcal{D}|_X+m} \;\Rightarrow\; \mathcal{D}_{|\mathcal{D}|_X+m+j} \models \varphi_{|\mathcal{D}|_X+m+j}.$$

Proof. See appendix.

6. ALGORITHM

In the previous section, we have shown that if for a dynamic system model $\mathcal{D}$, a property $\{\varphi_i\}_{i\in\mathbb{N}}$ and a set X that contains all labels necessary for verification of $\{\varphi_i\}_{i\in\mathbb{N}}$ on $\mathcal{D}$, we know the value m such that $\{\varphi_i\}_{i\in\mathbb{N}} \in Property(\mathcal{D}, m)$ and an over-approximation n of the value $|\mathcal{D}|_X$, then we can verify the property via verification of the models $\mathcal{D}_0, \ldots, \mathcal{D}_{m+n}$. We did not discuss how we can find appropriate X, m and n, which we are going to do now. The set X can be constructed automatically based on Definition 5.8. The value m such that $\{\varphi_i\}_{i\in\mathbb{N}} \in Property(\mathcal{D}, m)$ can be derived from the structure of the formulas, and hence we can suppose that this value is known already at the time when the formulas for a given property are constructed. The computation of a reasonable over-approximation of the value $|\mathcal{D}|_X$ does not need to be so straightforward. Hence in this section, we discuss the technique that allows us to compute the over-approximation of $|\mathcal{D}|_X$ for most of the dynamic system models $\mathcal{D}$ such that $|\mathcal{D}|_X \neq \infty$.

For this purpose, we employ the value $\|\mathcal{D}\|_X$, which reflects the number of clients that the provider can serve concurrently at any moment. The value $\|\mathcal{D}\|_X$ can be computed automatically and it always holds that $\|\mathcal{D}\|_X \leq |\mathcal{D}|_X$. For the most common type of dynamic systems where $|\mathcal{D}|_X < \infty$, there exists a computable value z such that $|\mathcal{D}|_X \leq 1 + \|\mathcal{D}\|_X \cdot z$.

6.1 Algorithm for computing $\|\mathcal{D}\|_X$

We first define the value $\|\mathcal{D}\|_X$ and then prove that $\|\mathcal{D}\|_X \leq |\mathcal{D}|_X$ always holds.

Definition 6.1. Let $\mathcal{D}$ be a dynamic system model, $X \subseteq \mathcal{C}_0$. Then a state $q \in Q$ is *not in a cycle of service* X iff it is in a set $N_{\mathcal{D},X} \subseteq Q$ defined inductively:

- $I \subseteq N_{\mathcal{D},X}$,
- $(q \in N_{\mathcal{D},X} \wedge \exists l \notin Comm(X,\mathbb{N}) \colon (q,l,q') \in \delta_1) \;\Rightarrow\; q' \in N_{\mathcal{D},X}$,
- $(q \in N_{\mathcal{D},X} \wedge \exists l \notin Comm(X,\mathbb{N}) \colon (q',l,q) \in \delta_1) \;\Rightarrow\; q' \in N_{\mathcal{D},X}$.

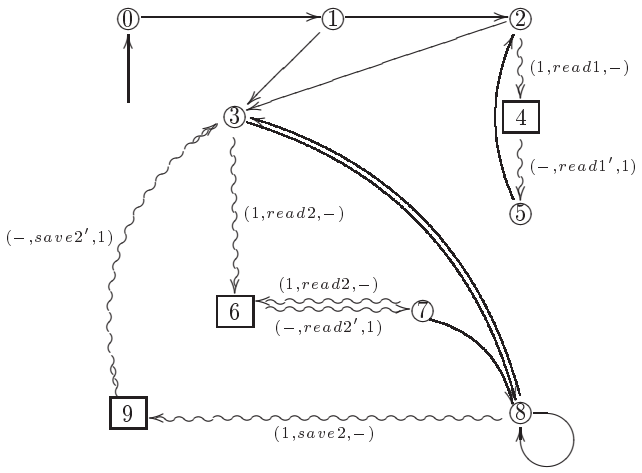


Figure 6: Sets $N_{\mathcal{D}B,\mathcal{L}_{\mathcal{C}_0}}$, $\mathcal{L}^{N\rightarrow}_{\mathcal{D}B,\mathcal{L}_{\mathcal{C}_0}}$ and $\mathcal{L}^{N\leftarrow}_{\mathcal{D}B,\mathcal{L}_{\mathcal{C}_0}}$.

Notation. *Let* $\mathcal{D}$ *be a dynamic system model and* $X \subseteq \mathcal{L}_{\mathcal{C}_0}$. *Then* $\|\mathcal{D}\|_X$ *denotes the minimal* $i \in \mathbb{N}_0$ *such that for each* $j \in \mathbb{N}_0$ *and each reachable state* q *of the automaton* $\mathcal{D}_j$, *the number of clients that are in the state* q *in a cycle of service* X *is less or equal to* i, *if there is any. If there is no such* j *then* $\|\mathcal{D}\|_X = \infty$.

Example 6.1. In the model $\mathcal{D}B$ from Figure 2 at any time, the clients can access either reading or writing on the database. If the database serves the reading, there can be at most two clients that are using it, and if the writing is served, there can be at most one client using the service. Hence it holds that:

- $\|\mathcal{D}B\|_{\{(1,read1,-),(-,read1',1),(1,read2,-),(-,read2',1)\}} = 2$,
- $\|\mathcal{D}B\|_{\{(1,save2,-),(-,save2',1)\}} = 1$ and
- $\|\mathcal{D}B\|_{\mathcal{L}_{\mathcal{C}_0}} = 2$.

Notation. *Let* $\mathcal{D}$ *be a dynamic system model,* $X \subseteq \mathcal{L}_{\mathcal{C}_0}$. *As the sets of states of automata* $\{\mathcal{C}_i\}_{i\in\mathbb{N}}$ *are identical, we denote:*

- $N^{\rightarrow}_{\mathcal{D},X} = \{q \in N_{\mathcal{D},X} \mid \exists q' \notin N_{\mathcal{D},X} : \; (q,l,q') \in \delta_1\}$ *– the states of the automaton* $\mathcal{C}_1$ *that are not in a cycle of service, and from which there is a transition to a state in a cycle of service* X.
- $N^{\leftarrow}_{\mathcal{D},X} = \{q \in N_{\mathcal{D},X} \mid \exists q' \notin N_{\mathcal{D},X} : \; (q',l,q) \in \delta_1\}$ *– the states of the automaton* $\mathcal{C}_1$ *that are not in a cycle of service, and to which there is a transition from a state in a cycle of service* X.
- $\mathcal{L}^{N\rightarrow}_{\mathcal{D},X}$ *– labels, over which a transition from a state in* $N_{\mathcal{D},X}$ *to a state in* $Q \setminus N_{\mathcal{D},X}$ *exists in the automaton* $\mathcal{C}_1$.
- $\mathcal{L}^{N\leftarrow}_{\mathcal{D},X}$ *– labels, over which a transition from a state in* $Q \setminus N_{\mathcal{D},X}$ *to a state in* $N_{\mathcal{D},X}$ *exists in the automaton* $\mathcal{C}_1$.

Example 6.2. For the dynamic system model $\mathcal{D}B$ from Figure 2, it holds that

$\|\mathcal{D}B\|_{\mathcal{L}_{\mathcal{C}_0}} = 2$,

$N_{\mathcal{D}B,\mathcal{L}_{\mathcal{C}_0}} = \{0,1,2,3,5,7,8\}$,
$N^{\rightarrow}_{\mathcal{D}B,\mathcal{L}_{\mathcal{C}_0}} = \{2,3,7,8\}$,
$N^{\leftarrow}_{\mathcal{D}B,\mathcal{L}_{\mathcal{C}_0}} = \{3,5,7\}$,

$\mathcal{L}^{N\rightarrow}_{\mathcal{D}B,\mathcal{L}_{\mathcal{C}_0}} = \{(1,read1,-),(1,read2,-),(1,save2,-)\}$,
$\mathcal{L}^{N\leftarrow}_{\mathcal{D}B,\mathcal{L}_{\mathcal{C}_0}} = \{(-,read1',1),(-,read2',1),(-,save2',1)\}$.

Figure 6 illustrates these terms graphically – the states from

$N_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}$ are depicted as ⓘ, the transitions with labels from $\mathcal{L}^{N\rightarrow}_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}$ and $\mathcal{L}^{N\leftarrow}_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}$ are depicted as $\rightsquigarrow$.

As the value $\|D\|_X$ is algorithmically computable, the following lemma helps in finding the under-approximation of $|\mathcal{D}|_X$ and it implies that if $\|\mathcal{D}\|_X = \infty$, then also $|\mathcal{D}|_X = \infty$. In the next sub-section, we more demonstrate possible usage of the value $\|\mathcal{D}\|_X$ for computing the over-approximation of $|\mathcal{D}|_X$.

Lemma 6.1. *Let $\mathcal{D}$ be a dynamic system model and $X \subseteq \mathcal{L}_{\mathcal{C}_0}$. If $\mathcal{L}^{N\rightarrow}_{\mathcal{D},X} \cap \mathcal{L}^{N\leftarrow}_{\mathcal{D},X} = \emptyset$ then $\|\mathcal{D}\|_X \leq |\mathcal{D}|_X$.*

Proof. See appendix.

Example 6.3. The previous lemma together with Example 6.1 for the model $\mathcal{DB}$ (Figure 2) claims that
- $|\mathcal{DB}|_{\{(1,read1,-),(-,read1',1),(1,read2,-),(-,read2',1)\}} \geq 2$,
- $|\mathcal{DB}|_{\{(1,save2,-),(-,save2',1)\}} \geq 1$ and
- $|\mathcal{DB}|_{\mathcal{L}_{\mathcal{C}_0}} \geq 2$.

6.2 Over-approximation of $|\mathcal{D}|_{\mathcal{L}_{\mathcal{C}_0}}$

Here we focus on finding an over-approximation of $|\mathcal{D}|_X$ using the value $\|\mathcal{D}\|_X$. This task is very complicated and for space reasons, we are not going to present here a general algorithm for computing this value. On the other hand, the computation of the over-approximation is relatively straightforward in many specific cases, and if X is very small (two to four items), it usually suffices to employ the observation that:

"If for a dynamic system $\mathcal{D}$ the automaton $\mathcal{D}_n$ generates all possible runs with respect to the observable labels X, then for every $i \in \mathbb{N}$ the automaton $\mathcal{D}_{n+i}$ generates again the same runs. Hence it holds that $|\mathcal{D}|_X \leq n$."

This implies the next observation that the most difficult case of getting the over-approximation of $|\mathcal{D}|_X$ is for $X = \mathcal{L}_{\mathcal{C}_0}$ (when all labels are observable). Hence we concentrate on this case only. And again for space reasons, not on the general algorithm for computing an over-approximation of $|\mathcal{D}|_{\mathcal{L}_{\mathcal{C}_0}}$, but on an algorithm that can be employed for most practically used dynamic systems. More, this algorithm presents the core idea that can drive construction of the general algorithm.

From now on, we suppose that
- every dynamic system $\mathcal{D}$ fulfils $\mathcal{L}^{N\rightarrow}_{\mathcal{D},X} \cap \mathcal{L}^{N\leftarrow}_{\mathcal{D},X} = \emptyset$,
- in the model, any transition that can synchronise with the provider has at least one of its states in a cycle of service.

These conditions are very weak and common models (which model requests and responses separately) most likely satisfy it.

Notation. *Let $\mathcal{D}$ be a dynamic system model, X a set of labels and $q \in N_{\mathcal{D},X}$. Then* a set of paths of service starting in *q is the set $ServicePath_{\mathcal{D},X}(q)$ of paths of the automaton $\mathcal{C}_1$ that start in the state q, finish in a state $q' \in N^{\leftarrow}_{\mathcal{D},X}$, and none of their internal states is in $N_{\mathcal{D},X}$.*
For $\pi \in ServicePath_{\mathcal{D},X}(q)$ we denote $\pi_{observable}$ the sequence that results from π after removing the states and labels that can not be used for synchronisation with the provider and $ServiceOPath_{\mathcal{D},X}(q)$ denotes the set $\{\pi_{observable} \mid \pi \in ServicePath_{\mathcal{D},X}(q)\}$.

Example 6.4. For the dynamic system model $\mathcal{DB}$ from Figure 2 the following holds:
$N^{\rightarrow}_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}} = \{2,3,7,8\}$

$ServicePath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(2) = \{2,{}_{(1,read1,-)},4,{}_{(-,read1',1)},5\}$,
$ServicePath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(3) = \{3,{}_{(1,read2,-)},6,{}_{(-,read2',1)},7\}$,
$ServicePath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(7) = \{7,{}_{(1,read2,-)},6,{}_{(-,read2',1)},7\}$,
$ServicePath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(8) = \{8,{}_{(1,save2,-)},9,{}_{(-,save2',1)},3\}$.

$ServiceOPath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(2) = \{{}_{(1,read1,-),(-,read1',1)}\}$,
$ServiceOPath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(3) = ServiceOPath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(7) = \{{}_{(1,read2,-),(-,read2',1)}\}$,
$ServiceOPath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(8) = \{{}_{(1,save2,-),(-,save2',1)}\}$.

Notation. *For a dynamic system model $\mathcal{D}$ and the set X, the automaton $\mathcal{C}_i$ restricted to the states that are in $N_{\mathcal{D},X}$ is denoted $\overline{\mathcal{C}_i}$.*

Example 6.5. For the dynamic system model $\mathcal{DB}$ from Figure 2 and $X = \mathcal{L}_{\mathcal{C}_0}$, the automaton $\overline{\mathcal{C}_1}$ contains all states depicted in Figure 6 but it contains only the transitions which are depicted as $\longrightarrow$.

Notation. *The subset of $N^{\rightarrow}_{\mathcal{D},X}$ reachable from the initial state of $\overline{\mathcal{C}_1}$ is denoted $Ninit^{\rightarrow}_{\mathcal{D},X}$.*

Example 6.6. For $\mathcal{DB}$ as the dynamic system model from the Figure 2, $Ninit^{\rightarrow}_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}} = \{2,3\}$. Figure 6 shows, that those states are the only states in $N^{\rightarrow}_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}$ reachable from the initial state only via transitions denoted $\longrightarrow$.

Definition 6.2. Let $\mathcal{D}$ be a dynamic system model and X be a set and $q \notin N^{\rightarrow}_{\mathcal{D},X}$. The set of maximal paths in a cycle of service starting in q, denoted $SCPath_{\mathcal{D},X}(q)$, is the set of all maximal paths from q which do not contain a state from $N^{\rightarrow}_{\mathcal{D},X}$ and which do not contain any state twice.
Let $\pi \in SCPath_{\mathcal{D},X}(q)$ for some $q \notin N^{\rightarrow}_{\mathcal{D},X}$, then π_{trace} corresponds to the path π, from which there have been removed all the labels and states that either belong to the set $N_{\mathcal{D},X}$ or do not belong to $N_{\mathcal{D},X}$ and are succeeded by a state not belonging to $N_{\mathcal{D},X}$. Thus π_{trace} is a sequence of states not belonging to $N_{\mathcal{D},X}$ whose successor in π belongs to $N_{\mathcal{D},X}$.
For $q \in Q$, let $MaxSCTrace_{\mathcal{D},X}(q)$ denotes a subset of the set $\{\pi_{trace} \mid \pi \in SCPath_{\mathcal{D},X}(q)\}$, which contains only the maximal sequences.
Let $MaxCSTrace_{\mathcal{D},X} = \bigcup_{q \in Ninit^{\rightarrow}_{\mathcal{D},X}} MaxSCTrace_{\mathcal{D},X}(q)$.

Example 6.7. For the dynamic system model $\mathcal{DB}$ depicted in Figure 2 it holds:
$\{\pi_{trace} \mid \pi \in SCPath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(2)\} = \{4\}$,
$\{\pi_{trace} \mid \pi \in SCPath_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(3)\} = \{6;\ 6,9\}$,
$MaxSCTrace_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(2) = \{4\}$,
$MaxSCTrace_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}}(3) = \{6,9\}$,
$MaxSCTrace_{\mathcal{DB},\mathcal{L}_{\mathcal{C}_0}} = \{4;\ 6,9\}$.

The following lemma contains two preconditions. The first one requires that anytime after serving a client, the client must be able to launch the same serving again. This precondition is requisite. However if it is not fulfilled, $|\mathcal{D}| = \infty$ very likely occurs. The second precondition simplifies the notation and it could be weakened if necessary.

Lemma 6.2. *Let $\mathcal{D}$ be a dynamic system model such that:*
- $\forall q \in N^{\rightarrow}_{\mathcal{D},\mathcal{L}_{\mathcal{C}_0}},\ \pi = {}_{q_0,l_0,\ldots,l_n,q_{n+1}} \in ServicePath_{\mathcal{D},\mathcal{L}_{\mathcal{C}_0}}(q)$

$\exists q'_0 \in N^{\rightarrow}_{\mathcal{D},\mathcal{L}_{\mathcal{C}_0}}$ *reachable in* $\overline{\mathcal{C}_1}$ *from the state* q_{n+1} *and* $\pi' \in ServicePath_{\mathcal{D}}(q'_0) : \pi_{observable} = \pi'_{observable}$

• $\forall q \in N_{\mathcal{D},\mathcal{L}_{\mathcal{C}_0}}$, $\pi_1, \pi_2 \in ServicePath_{\mathcal{D}}(q)$*: last state of* π_1 *is equal to last state of* π_2*.*

Then

$$|\mathcal{D}|_{\mathcal{L}_{\mathcal{C}_0}} \leq 1 + \|\mathcal{D}\|_{\mathcal{L}_{\mathcal{C}_0}} \cdot \sum_{\pi \in MaxSCTrace_{\mathcal{D},\mathcal{L}_{\mathcal{C}_0}}} |\pi|,$$

where $|\pi|$ *denotes the number of states in* π*.*

Proof. See appendix.

Example 6.8. For the dynamic system model $|\mathcal{D}\mathcal{B}|_{\mathcal{L}_{\mathcal{C}_0}}$ from Figure 2 it holds $\|\mathcal{D}\mathcal{B}\|_{\mathcal{L}_{\mathcal{C}_0}} = 2$, $MaxTrace_{\mathcal{D}\mathcal{B}} = \{4;\ 6, 9\}$ (see Example 6.3, 6.7) and this system fulfils preconditions of Lemma 6.2. Thus Lemma 6.1 and Lemma 6.2 claim

$$2 \leq |\mathcal{D}\mathcal{B}|_{\mathcal{L}_{\mathcal{C}_0}} \leq 1 + 2 \cdot (1 + 2) = 7.$$

Therefore for the dynamic system model $|\mathcal{D}\mathcal{B}^{impl}|_{\mathcal{L}_{\mathcal{C}_0}}$ from Example 5.3 it holds

$$|\mathcal{D}\mathcal{B}^{impl}|_{\mathcal{L}_{\mathcal{C}_0}} \leq 7.$$

7. CONCLUSIONS AND FUTURE WORK

The paper introduces our approach to the verification of dynamic systems. It is based on getting a value n, which guarantees that if the examined property is not violated on the system with less then n dynamic components, it will always hold on the system, no matter how many dynamic components are going to be used during its execution. This result is possible thanks to the assumption, that dynamic components share a common type, which makes their behaviour predictable. Besides the proofs of the propositions constituting this result, we also present the intuition for designing algorithms for getting the approximation of n.

Our technique is applicable not only to the verification of dynamic systems. It can be very helpful also during modelling of a dynamic system, because it can be used to check that the model does not have any unpredictable behaviour comparing to the real system (see [12] for discussion of this). In this case, also verification of simple properties, or information about the number of clients that can be served by a provider, can be interesting.

In future, we aim to finish implementation of the algorithms and extend them also to more general types of dynamic models. We also aim at broadening the set of properties and evaluating the approach thoroughly on realistic case studies.

8. REFERENCES

[1] J. Adámek. Addressing Unbounded Parallelism in Verification of Software Components. In *SNPD*, pages 49–56, 2006.

[2] T. Ball and S. K. Rajamani. Automatically Validating Temporal Safety Properties of Interfaces. *Lecture Notes in Computer Science*, 2057:103+, 2001.

[3] J. Barnat, L. Brim, I. Černá, P. Moravec, P. Ročkai, and P. Šimeček. Divine - A Tool for Distributed Verification. In *Proc. of the 18th International Conference CAV'06*, volume 4144 of *LNCS*, pages 278–281. Springer, 2006.

[4] T. Barros, L. Henrio, and E. Madelaine. Behavioural models for hierarchical components. In *Proceedings of the SPIN 2005 Workshop*, pages 154–180, San Francisco, USA, August 2005. LNCS Springer-Verlag.

[5] L. Brim, I. Černá, P. Vařeková, and B. Zimmerova. Component-Interaction Automata as a Verification-Oriented Component-Based System Specification. In *Proceedings of SAVCBS'05*, pages 31–38, Lisbon, Portugal, September 2005.

[6] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, USA, January 2000.

[7] J. Corbett, M. Dwyer, and J. Hatcliff. Expressing checkable properties of dynamic systems: The Bandera Specification language, 2000.

[8] F. Plasil and S. Visnovsky. Behavior protocols for software components. *IEEE Transactions on Software Engineering*, 28(11):1056–1076, November 2002.

[9] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.

[10] A. Rensink, Á. Schmidt, and D. Varró. Model Checking Graph Transformations: A Comparison of Two Approaches. In H. Ehrig, G. Engels, F. Parisi-Presicce, and G. Rozenberg, editors, *International Conference on Graph Transformations (ICGT)*, volume 3256 of *Lecture Notes in Computer Science*, pages 226–241. Springer-Verlag, 2004.

[11] I. Černá, P. Vařeková, and B. Zimmerova. Component-Interaction Automata Modelling Language. Technical Report FIMU-RS-2006-08, Masaryk University, Faculty of Informatics, Brno, Czech Republic, October 2006.

[12] B. Zimmerova, P. Vařeková, N. Beneš, I. Černá, L. Brim, and J. Sochor. *The Common Component Modeling Example: Comparing Software Component Models*, chapter Component-Interaction Automata Approach (CoIn). To appear in LNCS, 2007.

APPENDIX

A. DEFINITIONS

In the following, you may find the formal definition of the *complete transition space* $\Delta_{\mathcal{S}}$ defined informally in Definition 3.3.

Notation. *Let $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ be a nonempty set of integers with $i_1 < \cdots < i_n$, and let Q_i be a set for each $i \in \mathcal{I}$. Then $\Pi_{i\in\mathcal{I}} Q_i$ denotes the set*

$$\{(q_{i_1}, q_{i_2}, \ldots, q_{i_n}) \mid \forall j \in \{1, \ldots, n\} : q_{i_j} \in Q_{i_j}\}.$$

For any $j \in \mathcal{I}$, pr_j denotes a function $pr_j : \Pi_{i\in\mathcal{I}} Q_i \to Q_j$ such that $pr_j((q_i)_{i\in\mathcal{I}}) = q_j$.

Definition A.1. Let $\mathcal{S} = \{(Q_i, Act_i, \delta_i, I_i, H_i)\}_{i\in\mathcal{I}, \mathcal{I}\subseteq\mathbb{N}}$ be a finite composable set of component-interaction automata. Then the *complete transition space* for $\mathcal{S}$ is $\Delta_{\mathcal{S}} = \Delta_{\mathcal{S},old} \cup \Delta_{\mathcal{S},new}$, where

$$\Delta_{\mathcal{S},old} = \big\{(q,(o,a,o'),q') \mid q,q' \in \Pi_{i\in\mathcal{I}} Q_i,\ o,o' \in \mathbb{N}\cup\mathbb{A}\cup\{-\},\ \exists j \in \mathcal{I}:\ ((pr_j(q),(o,a,o'),pr_j(q')) \in \delta_j \wedge \forall i \in (\mathcal{I}\setminus\{j\})\ pr_i(q) = pr_i(q'))\big\}$$

$$\Delta_{\mathcal{S},new} = \big\{(q,(o,a,o'),q') \mid q,q' \in \Pi_{i\in\mathcal{I}} Q_i,\ o,o' \in \mathbb{N}\cup\mathbb{A},\ \exists j,j' \in \mathcal{I}, j \neq j':\ ((pr_j(q),(o,a,-),pr_j(q')) \in \delta_j \wedge (pr_{j'}(q),(-,a,o'),pr_{j'}(q')) \in \delta_{j'} \wedge \forall i \in (\mathcal{I}\setminus\{j,j'\})\ pr_i(q) = pr_i(q'))\big\}$$

B. PROOFS

This appendix provides the reader with the proofs for the lemmas presented in the text.

Proof. Lemma 5.2 Because $\otimes^{\mathcal{F}'}\{\mathcal{C}_0, \mathcal{C}'\}$ is defined and $S_{H'} \subseteq \mathbb{A}$, it can be simply proved that $\mathcal{D}'$ is a dynamic system model by a checking of conditions in the definition of dynamic system model.

To proof that 1), 2) and 3) implies $|\mathcal{D}'|_X \leq |\mathcal{D}|_X$ it suffice to show that

$$\{\sigma_{\Delta,X} \mid \sigma \in Path^{Init}_{Inf}(\mathcal{D}'_k)\} = \{\sigma_{\Delta,X} \mid \sigma \in Path^{Init}_{Inf}(\mathcal{D}'_{|\mathcal{D}|_X})\}.$$

We prove that for each $k \geq |\mathcal{D}|_X$ it is fulfilled

$$\{\sigma_{\Delta,X} \mid \sigma \in Path^{Init}_{Inf}(\mathcal{D}'_k)\} \subseteq \{\sigma_{\Delta,X} \mid \sigma \in Path^{Init}_{Inf}(\mathcal{D}'_{|\mathcal{D}|_X})\},$$

the inverse follows immediately from precondition 3). For $k > |\mathcal{D}|_X$ let $\pi'^k = {}_{((q_0^0,q_0'),q_0^1,\ldots,q_0^k),\, l_0,\, ((q_1^0,q_1'),q_1^1,\ldots,q_1^k),\, l_1,\, \ldots}$ be a sequence from $\{\sigma_{\Delta,X} \mid \sigma \in Path^{Init}_{Inf}(\mathcal{D}'_k)\}$. Then $\overline{\pi}'^k$ denotes a sequence created from π'^k by

a) skipping actions involving only the automaton $\mathcal{C}'$,
b) replacing internal actions over labels in $Comm(X, S_{H'})$ by external actions of the automaton $\mathcal{C}$ which model the part of communication involving the automaton $\mathcal{C}$,
c) skipping of parts of the states which model states of the automaton $\mathcal{C}'$.

Formally $\overline{\pi}'^k = {}_{f((q_0^0,q_0^1,\ldots,q_0^k),l_0),f((q_1^0,q_1^1,\ldots,q_1^k),l_1),\ldots}$ where the function f is defined:

$$f(q,(o,a,o')) = \begin{cases} \epsilon, & o \in S_{H'},\ o' \in S_{H'}, \\ q,(o,a,-), & (o,a,o') \in Comm(X,S_{H'}),\ o' \in S_{H'}, \\ q,(-,a,o'), & (o,a,o') \in Comm(X,S_{H'}),\ o \in S_{H'}, \\ q,(o,a,o'), & otherwise. \end{cases}$$

From the precondition (3) it follows that there has to exist a path $\pi^k \in \{\sigma_{\Delta,X} \mid \sigma \in Path^{Init}_{Inf}(\mathcal{D}_k)\}$ satisfying either $\overline{\pi}'^k = \pi^k$ or $\overline{\pi}'^k$ is a prefix of π^k. Because $k \geq |\mathcal{D}|_X$, Definition 5.2 claims that a path $\pi^{|\mathcal{D}|_X} \in Path^{Init}_{Inf}(\mathcal{D}_{|\mathcal{D}|_X})$ such that $\pi^k_{\Delta,X} = \pi^{|\mathcal{D}|_X}_{\Delta,X}$ must exist. Preconditions 1) and 2) imply that it is possible to construct from the path $\pi^{|\mathcal{D}|}$ (inversely to the steps a), b) and c)) a sequence $\pi'^{|\mathcal{D}|_X} \in \{\sigma_{\Delta,X} \mid \sigma \in Path^{Init}_{Inf}(\mathcal{D}_{|\mathcal{D}|_X})\}$ such that $\pi'^k_{\Delta,X} = \pi'^{|\mathcal{D}|_X}_{\Delta,X}$. □

Proof. Lemma 5.3 By induction to structure of the formula.

- $\varphi = \mathcal{E}(l)$: Follows directly from Definition 5.4.
- $\varphi = \varphi_1 \wedge \varphi_2$: According to IH, φ_1 and φ_2 are invariant under stuttering. Let ρ and σ be stuttering equivalent paths wrt. $\mathcal{L}_\varphi$. Then $\rho \models \varphi \Leftrightarrow \rho \models \varphi_1 \wedge \rho \models \varphi_2 \Leftrightarrow \sigma \models \varphi_1 \wedge \sigma \models \varphi_2 \Leftrightarrow \sigma \models \varphi$.
- $\varphi = \neg\varphi'$: According to IH, φ is invariant under stuttering. Let ρ and σ be stuttering equivalent paths wrt. $\mathcal{L}_\varphi$. Then $\rho \models \varphi \Leftrightarrow \rho \not\models \varphi' \Leftrightarrow \sigma \not\models \varphi' \Leftrightarrow \sigma \models \varphi$.
- $\varphi = (\varphi_1 \wedge \neg\mathcal{P}(l_1))\ \mathcal{U}\ (\mathcal{P}(l_2) \wedge \varphi_2)$: This covers all three cases enumerated in Lemma 5.3. According to IH, φ_1 and φ_2 are invariant under stuttering. Let ρ and σ be stuttering equivalent paths wrt. $\mathcal{L}_\varphi$. Then $\rho \models \varphi \Leftrightarrow (1) : \exists m \in \mathbb{N}_0 : \rho^m \models \mathcal{P}(l_2) \wedge \rho^m \models \varphi_2$ and $(2) : (\forall n < m : \rho^n \models \varphi_1 \wedge \rho^n \models \neg\mathcal{P}(l_1))$. According to stuttering equivalence of ρ and σ and stuttering invariance of φ_1 and φ_2, the condition (1) is equivalent to the condition $(1') : \exists m' \in \mathbb{N}_0 : \sigma^{m'} \models \mathcal{P}(l_2) \wedge \sigma^{m'} \models \varphi_2$. Moreover, the condition (2) is equivalent to the condition $(2') : \forall n' < m' : \sigma^{n'} \models \varphi_1 \wedge \sigma^{n'} \models \neg\mathcal{P}(l_1))$ (this follows from the third requirement of Definition 5.4). Putting those two equivalencies together, we conclude that $\rho \models \varphi \Leftrightarrow (1) \wedge (2) \Leftrightarrow (1') \wedge (2') \Leftrightarrow \sigma \models \varphi$.
- $\varphi = \varphi_1\, \mathcal{U}\, \varphi_2$: According to IH, φ_1 and φ_2 are invariant under stuttering. Let ρ and σ be stuttering equivalent paths wrt. $\mathcal{L}_\varphi$. Then $\rho \models \varphi \Leftrightarrow \exists m \in \mathbb{N}_0 : \rho^m \models \varphi_2 \wedge \forall n < m : \rho^n \models \varphi_1$. According to Definition 5.4, let k be the maximal index such that $i_k \leq m$. Then it also holds: $\rho \models \varphi \Leftrightarrow \exists i_k \in \mathbb{N}_0 : \rho^{i_k} \models \varphi_2 \wedge \forall n < i_k : \rho^n \models \varphi_1$. Since ρ^{i_k} and σ^{j_k} are stuttering equivalent, $\rho^{i_k} \models \varphi_2 \Leftrightarrow \sigma^{j_k} \models \varphi_2$. By a similar argument we can prove that $\forall n < i_k : \rho^n \models \varphi_1 \Leftrightarrow \forall n' < j_k : \sigma^{n'} \models \varphi_1$. Putting the results together we obtain that $\rho \models \varphi \Leftrightarrow \exists j_k \in \mathbb{N}_0 : \sigma^{j_k} \models \varphi_2 \wedge \forall n' < j_k : \sigma^{n'} \models \varphi_1 \Leftrightarrow \sigma \models \varphi$. □

Proof. Lemma 5.4 For space reasons let shortcut d means $|\mathcal{D}|_X$. Assume that $\mathcal{D}_{d+j} \not\models \varphi_{d+j}$. Let for any $j \in \mathbb{N}$, $\sigma^{d+j} \in Path^{Init}_{Inf}(\mathcal{D}_{d+j})$ be a path such that $\sigma^{d+j} \not\models \varphi_{d+j}$. Then according to Definition 5.2, there exists a sequence from $Path^{Init}_{Inf}(\mathcal{D}_d)$

$$\sigma^d = {}_{(q_0^0,q_0^1,\ldots,q_0^d),l_0,(q_1^0,q_1^1,\ldots,q_1^d),l_1,\ldots}$$

such that $\sigma^{d+j}_{\Delta,X} = \sigma^d_{\Delta,X}$. A path $\sigma'^{d+j} =$

$${}_{(q_0^0,\ldots,q_0^d,q_0^{d+1},\ldots,q_0^{d+j}),l_0,(q_1^0,\ldots,q_1^d,q_0^{d+1},\ldots,q_0^{d+j}),l_1,\ldots,}$$

(where $q_0^{d+1}, \ldots, q_0^{d+j}$ are initial states of automata $\mathcal{C}_{d+1}$, $\ldots$, $\mathcal{C}_{d+j}$) is obviously in the set $Path^{Init}_{Inf}(\mathcal{D}_{d+j})$. From the fact that $\sigma'^{d+j}{}_{\Delta,X} = \sigma^d_{\Delta,X}$ it follows that there are two finite or infinite sequences $0 = i_0^1 < i_1^1 < \cdots$ and $0 = i_0^2 < i_1^2 < \cdots$ such that for each index $n \geq 0$:

- for each $l \in \bigcup_{i\in\mathbb{N}_0} \mathcal{L}_{\varphi_i}$ all states of the automaton $\mathcal{C}_0$ $pr_1(\sigma^{d+j}(i_n^1)), \ldots, pr_1(\sigma^{d+j}(i_{n+1}^1 - 1))$ and $pr_1(\sigma'^{d+j}(i_n^2)), \ldots, pr_1(\sigma'^{d+j}(i_{n+1}^2 - 1))$ satisfy $\models E(l)$ or none of them satisfy it,
- labels $\mathcal{L}(\sigma^d, i_n^1), \ldots, \mathcal{L}(\sigma^d, i_{n+1}^1 - 2)$ and $\mathcal{L}(\sigma'^d, i_n^2), \ldots, \mathcal{L}(\sigma'^d, i_{n+1}^2 - 2) \notin \bigcup_{i\in\mathbb{N}_0} \mathcal{L}_{\varphi_i}$
- labels ${}_{(o_1,a,o_1')=\mathcal{L}(\sigma^d,i_{n+1}^1-1)}$ and ${}_{(o_2,a,o_2')=\mathcal{L}(\sigma'^d,i_{n+1}^2-1)}$ model a communication which involves the provider and ${}_{(f(o_1),a,f(o_1'))=(f(o_2),a,f(o_2'))}$, where

$$f(o) = \begin{cases} o, & o \notin \mathbb{N}, \\ *, & o \in \mathbb{N}. \end{cases}$$

Thus according to Definitions 5.6 and 5.7, $\sigma'^{d+j} \not\models \varphi_{d+j}$. So the second condition of the Definition 5.3 implies $\sigma^d \not\models \varphi_d$ and the implication is proved. □

Proof. Theorem 5.1 For space reasons let shortcut d means $|\mathcal{D}|_X$. Suppose that

$$\mathcal{D}_{d+m+j} \not\models \varphi_{d+m+j},$$

then it is sufficient to prove that there exists a path $\sigma \in Path_{Inf}^{Init}(\mathcal{D}_{d+m+j})$, which does not satisfy φ_{d+m+j} and whose transitions do not involve clients with names larger than $d+m$. The fact $\mathcal{D}_{d+m} \not\models \varphi_{d+m}$ then follows immediately from the second condition of the Definition 5.3.

Firstly we define new dynamic systems $\mathcal{D}'$, $\mathcal{D}''$. In this proof $\mathcal{C}'_1, \mathcal{C}'_2, \ldots, \mathcal{C}'_m$ are CI automata $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$ with renamed component names such that the triple

$$(\mathcal{C}_0 \otimes (\otimes\{\mathcal{C}'_i\}_{i\in\{1,\ldots,m\}}), \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \mathcal{F})$$

is a dynamic system model. The injection from the set $\{1,\ldots,m\}$ to the set $\mathbb{A}$ which corresponds to this renaming will be denoted r.

Let $\mathcal{D}'$ be a dynamic system model

$$(\mathcal{C}_0 \otimes (\otimes\{\mathcal{C}'_i\}_{i\in\{1,\ldots,m\}}), \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \overline{\mathcal{F}}),$$

where

$$\overline{\mathcal{F}} = \mathcal{F} \cup \{(a_1, act, o_2) \mid a_1 \in \{r(1),\ldots r(m)\}, (o_1, act, o_2) \in \mathcal{F}, o_1 \in \mathbb{N}, o_2 \notin \mathbb{N}\} \cup \{(o_1, act, a_2) \mid a_2 \in \{r(1),\ldots r(m)\}, (o_1, act, o_2) \in \mathcal{F}, o_1 \notin \mathbb{N}, o_2 \in \mathbb{N}\} \cup \{(a_1, act, a_1) \mid a_1 \in \{r(1),\ldots r(m)\}, (o_1, act, o_1) \in \mathcal{F}, o_1 \in \mathbb{N}\}.$$

Dynamic system models $\mathcal{D}'$ and $\mathcal{D}'' = (\mathcal{C}_0, \{\mathcal{C}_i\}_{i\in\mathbb{N}}, \overline{\mathcal{F}})$, the set of labels X and the set $X' = Comm(X, \{r(i)\}_{i\in\{1,\ldots,m\}})$ satisfy preconditions of Lemma 5.2 thus $|\mathcal{D}'|_{X'} \leq |\mathcal{D}''|_X$. Because $\mathcal{D}_i = \mathcal{D}''_i$ for each $i \in \mathbb{N}_0$ it holds $d = |\mathcal{D}|_X = |\mathcal{D}''|_X$ and thus $|\mathcal{D}'|_{X'} \leq d$.

Let $\mathcal{D}_{d+m+j} \not\models \varphi_{d+m+j}$, then Definition 5.6 and the first condition from Definition 5.3 imply that there exists a sequence

$$\pi = (q_0^0, \ldots, q_0^{d+m+j}), l_0, (q_1^0, \ldots, q_1^{d+m+j}), l_1, \ldots$$

in $Path_{Inf}^{Init}(\mathcal{D}_{d+m+j})$ and a subformula φ of formula φ_{d+m+j} such that $\pi \not\models \varphi$ and $\neg\varphi \Rightarrow \neg\varphi_j$ and formula φ does not involve automata modelling components with numerical names larger than m.

Formulas $\{\varphi_i\}_{i\in\mathbb{N}}$ and φ after renaming all names of components by the function r will be denoted $r(\varphi_i)$ and $r(\varphi)$. $r(\pi)$ will denote a sequence

$$((q_0^0,\ldots,q_0^m),q_0^{m+1}\ldots,q_0^{d+m+j}),r(l_0),((q_1^0,\ldots,q_1^m),q_1^{m+1}\ldots,q_1^{d+m+j}),r(l_1),\ldots$$

in $Path_{Inf}^{Init}(\mathcal{D}'_{d+j})$, where $r_{((o,a,o'))}$ is a label $_{(f(o),a,f(o'))}$ such that $_{f(o)}=\begin{cases} o & o\in\mathbb{A}\cup\{-\}, \\ r(o) & o\in\{1,\ldots,m\}, \\ o-m & otherwise. \end{cases}$

It is clear that $r(\pi) \not\models r(\varphi)$ and $\neg r(\varphi) \Rightarrow \neg r(\varphi_{d+m+j})$ so it holds $\mathcal{D}'_{d+j} \not\models r(\varphi)$ and $\mathcal{D}'_{d+j} \not\models r(\varphi_{d+m+j})$.

The formula $r(\varphi)$ does not involve clients, is invariant under stuttering. Moreover a sequence of formulas $\{\varphi'_i = r(\varphi)\}_{i\in\mathbb{N}}$ is in the set $Property(\mathcal{D}', 0)$ and X' contains all labels necessary for verification of formulas $\{\varphi'_i\}_{i\in\mathbb{N}_0}$ Consequently Lemma 5.2 claims $\mathcal{D}'_d \not\models r(\varphi'_d)$. So it exists a sequence $\sigma'_d \in Path_{Inf}^{Init}(\mathcal{D}'_d)$ such that $\sigma'_d \not\models \varphi'_d = r(\varphi)$.

Then for the path $r^{-1}(\sigma') \in Path_{Inf}^{Init}(\mathcal{D}_{d+m})$, where r^{-1} is defined inversely to the function r holds $r^{-1}(\sigma') \not\models \varphi_{d+m}$ (for the same reasons as in previous proof of Lemma 5.4). □

Proof. Lemma 6.1 Let $j \in \mathbb{N}$ and $\pi^{|\mathcal{D}|_X+j} \in Path_{Inf}^{Init}(\mathcal{D}_{|\mathcal{D}|_X+j})$ be arbitrary but fixed. From the prerequisite it is clear that a sequence $\pi^{|\mathcal{D}|_X} \in Path_{Inf}^{Init}(\mathcal{D}_{|\mathcal{D}|_X})$ satisfying $\pi^{|\mathcal{D}|_X+j}_{\Delta,X} = \pi^{|\mathcal{D}|_X}_{\Delta,X}$ must exist. For each $i \in \mathbb{N}$ transitions in $\mathcal{D}_i$ between the sets $N_{\mathcal{D},X}$ and $Q \setminus N_{\mathcal{D},X}$ model a communication of a client which involves provider who performs an action from X. So from the definition of $\pi^{|\mathcal{D}|_X+j}$ and $\pi^{|\mathcal{D}|_X}$ it follows that there exist two finite or infinite sequences $0 = i_0^1 < i_1^1 < \cdots$ and $0 = i_0^2 < i_1^2 < \cdots$ such that for each index $n \geq 0$:

- labels $\mathcal{L}(\sigma^{|\mathcal{D}|_X}, i_n^1), \ldots, \mathcal{L}(\sigma^{|\mathcal{D}|_X}, i_{n+1}^1 - 2)$ and $\mathcal{L}(\sigma'^{|\mathcal{D}|_X}, i_n^2), \ldots, \mathcal{L}(\sigma'^{|\mathcal{D}|_X}, i_{n+1}^2 - 2)$ do not correspond to a communication over a label in $Comm(X, \mathbb{N})$,
- $_{(o_1,a,o'_1)=\mathcal{L}(\sigma^{|\mathcal{D}|_X},i_{n+1}^1-1)}$ and $_{(o_2,a,o'_2)=\mathcal{L}(\sigma'^{|\mathcal{D}|_X},i_{n+1}^2-1)}$ are labels from $Comm(X, \mathbb{N}) \cup X$ and moreover $_{(f(o_1),a,f(o'_1))=(f(o_2),a,f(o'_2))}$, where $f(o) = o$ for $o \notin \mathbb{N}$ and $f(o) = *$ otherwise.

Since $\mathcal{L}^{N\rightarrow}_{\mathcal{D},X} \cap \mathcal{L}^{N\leftarrow}_{\mathcal{D},X} = \emptyset$ and no more than $|\mathcal{D}|_X$ clients can be in $Q \setminus N_{\mathcal{D},X}$ for any state of the path $\pi^{|\mathcal{D}|_X}$, the same property must hold for the path $\pi^{|\mathcal{D}|_X+j}$. Thus for an arbitrary $j \in \mathbb{N}$ and $\pi^{|\mathcal{D}|_X+j} \in Path_{Inf}^{Init}(\mathcal{D}_{|\mathcal{D}|_X+j})$ the number of components, which are in $Q \setminus N_{\mathcal{D},X}$ during the run, is at most $|\mathcal{D}|_X$. □

Proof. Lemma 6.2 For a $q \notin N_{\mathcal{D}}$ and $End_{\mathcal{D}}(q) = \{q_e\}$, let us define $Succs(q_e)$ ($AllSuccs(q_e)$) as the set of states $q' \in N^{\leftarrow}_{\mathcal{D}}$ ($q' \in Q$, respectively) such that there is a path in the automaton $\mathcal{C}_1$ not containing any state twice, containing a state from $Ninit^{\rightarrow}_{\mathcal{D}}$ as the very first state only, passing q_e and finishing in q'. Moreover, let $rank(q_e)$ be a number of occurrences of the state q_e in $MaxTrace_{\mathcal{D}}$ and $Succs\_rank(q_e) = \sum_{q\in Succs(q_e)} rank(q)$. Now we describe how to simulate a path π (with an arbitrary number of involved components) by a path π' where at most $1 + \|\mathcal{D}\| \cdot \sum_{\pi\in MaxTrace_{\mathcal{D}}} |\pi|$ components are involved. Observe that for any state $q_e \in Ninit_{\mathcal{D}}$, $Succs\_rank(q_e) = |MaxTrace_{\mathcal{D}}(q_e)|$. The idea of the proof follows from the fact that it suffices to deal with at most $Succs\_rank(q_e) \cdot \|\mathcal{D}\|$ components to simulate any behaviour of any number of components which change their local states among states $AllSuccs(q_e)$.

Firstly let us suppose that π contains infinite many executions of a cycle of service. π' is then constructed iteratively according to the sequence of transitions in π. Let $t = q \xrightarrow{l} q'$ be the first not yet simulated transition in π. If t is a part of a cycle of service, it is executed in π' as well. Otherwise let $q_e \in N^{\leftarrow}_{\mathcal{D}}$ be a state such that $AllSuccs(q_e)$ is the minimal set $AllSuccs(\_)$ containing the state q. If less than $Succs\_rank(q_e)$ local states of components are in $AllSuccs(q_e)$, then t can be directly simulated in π' since the component executing t is below the limit of $Succs\_rank(q_e)$ components allowed to simulate a transition among the states from $AllSuccs(q_e)$. Otherwise (at least $Succs\_rank(q_e)$ local states of components are in $AllSuccs(q_e)$), by an argument based on induction we can mimic the transition by a transition of one of $Succs\_rank(q_e)$ components whose local state is within the set $AllSuccs(q_e)$. The correctness of this simulation follows from the fact that the number $Succs\_rank(q_e)$ covers all possible distinguishable paths within the states in $AllSuccs(q_e)$.

If π contains a finite number of executions of any cycle of service, then it suffices to mimic in π' the finite sequence of cycles of service as occurred in π and then to let a component to execute a loop outside any cycle of service forever. To mimic executions of cycles of services in π we can use at most $\|\mathcal{D}\| \cdot \sum_{\pi\in MaxTrace_{\mathcal{D}}} |\pi|$ components (see the previous paragraph) and a never-ending execution of a loop outside any cycle of service can be simulated by one extra component. □