Error-preserving local transformations on communication protocols

Monika Kapus-Kolar*,†

Jožef Stefan Institute, Department of Communication Systems, Ljubljana, Slovenia

SUMMARY

Recently, two transformations were proposed for modifying a member of a closed system of communicating state machines (CSMs) without considering the other CSMs and nevertheless securing that after the modification, the system can reach the same dead states and overfill the same channels, where channels are assumed to be error-free unidirectional first-in-first-out queues. When verifying general correctness properties of a communication protocol whose specification is such a system, one can employ the two error-preserving local transformations (EPLTs) for simplifying individual CSMs and thereby the subsequent reachability analysis. The paper proves four new simple EPLTs and a generic EPLT which strongly generalizes all the six EPLTs and from which further easily applicable EPLTs can be derived simply by specialization. For each of the EPLTs, it also discusses how (non-)executability of CSM transitions in the new system version reflects (non-)executability of those in the old one.

Received June 23, 2009; Revised October 19, 2010

KEY WORDS: communication protocols; communicating state machines; reachability analysis; logical design errors; model checking

1. INTRODUCTION

The contribution of the paper is twofold. To *practitioners*, it offers four new easily applicable transformations for modifying a member of a closed system of *communicating state machines* (CSMs) without considering the other CSMs and nevertheless securing that after the modification, the system can reach the same dead states and overfill the same channels. The transformations operate on CSMs which are slight generalizations of the *communicating finite state machines* (CFSMs) defined by Bochmann [1]. The assumed type of channels are error-free unidirectional first-in-first-out queues

*Correspondence to: Monika Kapus-Kolar, Jožef Stefan Institute, Jamova 39, SI-1000 Ljubljana, Slovenia.
†Email: monika.kapus-kolar@ijs.si

whose actual capacity to store messages is virtually infinite, but whose nominal capacity is not. When verifying general correctness properties of a communication protocol whose specification is such a system, one can employ the *error-preserving local transformations* (EPLTs) for simplifying individual CSMs and thereby the subsequent reachability analysis of the system, both concrete and symbolic. To *theoreticians*, the paper offers a formal setting for straightforward development of further EPLTs.

As a motivation for studying EPLTs for systems of asynchronously communicating state machines, remember that with the increasing pervasiveness of multi-agent systems, protocols governing agent interaction are becoming the central concept, that geographically distributed agents seldom communicate synchronously, that in global distributed systems, protocols containing deadlocks or overfilling channels can lead to total chaos and security problems, and that state space explosion makes the reachability analysis of asynchronous protocols an extremely complex task. With EPLTs, one can attack the explosion at its very source.

For any CSM (sub)system, the information comprised in its component-wise specification can be re-organized into a specification modelling the (sub)system as one CSM, *the CSM of the (sub)system*. It is this *composite CSM* for whose reachable dead states and channel overfillings one looks in the *reachability analysis* of the (sub)system. In the *advanced reachability analysis*, one virtually performs the search by generating, up to the first error encountered, the reachable part not of the (sub)system CSM, but of a CSM obtained from the CSM by some simplifying EPLTs. One may as well proceed *compositionally*, i.e., first try to simplify, preferably in a context-independent way, i.e. with EPLTs, individual system components and then compose larger and larger sets of components, after every composition trying to simplify the resultant CSM. Here it is desirable to rely on *advanced CSM-composition operators* [2], which can be regarded as operators that for a specific kind of EPLTs directly (and, hence, more efficiently) compute a composite CSM simplified with exhaustive application of the EPLTs. EPLTs are, hence, a foundational concept in every phase of verification.

The paper was motivated by the work of Duan and Chen [3], who innovatively proposed two EPLTs for systems of CSMs differing from the classical CFSM definition [1] in that *an individual transition could denote un-interruptible and atomic execution of a sequence of multiple message receptions and/or transmissions*. Their motivation for considering a more general kind of CSMs was that in the reachability analysis, it is sometimes acceptable to treat specific sequences of events as one atomic action [4]. Both EPLTs, in the following called Θ_1 and Θ_2, are very elementary, but when applied to a CSM in an adequate combination, they facilitate *elimination of its globally insignificant states and exploitation of commutation of its mutually independent transitions*. It is, hence, not surprising that, as demonstrated by Duan and Chen [3], their application to individual system components complements *partial order reduction* (POR) [5,6], the most typical EPLT applied *during* reachability analysis, whose essence is exploitation of commutation of mutually independent transitions belonging to *different* CSMs of the considered system. Pre-processing with Θ_1 and Θ_2 is helpful [3] also if the employed reachability analysis technique is *simultaneous reachability analysis* (SRA) [7, 8], another technique exploiting commutation of transitions belonging to different system components. After exhaustive pre-processing with Θ_1 and Θ_2, there is, however, no need to use the more advanced *blocking* SRA (BSRA) [4], whose advantage is in treating specific sequences of transitions of individual system components as compound transitions, where the pre-processing already transforms every such sequence into one transition.

As a general rule, pre-processing with a specific combination of EPLTs is helpful if the EPLTs complement the EPLTs employed during the reachability analysis. As different reachability analysis

DOI: 10.1002/stvr

techniques rely on different EPLTs and new techniques are being invented, it is desirable to have both a variety of easily applicable EPLTs to choose from for the pre-processing and a formal setting for easy development of further useful EPLTs. The paper drops the assumption of Duan and Chen [3] that every system component is deterministic and minimal and has only a finite number of specified states and transitions, at most one outgoing channel per destination, at most one incoming channel per source and no private channels. It proves that Θ_1 and Θ_2 nevertheless remain error-preserving. Furthermore, it proposes *four new simple EPLTs*, in the following called Θ_3 to Θ_6, and a *generic EPLT* Θ which strongly generalizes Θ_1 to Θ_6 and *from which further easily applicable EPLTs can be derived simply by specialization*. For all the transformations, the paper, like Duan and Chen [3], also discusses how (non-)executability of CSM transitions in the new system version reflects (non-)executability of those in the old one.

The rest of the paper is organized as follows. Section 2 presents the basic terminology and notation used throughout the paper. Section 3 discusses the two old and presents the four new elementary EPLTs. Section 4 defines and studies the generic EPLT. Because of the complexity of the subject, the section is unavoidably rather mathematical (though with the essence of every formal statement presented also informally) and as such intended primarily for theoreticians planning to develop further practically interesting EPLTs. Section 5 brings a wider discussion of the potentials of the generic EPLT and its specializations and results of an extensive empirical study of the new elementary EPLTs.

2. BASIC TERMINOLOGY AND NOTATION

2.1. Sequences

A *sequence* is an $o_1 o_2 \dots o_k$ with o_1 to o_k objects of any kind. Where an o will not denote the object o, but the sequence comprising the object as its only element, this will be evident from the context.

For a sequence $\bar{o} = o_1 \dots o_k$, let $ln(\bar{o})$ denote its *length* k. Let ε denote an *empty sequence*, i.e., a sequence of length 0. For a non-empty sequence $\bar{o}$, let $first(\bar{o})$ denote its *first element* and $last(\bar{o})$ its *last element*.

For two sequences $\bar{o} = o_1 \dots o_k$ and $\bar{o}' = o'_1 \dots o'_{k'}$, let $\bar{o} \cdot \bar{o}'$ denote their *concatenation*, i.e., the sequence $o_1 \dots o_k o'_1 \dots o'_{k'}$. For two sequences $\bar{o}$ and $\bar{o}'$, let $\bar{o} \leq \bar{o}'$ denote that $\bar{o}$ is a *prefix* of $\bar{o}'$, i.e., that $\bar{o}'$ is an $\bar{o} \cdot \bar{o}''$.

For a sequence $\bar{o}$ and a set O, let $\bar{o}|_O$ denote the *projection* of $\bar{o}$ on O, i.e., the sequence obtained from $\bar{o}$ by deleting every element not belonging to O. For a sequence $\bar{o}$, let $set(\bar{o})$ denote *the set of all the objects of which* $\bar{o}$ *comprises an instance*, i.e., the smallest set O with $\bar{o}|_O = \bar{o}$.

2.2. Rooted directed graphs

A rooted directed graph (RDG) g is a triplet (V, E, v^*) with V the set $vr(g)$ of its *vertices*, E the set $ed(g)$ of its *edges* and v^* the member of V designated as its *initial vertex* (the root) $init(g)$. Every edge e in E is assumed to be a triplet (v, l, v') with v its *initial vertex* $init(e)$, a member of V, with v' its *final vertex* $fin(e)$, also a member of V, and with l its *label* $lab(e)$, a finite sequence of objects of any kind. The convention adopted in the paper for drawing an RDG is to depict every vertex as a frame carrying its name and every edge (v, l, v') as an arrow directed from the v frame to v' frame

DOI: 10.1002/stvr

and labelled l, with the root frame marked with an incoming arrow with an unspecified source (see, for example, the two RDGs in Figure 1 (Section 3), in which vertices are depicted as circles).

For a vertex v of an RDG g, let $in(v)$ denote the set comprising every edge e in $ed(g)$ with $fin(e) = v$, and $out(v)$ the set comprising every edge e in $ed(g)$ with $init(g) = v$.

A sequence $w = (v_1, l_1, v_2)(v_2, l_2, v_3) \ldots (v_k, l_k, v_{k+1})$ of consecutive edges of an RDG g is a *walk* of g, for which v_1 is the *initial vertex* $init(w)$, $l_1 \cdot \ldots \cdot l_k$ is the *label* $lab(w)$ and l_{k+1} is the *final vertex* $fin(w)$. In the degenerated case of $k = 0$, $init(w)$ and $fin(w)$ are equal and assumed to be known from the context. If $init(w) = init(g)$, the walk is *rooted*. A walk w of an RDG g is *executable* if it is a segment of a rooted walk of g.

A vertex v of an RDG g is *reachable* if g has a rooted walk w with $fin(w) = v$. If in an RDG g, every vertex in $vr(g)$ is reachable, g is *initially connected*. For an RDG g, let $rch(g)$ denote its *reachable part*, i.e., the largest RDG among its initially connected sub-RDGs.

2.3. State machines

A *state machine* (SM) is a machine which is at any time either idling in a *state*, at the beginning in its pre-defined *initial state*, or executing a *transition* to its next state, which might as well be the same as the previous one. For an SM p, let $st(p)$ denote the set of its specified states, $tr(p)$ the set of its specified transitions and $init(p)$ its initial state. A transition t is an (s, η, s') with s its *initial state* $init(t)$, s' its *final state* $fin(t)$ and η its *label* $lab(t)$, a finite sequence specifying the work assumed to be accomplished during the transition. It is assumed that no transition is ever initiated before all the resources necessary for its completion have been secured. If such a resource is one for whose provision the environment of the SM is responsible, the resource is assumed to be available by definition.

For an SM p, let $rdg(p)$ denote the RGD $(st(p), tr(p), init(p))$. All pictures in the paper said to show an SM actually show its RDG. Moreover, whenever the paper in the following for an SM p refers to an attribute originally defined for RDGs, this should be interpreted as a reference to that attribute of $rdg(p)$. For example, $rch(p)$ is used as an abbreviation for $rch(rdg(p))$, and "SM p can reach" as an abbreviation for "$rdg(p)$ can reach".

2.4. Channels

A *channel* is a first-in-first-out queue of *messages*. For a channel c, let $msg(c)$ denote the set of the messages whose instances it is able to store. Without loss of generality, the paper assumes $msg(c) \cap msg(c') = \emptyset$ for every other channel c'.

Initially, every channel is empty. To append to the tail of a channel c a message m, one has to execute a *transmission* $-m$, with m assumed to be in $msg(c)$. To remove from a non-empty channel c its head message, an m, one has to execute a *reception* $+m$. It is assumed that no channel c ever blocks a transmission of a message m in $msg(c)$. It is, however, considered an error to make a channel c comprise more messages than its *nominal capacity* $nc(c)$.

Transmissions and receptions are two kinds of *events*, the atoms of behaviour. The paper assumes that every event is instantaneous, virtually non-overlapping with other events and either a transmission or a reception. For a channel c, let $ev^-(c)$ denote the set comprising every transmission $-m$ with m in $msg(c)$, $ev^+(c)$ the set comprising every reception $+m$ with m in $msg(c)$, and $ev(c)$ the event set

DOI: 10.1002/stvr

$ev^-(c) \cup ev^+(c)$. For an event sequence η which is a $+m_1 \ldots + m_k$ or a $-m_1 \ldots - m_k$, let $msg(\eta)$ denote the message sequence $m_1 \ldots m_k$.

Alternatively, a channel may be regarded as an SM whose current state is its current contents and which undergoes a transition whenever a transmission or a reception is executed on it. For a channel c, $st(c)$ comprises every sequence of messages in $msg(c)$, with $init(c) = \varepsilon$.

2.5. Communicating state machines

A *communicating state machine* (CSM) is an SM connected to some channels and acting as their user. For a CSM, the label of every specified transition is assumed to be a non-empty sequence of events, with each of the events either a transmission on one of its *outgoing channels* or a reception on one of its *incoming channels*. On each of its outgoing channels c, a CSM is assumed to be the only transmitter, call it $csm^-(c)$. On each its incoming channels c, a CSM is assumed to be the only receiver, call it $csm^+(c)$. A channel which is both an outgoing and an incoming channel of a CSM is, hence, its *private channel*. For a CSM p, let $ch^-(p)$ denote the set of its outgoing channels, $ch^+(p)$ the set of its incoming channels, $ch(p)$ the channel set $ch^-(p) \cup ch^+(p)$, $pch(p)$ the set of its private channels, $sch(p)$ the set $ch(p) \setminus pch(p)$ of its *shared channels*, $sch^-(p)$ the channel set $sch(p) \cap ch^-(p)$ and $sch^+(p)$ the channel set $sch(p) \cap ch^+(p)$.

For a CSM p, let $ev^-(p)$ denote the set comprising every transmission in the labels of its specified transitions, $ev^+(p)$ the set comprising every reception in the labels of its specified transitions, $ev(p)$ the event set $ev^-(p) \cup ev^+(p)$, $sev^-(p)$ the set comprising every transmission in $ev^-(p)$ on a channel in $sch^-(p)$, $sev^+(p)$ the set comprising every reception in $ev^+(p)$ on a channel in $sch^+(p)$, $sev(p)$ the event set $sev^-(p) \cup sev^+(p)$, and $pev(p)$ the event set $ev(p) \setminus sev(p)$. Initiation of a transition t of a CSM requires that for every private channel c, the current contents μ_c allows execution of the entire $lab(t)|_{ev(c)}$, i.e., that for every prefix η of $lab(t)$, μ_c satisfies $msg(\eta|_{ev^+(c)}) \leq \mu_c \cdot msg(\eta|_{ev^-(c)})$.

Like Duan and Chen [3] implicitly do, the paper for every CSM p and channel c in $pch(p)$ assumes that for every state s in $st(p)$, the specification comprises also the specification, call it $s|_c$, of the contents which c has when p is in s, with $s = init(p)$ implying $s|_c = \varepsilon$. With state information comprising the information on the contents of every channel critical for transition execution, the paper in addition assumes that for every state s in $st(p)$, $out(s)$ comprises only transitions which are actually executable from s and that the final state of the transitions reflects the new contents of the channels, i.e., that every transition t in $tr(p)$ for every channel c in $pch(p)$ satisfies $msg(\eta|_{ev^+(c)}) \leq init(t)|_c \cdot msg(\eta|_{ev^-(c)})$ for every prefix η of $lab(t)$ and $msg(lab(t)|_{ev^+(c)}) \cdot fin(t)|_c = init(t)|_c \cdot msg(lab(t)|_{ev^-(c)})$.

If for a transition t of a CSM p, $set(lab(t)) \subseteq ev^-(p)$, the paper calls t *transmission transition*. If instead $lab(t)|_{sev^+(p)} \neq \varepsilon$, t is called *blocking transition*. If for a state s, $out(s)$ comprises no non-blocking transition, s is called *stable*. A CSM state s with $out(s)$ empty is *dead*. A walk w of a CSM p *overfills* a specific channel c if c is in $pch(p)$ and $ln(init(w)|_c) + ln(\eta|_{ev^-(c)}) - ln(\eta|_{ev^+(c)}) > nc(c)$ for a prefix η of $lab(w)$. A CSM p overfills a specific channel c if one of its rooted walks does.

Two events in the label of a transition t of a CSM p are *complementary* if for a message m, a member of $msg(c)$ of a private channel c of p, they are the transmission and the reception of the same instance of m, i.e., if in the case of their deletion from $lab(t)$, t will still satisfy $msg(\eta|_{ev^+(c)}) \leq init(t)|_c \cdot msg(\eta|_{ev^-(c)})$ for every prefix η of $lab(t)$ and $msg(lab(t)|_{ev^+(c)}) \cdot fin(t)|_c = init(t)|_c \cdot msg(lab(t)|_{ev^-(c)})$.

DOI: 10.1002/stvr

2.6. Systems of communicating state machines

Every non-empty set P of CSMs with $ch^-(p) \cap ch^-(p')$ and $ch^+(p) \cap ch^+(p')$ empty for every two different CSMs p and p' in P defines a ***CSM system***, call it $sys(P)$. In the following, the term system denotes a CSM system.

A system $sys(P)$ is a network consisting of the CSMs in P, in the following called its *components*, and of those channels which link them. For a system q, let $cmp(q)$ denote the set comprising its components and $lch(q)$ the set comprising its *link channels*, i.e, every channel c in $ch^-(p) \cap ch^+(p')$ for two different CSMs p and p' in $cmp(q)$. A system q evolves by letting the CSMs in $cmp(q)$ execute their transitions when the contents of their private channels and the channels in $lch(q)$ allows. The paper adopts the very common assumption that at any time, at most one component of a system is executing a transition.

For a system q, the information comprised in its component-wise specification can always be reorganized into a specification modelling the system as one CSM. Call the composite CSM $csm(q)$, *the CSM of the system* q. Whenever the paper in the following for a system q refers to an attribute originally defined for (C)SMs, this should be interpreted as a reference to that attribute of $csm(q)$. For example, $st(q)$ is used as an abbreviation for $st(csm(q))$, and "system q can reach" as an abbreviation for "$csm(q)$ can reach".

The incoming channels of a system are the incoming channels of its individual components. The outgoing channels of a system are the outgoing channels of its individual components. The private channels of a system are, hence, its link channels and the private channels of its individual components. If a system q has no shared channels, it is *closed*. Otherwise it is *open* and assumed to be a subsystem of a closed system whose components other than the members of $cmp(q)$ are unspecified. A stable state of a closed system is a *dead state*. For a component p of a closed system q, let $vp(p, q)$ denote $sys(cmp(q) \setminus \{p\})$, the *virtual partner* of p in q.

The current state of a system is denoted by the current state of its individual components (which, by assumption, precisely denotes also the current state of their private channels) and the current state of its individual link channels, i.e., of its remaining private channels. The specified states set $st(q)$ of a system q is the Cartesian product of the specified states sets of the elements of $cmp(q) \cup lch(q)$ for the default ordering of the elements. For a state s of a system q and a CSM p in $cmp(q)$, let $s|_p$ denote the state in which p resides when q is in s. For a state s of a system q and a channel c in $lch(q)$, let $s|_c$ denote the contents which c has when q is in s. The initial state of a system q is the state s in $st(q)$ with $s|_p = init(p)$ for every CSM p in $cmp(q)$ and $s|_c = \varepsilon$ for every channel c in $lch(q)$.

The specified transitions of a system q comprise every transition (s, η, s') with s and s' in $st(q)$ which for a CSM p in $cmp(q)$ satisfy all the following: (1) $(s|_p, \eta, s'|_p) \in tr(p)$, (2) $msg(\eta|_{ev^+(c)}) \cdot s'|_c = s|_c \cdot msg(\eta|_{ev^-(c)})$ for every channel c in $lch(q)$ and (3) $s|_{p'} = s'|_{p'}$ for every CSM p' in $cmp(q) \setminus \{p\}$. For a system transition t, with the events in $lab(t)$ belonging to a system component p, let $ct(t)$ denote the component's transition $(init(t)|_p, \eta, fin(t)|_p)$. For a system walk $w = t_1 \dots t_k$, let $ct(w)$ denote the sequence $ct(t_1) \dots ct(t_k)$. For a system walk w and a component p of the system, let $cw(w, p)$ denote the walk which p executes within w, i.e. the sequence $ct(w)|_{tr(p)}$.

A state s of a component p of a system q is *reachable in* q if q has a reachable state s' with $s'|_p = s$. A walk w of a component p of a system q is *executable in* q if q has a rooted walk w' with w a segment of $cw(w', p)$.

DOI: 10.1002/stvr

2.7. System transformations

A system transformation is *local* if in any application, (1) all that it does is to modify the RDG of a component of the target system and (2) the modification in no way depends on the nature of the other system components. If only the first condition is satisfied in every application, the transformation is *quasi-local*.

A system transformation is *error-preserving* if in any application, the old and the new version of the target system exhibit the same errors of the considered class, where *the default class of errors* comprises the dead states which the considered system can reach and the channels which it can overfill. A well-known error-preserving local transformation (EPLT), call it Θ_{rch}, is that for the target system component p, $rdg(p)$ is simplified into $rch(p)$. A well-known non-local error-preserving transformation is that for a system component p of the target system q, a state of p not reachable in q is deleted from $st(p)$ or a transition of p not executable in q is deleted from $tr(p)$.

A system transformation is *semantically error-preserving* if in any application, the knowledge of which errors of the considered class the new version of the target system exhibits and of what the transformation was *suffices to deduce* which errors of the class the old version exhibits. A well-known semantically error-preserving local transformation (SEPLT), call it Θ_{mes}, is to merge two (relatively) equivalent states of the target system component. A well-known non-local semantically error-preserving transformation (SEPT), call it Θ_{csm}, is to modify the target system q into the system $sys((cmp(q) \setminus P) \cup \{csm(P)\})$ for a subset P of $cmp(q)$ with $|P| \geq 2$, i.e., to start regarding the target subsystem as a composite CSM.

A system transformation is *symmetric* if in any application, the reverse transformation is also one of its instances.

3. THE TWO OLD AND FOUR NEW SIMPLE EPLTs

3.1. Introduction

Sections 3.2 to 3.7 describe Θ_1 to Θ_6, respectively, and show that with the four additional transformations, the simplification of a system can go much further. Section 3.8 details to which extent the transformations preserve the system properties of interest.

As an example, Sections 3.2 to 3.5 gradually simplify a closed two-party system $sys(M, N)$ whose consecutive versions $sys(M_1, N_1)$ to $sys(M_9, N_9)$ are given in Figures 1 to 9, in which the transitions targeted in the next transformation of individual system components are those drawn dashed. The CSMs M and N, mutually connected by one channel per direction, specify the X.25 call establishment/clear protocol, more precisely, its maximally rich version, whose considered CSM-based specification is that of Gouda and Yu [9]. The paper of Duan and Chen [3] also uses the protocol as a benchmark, but obviously a simplified version, because the there reported number of the reachable system states is less than that of the system $sys(M_1, N_1)$. The messages which M and N exchange have the following meaning: a stands for call request, b for call connected, c for incoming call, d for call accepted, e for clear request, f for clear indication and g for clear confirmation.

DOI: 10.1002/stvr

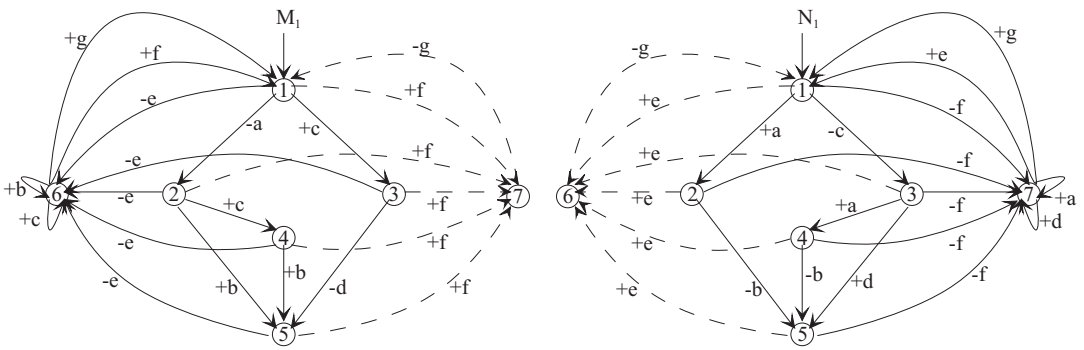

Figure 1. Version 1 of the X.25 call establishment/clear protocol.

3.2. State reduction

The first transformation of Duan and Chen [3], Θ_1, is called state reduction, because in the RDG of the target component p of the target closed system, one *by-passes and deletes an unstable state*, call it s. The considered part of $rdg(p)$ comprises s (call it the *target state*) and the transitions in $in(s) \cup out(s)$ (call them the *target transitions*). It is implicitly assumed that s is not $init(p)$, the state present in $st(p)$ by definition. One also assumes $init(t) \neq s$ for every transition t in $in(s)$, and thereby $fin(t') \neq s$ for every transition t' in $out(s)$. The third assumption is that $out(s)$ comprises exclusively transmission transitions, i.e., that p *autonomously decides which of the transitions to execute* and could as well take the decision before reaching s. The essence of the transformation is exactly *to speed up the decision and to combine the selected transition with the preceding one*: One deletes s and the transitions in $in(s) \cup out(s)$ after for every transition t in $in(s)$ and transition t' in $out(s)$ replacing the walk tt' with a transition $t_{tt'} = (init(tt'), lab(tt'), fin(tt'))$.

The system $sys(M_1, N_1)$ (Figure 1) has 111 reachable states and 241 executable transitions. By applying Θ_1 to the state 7 of M_1 and the state 6 of N_1, one obtains the system $sys(M_2, N_2)$ in Figure 2, which has only 104 reachable states and 229 executable transitions.

3.3. Transition reduction

The second transformation of Duan and Chen [3], Θ_2, is called transition reduction, because in the RDG of the target component p of the target closed system, one *deletes a transition*, call it t, the *target transition*. It is assumed that $rdg(p)$ has one or more walks w not comprising t which satisfy $init(w) = init(t)$, $fin(w) = fin(t)$ and $lab(w)|_{ev(c)} = lab(t)_{ev(c)}$ for every channel c, i.e., that p *autonomously decides whether to execute the events in* $lab(t)$ *by executing* t *or by executing such a by-pass walk* w and may as well decide to never chose execution of t, as all the alternatives are equivalent. The essence of the EPLT is exactly *to implement such a priori decision*.

DOI: 10.1002/stvr

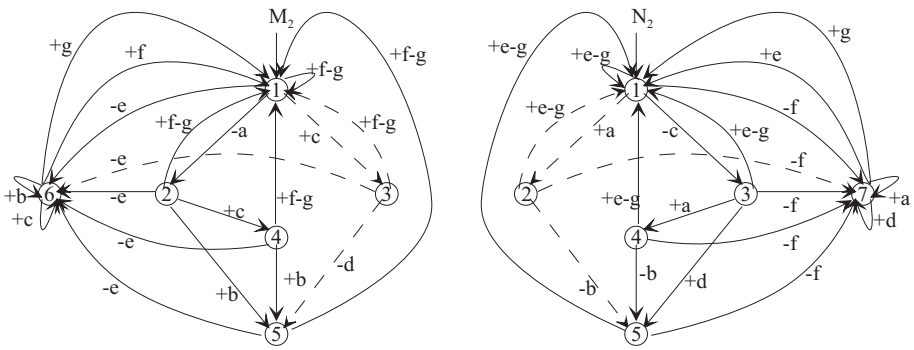

Figure 2. Version 2 of the X.25 call establishment/clear protocol.

Returning to the system $sys(M_2, N_2)$ (Figure 2), observe that neither Θ_1 nor Θ_2 is applicable to M_2 or N_2. $sys(M_2, N_2)$ is, hence, where the simplification of $sys(M_1, N_1)$ stops if one relies solely on the ideas of Duan and Chen [3]. With Θ_3, however, one in the following obtains versions of M and N to which Θ_2 can be applied.

3.4. More general state reduction

The first new transformation, Θ_3, is defined on the target component p of the target closed system as Θ_1 with one of its assumptions slightly weakened: Instead of assuming for the target non-initial and unstable state s in $st(p)$ that $out(s)$ comprises solely transmission transitions, one for every transition t in $in(s)$ and transition t' in $out(s)$ assumes that the execution of t includes no transmissions to the other system components or that the execution of t' includes no receptions from the other system components, formally $(lab(t)|_{sev^-(p)} = \varepsilon) \vee (lab(t')|_{sev^+(p)} = \varepsilon)$. Θ_3 is, hence, a generalization of Θ_1. The assumption secures that *s is in no walk of p a point in which the initiative is passed from p to the rest of the system*, for otherwise the state would be globally essential and its elimination inappropriate.

Consider again the system $sys(M_2, N_2)$ (Figure 2). By applying Θ_3 to the state 3 of M_2 and the state 2 of N_2, one obtains the system $sys(M_3, N_3)$ in Figure 3, which has only 100 reachable states and 222 executable transitions. One can then apply Θ_2 to the dashed transitions of M_3 and N_3, producing the system $sys(M_4, N_4)$ in Figure 4, which has 100 reachable states and only 219 executable transitions. By applying Θ_3 to the state 4 of M_4 and the state 4 of N_4, one obtains the system $sys(M_5, N_5)$ in Figure 5, which has only 91 reachable states and 201 executable transitions. One can then apply Θ_2 to the dashed transitions of M_5 and N_5, producing the system $sys(M_6, N_6)$ in Figure 6, which has 91 reachable states and only 193 executable transitions.

DOI: 10.1002/stvr

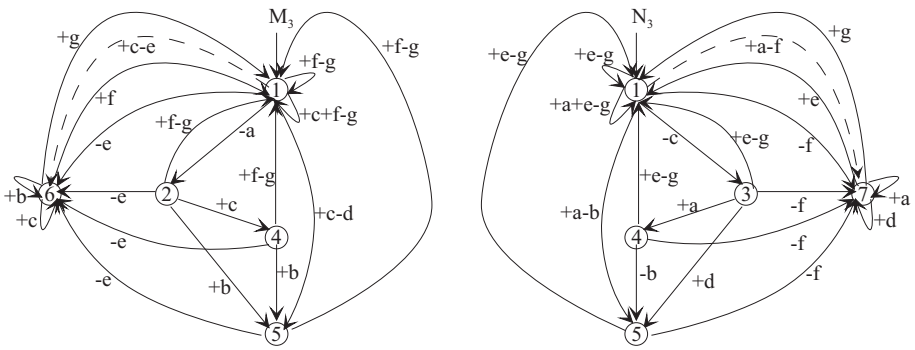

Figure 3. Version 3 of the X.25 call establishment/clear protocol.

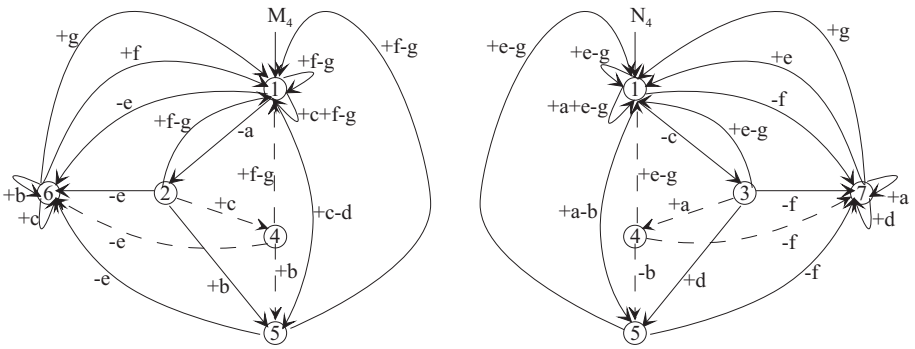

Figure 4. Version 4 of the X.25 call establishment/clear protocol.

3.5. Partial by-passing

The second new transformation, Θ_4, in the RDG of the target component p of the target closed system targets an unstable state, call it s, which cannot be eliminated by Θ_1 or Θ_3. The essence of Θ_4 is *to by-pass s from a selected set of directions which allow that*. The *target transitions* are those in $out(s)$ and those in a selected set T of those transitions t in $in(s)$ which satisfy $lab(t)|_{sev^-(p)} = \varepsilon$ and, hence, also the expected $(lab(t)|_{sev^-(p)} = \varepsilon) \vee (lab(t')|_{sev^+(p)} = \varepsilon)$ for every t' in $out(s)$. One deletes the transitions in T, for every such transition t and transition t' in $out(s)$ replacing the walk tt' with a transition $t_{tt'} = (init(tt'), lab(tt'), fin(tt'))$. Unlike in Θ_1 or Θ_3, s is not eliminated and may, hence, as well be $init(p)$. Besides, the requirement $fin(t') \neq s$ is dropped for those transitions t' in $out(s)$ which satisfy $lab(t')|_{sev^-(p)} \neq \varepsilon$.

DOI: 10.1002/stvr

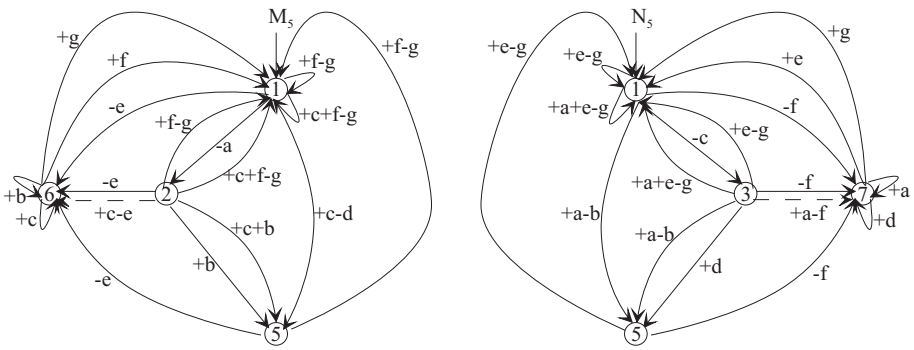

Figure 5. Version 5 of the X.25 call establishment/clear protocol.

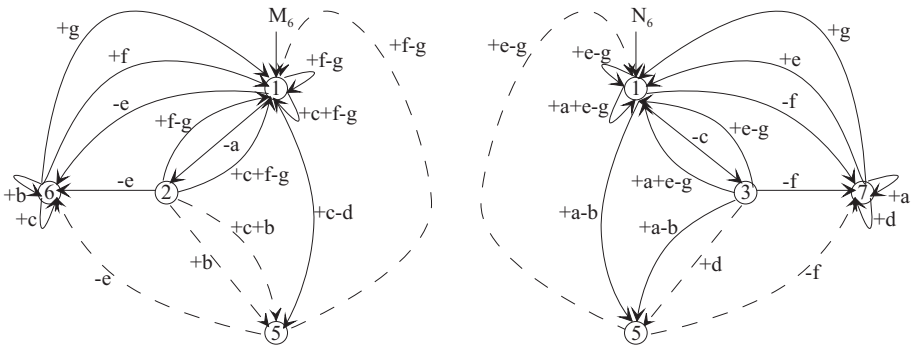

Figure 6. Version 6 of the X.25 call establishment/clear protocol.

By exhaustively applying Θ_4 to the state 5 of M_6 and the state 5 of N_6, one obtains the system $sys(M_7, N_7)$ in Figure 7, which has only 90 reachable states, although 194 executable transitions. One can then apply Θ_2 to the dashed transitions of M_7 and N_7, producing the system $sys(M_8, N_8)$ in Figure 8, which has 90 reachable states and only 188 executable transitions. By exhaustively applying Θ_4 to the state 1 of M_8 and the state 1 of N_8, one obtains the system $sys(M_9, N_9)$ in Figure 9, which has only 83 reachable states, although 204 executable transitions.

3.6. More general transition reduction

The third new transformation, Θ_5, is defined as Θ_2 with weaker assumptions. Θ_5 is, hence, a generalization of Θ_2, one *exploiting the specific nature of the private-channel events* of the target

DOI: 10.1002/stvr

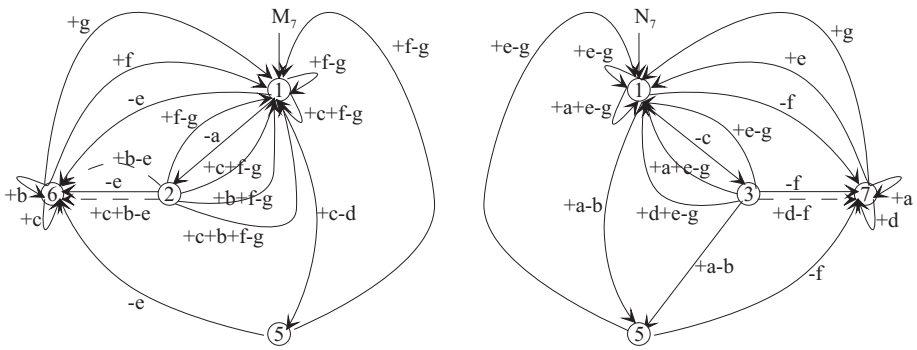

Figure 7. Version 7 of the X.25 call establishment/clear protocol.

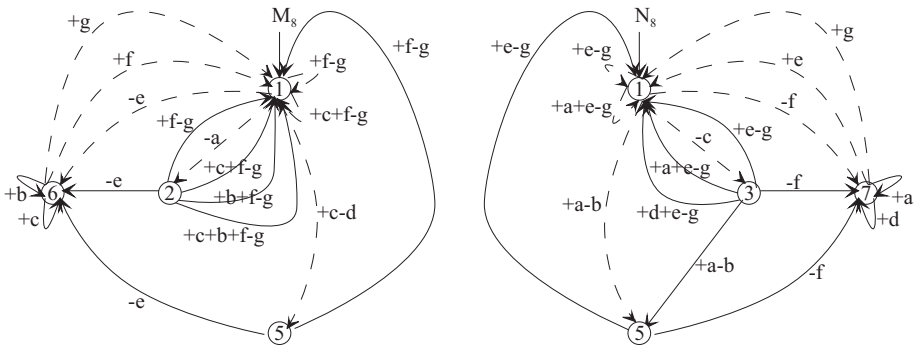

Figure 8. Version 8 of the X.25 call establishment/clear protocol.

component p of the target closed system. In Θ_5, one deletes a transition t in $st(p)$ under the following assumptions:

1. For t, $rdg(p)$ has *a by-pass walk representing an equivalent behaviour on the shared channels of* p, i.e., a walk w not comprising t which satisfies $init(w) = init(t)$, $fin(w) = fin(t)$ and $lab(w)|_{ev(c)} = lab(t)|_{ev(c)}$ for every channel c in $sch(p)$.
2. For every private channel c of p which t overfills, $rdg(p)$ has for t *an alternative walk which also overfills* c *and is executable whenever* t *is*, i.e., a walk w_c not comprising t which also overfills c and satisfies $init(w_c) = init(t)$ and $lab(w_c)|_{ev(c')} \leq lab(t)|_{ev(c')}$ for every channel c' in $sch^+(p)$.

As an example, consider the initiator/responder system [10] in Figure 10. Note that the system is open, i.e., a subsystem of a closed system whose members are at least the initiator CSM I, the

DOI: 10.1002/stvr

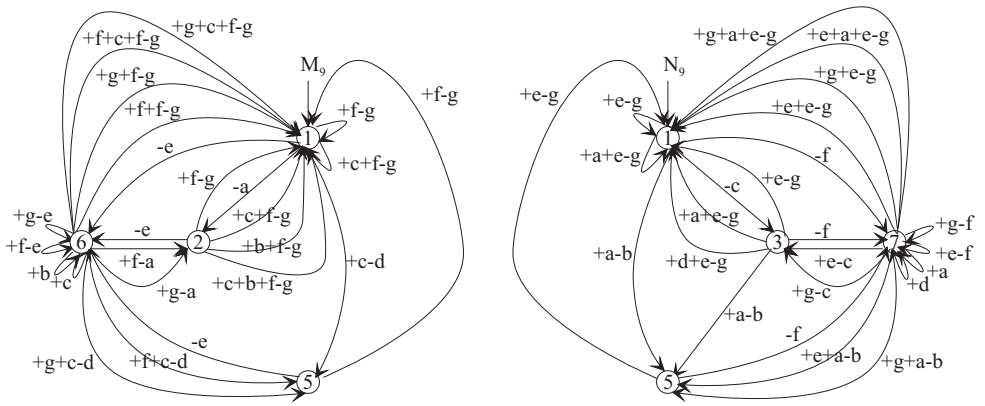

Figure 9. Version 9 of the X.25 call establishment/clear protocol.

responder CSM R and the unspecified users of the service which they provide in cooperation. The service messages have the following meaning: a_1 stands for connect request, a_2 for data request, b_1 for connect confirmation, b_2 for disconnect confirmation, c_1 for connect response, c_2 for disconnect request, d_1 for connect indication and d_2 for data indication. The messages exchanged by I and R have the following meaning: i_1 stands for connect request, i_2 for data, j_1 for connect confirmation, j_2 for disconnect request and j_3 for acknowledgement.

In the figure, I_1 and R_1 are the initial versions of I and R, respectively. By applying Θ_4 to the state 3 of I_1 and Θ_3 to the state 4 of R_1, one obtains the system $sys(I_2, R_2)$. A SEPT is to replace $sys(I_2, R_2)$ with $csm(I_2, R_2)$, of which Figure 10 shows a segment C_1. By applying Θ_3 to the states $23[][j_1]$ and $21[][j_1j_2]$, one transforms C_1 into C_2. Assuming that the nominal capacity of the two private channels is just 1, the vertical outgoing transition of the state $22[][]$ in C_2 cannot be deleted by Θ_5, unlike the vertical outgoing transition of the state $33[][]$ in C_3, produced by applying Θ_4 to the state $31[][j_2]$ of C_2.

3.7. Transition label simplification

In the fourth new transformation, Θ_6, one from the label of a transition t in the RDG of the target component p of the target closed system *deletes a pair of complementary events* on a private channel c of p, under the following assumption: If t overfills c and its new version, call it t', does not, $rdg(p)$ has for t *an alternative walk which also overfills* c *and is executable whenever* t *is*, i.e., a walk w not comprising t which also overfills c and satisfies $init(w) = init(t)$ and $lab(w)|_{ev(c')} \leq lab(t)|_{ev(c')}$ for every channel c' in $sch^+(p)$.

As an example, consider the CSM segment C_4 in Figure 10. It has been obtained from C_3 by an application of Θ_5 and an application of Θ_6.

DOI: 10.1002/stvr

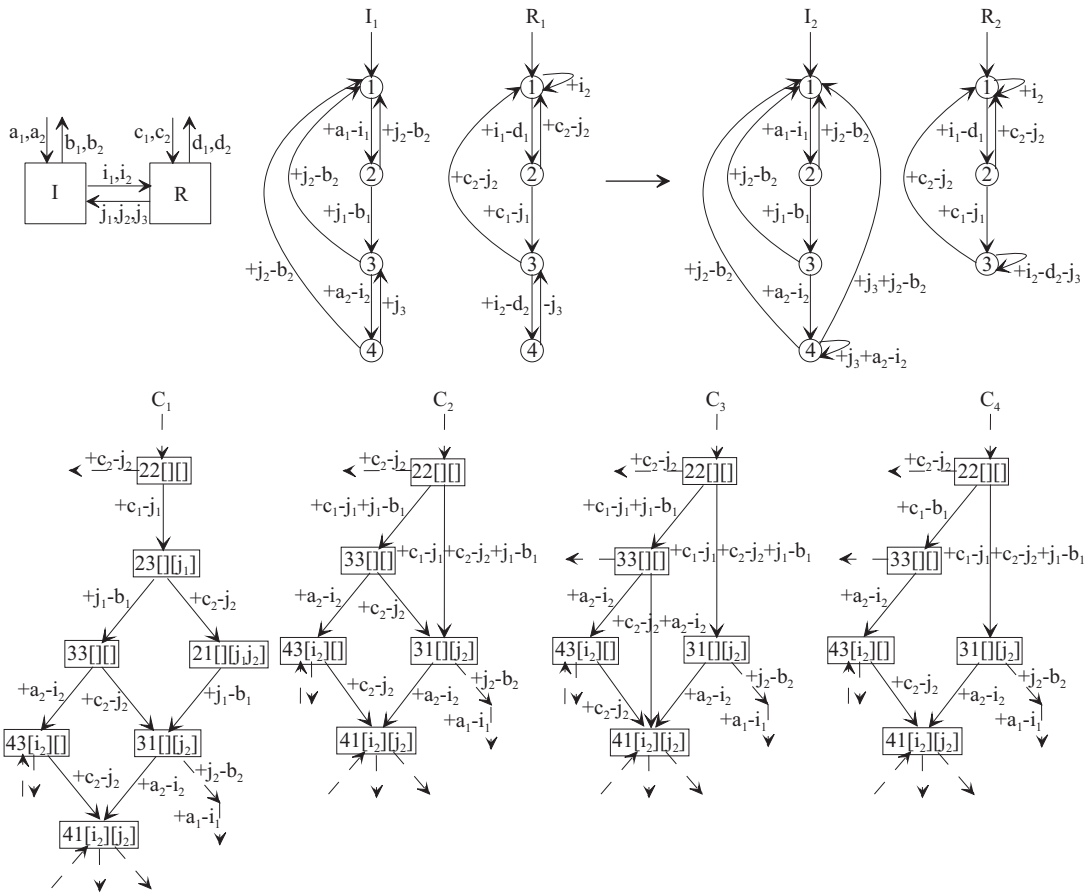

Figure 10. InRes: the system architecture, the original version $sys(I_1, R_1)$ of $sys(I, R)$, the result $sys(I_2, R_2)$ of local state reductions, a segment C_1 of $csm(I_2, R_2)$ and its consecutive simplified versions C_2, C_3 and C_4. In the second row, system states are composed of the state of I, the state of R, the contents of the I-to-R channel and the contents of the R-to-I channel.

3.8. Preserved system properties

Section 4 interprets Θ_1 to Θ_6 as specializations of the generic EPLT Θ. Under the interpretation, the Theorems 1-6 on Θ in Section 4.4 imply the following properties of Θ_1 to Θ_6, where q_1, q_2, p_1 and p_2 denote, respectively, the old and the new version of the target closed system q and the old and the new version of the target component p of q:

1. A specific system state is reachable and dead in q_2 if and only if it is in q_1.
2. q_2 can overfill a specific channel if and only if q_1 can.

Prepared using stvrauth.cls
DOI: 10.1002/stvr

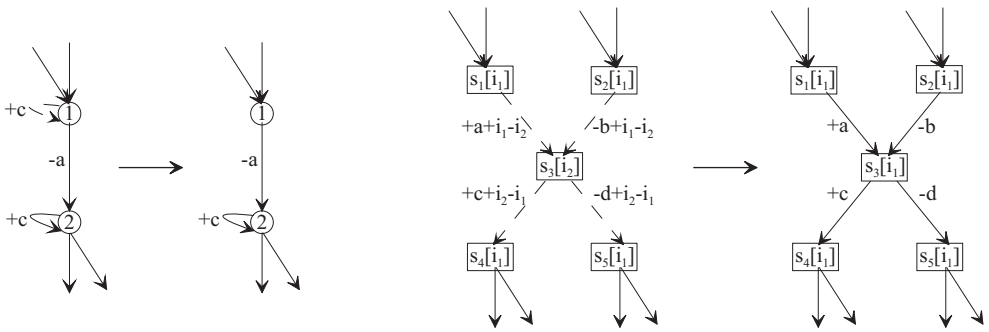

Figure 11. Two EPLTs beyond the EPLTs Θ_1 to Θ_6, but not beyond Θ. In the second example, the square brackets contents in the state names denotes the contents of the only private channel of the target CSM.

3. A specific transition of a CSM in $cmp(q) \setminus \{p\}$ is executable in q_2 if and only if it is in q_1.
4. A specific non-target transition of p is executable in q_2 if and only if it is in q_1.
5. If the EPLT is Θ_1, Θ_3 or Θ_4, a specific transition $t_{tt'}$ in $tr(p_2) \setminus tr(p_1)$ is executable in q_2 if and only if the corresponding walk tt' of p_1 is executable in q_1.
6. If the EPLT is Θ_4, a specific transition in $out(s)$ is executable in q_2 only if it is executable in q_1.
7. If the EPLT is Θ_6, the target transition t is executable in q_1 if and only if its new version t' is executable in q_2.

4. A GENERIC ERROR-PRESERVING LOCAL TRANSFORMATION

4.1. Introduction

Figure 11 shows two simple EPLTs which go beyond what the EPLTs Θ_1 to Θ_6 can do. Obviously, it is possible, and highly desirable, to develop further easily applicable EPLTs. To simplify the development, the paper in the following proposes a *generic EPLT*, Θ, which is a strong generalization of the EPLTs Θ_1 to Θ_6 and from which further simple EPLTs can be derived as specializations, so that one knows that they inherit all the nice properties proven for Θ.

Primarily intended as a template for EPLT development, Θ is deliberately extremely abstract, so that it represents a large class of specific EPLTs. Its definition in Section 4.4 and the proofs of its properties in the Appendix are based on some yet undefined concepts, introduced in Sections 4.2 and 4.3. Section 4 terminates by interpreting Θ_1 to Θ_6 as specializations of Θ (Section 4.5).

4.2. CSM regions

For every considered EPLT, the paper interprets the target part of the RDG of the target component of the target closed system as a *region* which individual walks of the component might repeatedly enter,

DOI: 10.1002/stvr

traverse and possibly leave for ever. Formally, a region r of (the RDG of) a system component p is an object characterized by:

1. the set $st(r) \subseteq st(p)$ of its *states*, assumed to be non-empty,
2. the (possibly empty) set $tr(r) \subseteq tr(p)$ of its *transitions*, assumed to comprise only transitions t with $init(t)$ and $fin(t)$ both in $st(r)$,
3. the set $init(r) \subseteq st(r)$ of its *initial states*, i.e., of its *formal entry points*, assumed to comprise at least its *potential entry points*, i.e., every state in $st(r)$ which is either $init(p)$ or the final state of a transition in $tr(p) \setminus tr(r)$, for these are the states in which p might start a traversal of r, and
4. the set $fin(r) \subseteq st(r)$ of its *final states*, i.e., of its *formal exit points*, assumed to comprise at least its *potential exit points*, i.e., every state in $st(r)$ which is either stable (meaning that the system might not allow p to proceed beyond it) or the initial state of a transition in $tr(p) \setminus tr(r)$, for these are the states in which p might end a traversal of r.

The reason why the paper characterizes a region by its formal and not by its potential entry and exit points is that in the conception of EPLTs, it is often convenient and always harmless to treat a state of a region as its entry or exit point even if it is never in this role. In particular, the approach helped to keep Section 4.5 simple.

4.3. Imitations of system component walks

The art of EPLTs operating on regions includes the art of securing that every partial or total traversal of the target region after the modification in every aspect has a substitute which is its adequate *imitation*. This section introduces the three degrees of imitation referred to in the subsequent definition of Θ.

First of all, to imitate a CSM walk w means to imitate its *event sequence*, call it $es(w)$. In the presence of multi-event transitions, the event sequence of a walk is a non-trivial concept. The assumed atomicity of the execution of individual transitions means that when a component p of a closed system within the system executes a walk $w = t_1 \dots t_k$, it for every $1 \leq i \leq k$ before initiating the transition t_i checks that every message supposed to be received during t_i from the other system components is already in place, i.e., virtually receives the messages *before* t_i, as if giving the receptions a higher priority in the execution of $lab(t_i)$. It is, hence, appropriate to say that $es(w)$ is not $lab(w)$, but $lab(t_1)|_{sev^+(p)} \cdot lab(t_1)|_{ev(p) \setminus sev^+(p)} \cdot \ldots \cdot lab(t_k)|_{sev^+(p)} \cdot lab(t_k)|_{ev(p) \setminus sev^+(p)}$.

For two event sequences η_1 and η_2, let $swp(\eta_1, \eta_2)$ denote that η_1 can be obtained from η_2 by just *swapping* pairs of consecutive events without ever swapping two events belonging to the same channel or letting a reception overtake a transmission. The following increasingly lower degrees to which a walk w of a component p of a closed system q might imitate a walk w' of the same or another version of p are of particular interest:

1. The paper says that w *simulates* w' if $init(w) = init(w')$, $swp(es(w)|_{sev(p)}, es(w')|_{sev(p)})$ and $fin(w) = fin(w')$, i.e., if it has the same initial and the same final state (and, hence, also the same cumulative effect on the private channels of p) and executes the same transmissions to and receptions from the other system components, though perhaps with some irrelevant reordering.
2. The paper says that w *implements* w' if $init(w) = init(w')$ and $swp(es(w)|_{sev(p)} \cdot \eta, es(w')|_{sev(p)}| \cdot \eta')$ for a sequence η of events in $sev^+(p)$ and a sequence η' of events in $sev^-(p)$. The relation differs from simulation in that w might differ from w' also in that it has a

DOI: 10.1002/stvr

different final state (and, hence, possibly a different cumulative effect on the private channels of p), executes some additional transmissions to the other system components and/or omits some currently irrelevant receptions from the other system components.

3. The paper says that w *traces* w' if $init(w) = init(w')$ and $(swp(\eta, es(w')|_{sev(p)})) \wedge (\eta' \leq \eta) \wedge (swp(es(w)|_{sev(p)}, \eta' \cdot \eta''))$ for some sequences η and η' of events in $sev(p)$ and a sequence η'' of events in $sev^{-}(p)$. The relation differs from implementation in that w might differ from w' also in that it omits some currently irrelevant transmissions to the other system components.

Note the following:

1. If w simulates w', it also implements it.
2. If w implements w', it also traces it.
3. If w traces w', the virtual partner $vp(p, q)$ of p in q enables execution of w whenever it enables execution of w'.
4. If w implements w', $vp(p, q)$ can through the intervening channels distinguish w from w' only by receiving the messages which w sends in addition.
5. If w simulates w', the two walks are through the intervening channels indistinguishable for $vp(p, q)$.

4.4. The generic transformation and its properties

In Θ, one transforms the current version r_1 of the target region r of the target component p of the target closed system q into its new version r_2, respecting the following specific constraints, in which p_1 and p_2 denote, respectively, the old and the new version of p:

1. *Constraints securing that the initial state of p and the exterior and the boundaries of r remain the same*

   (a) $init(p_1) = init(p_2)$.
   (b) $st(p_1) \setminus st(r_1) = st(p_2) \setminus st(r_2)$.
   (c) $tr(p_1) \setminus tr(r_1) = tr(p_2) \setminus tr(r_2)$.
   (d) $init(r_1) = init(r_2)$.
   (e) $fin(r_1) = fin(r_2)$.

2. *Constraints securing that once p during a system run exits r, the system forgets whether r_1 or r_2 was traversed*

   (a) For every walk w of p_1 with $init(w) \in init(r_1)$, $set(w) \subseteq tr(r_1)$ and $fin(w) \in fin(r_1)$ (i.e., for every total traversal w of r_1), p_2 has a walk which simulates it (i.e., w itself or an alternative for w which is enabled by the virtual partner $vp(p, q)$ of p in q whenever w is, leads p to the same state, has the same cumulative effect on channels and is for $vp(p, q)$ indistinguishable from w).
   (b) For every walk w of p_2 with $init(w) \in init(r_2)$, $set(w) \subseteq tr(r_2)$ and $fin(w) \in fin(r_2)$, p_1 has a walk which simulates it.

DOI: 10.1002/stvr

3. *Constraints securing that when p during a system run reaches an exit point of r, the possibility that it will be able to progress, either by a transition leaving r or by another transition in r, in no way depends on whether the just completed total traversal of r was of r_1 or of r_2*

   (a) For every transition t in $tr(r_1)$ with $init(t) \in fin(r_1)$ (i.e., for every transition starting in an exit point of r_1), $tr(p_2)$ comprises a transition t' with $init(t') = init(t)$ and $lab(t')|_{ev^+(c)} \leq lab(t)|_{ev^+(c)}$ for every channel c in $sch^+(p)$ (i.e. t itself or an alternative for t which is enabled by $vp(p, q)$ whenever t is).
   (b) For every transition t in $tr(r_2)$ with $init(t) \in fin(r_2)$, $tr(p_1)$ comprises a transition t' with $init(t') = init(t)$ and $lab(t')|_{ev^+(c)} \leq lab(t)|_{ev^+(c)}$ for every channel c in $sch^+(p)$.

4. *Constraints securing that when p during a system run reaches an entry point of r, the possibility that it will in the future cause an overfilling of a channel in $pch(q) \setminus pch(p)$ in no way depends on whether it is entering r_1 or r_2*

   (a) For every walk w of p_1 with $init(w) \in init(r_1)$ and $set(w) \subseteq tr(r_1)$ (i.e., for every partial or total traversal w of r_1), p_2 has a walk which implements it (i.e., w itself or an alternative for w which is enabled by $vp(p, q)$ whenever w is and whose potential to cause an overfilling of a channel in $pch(q) \setminus pch(p)$ is at least as high).
   (b) For every walk w of p_2 with $init(w) \in init(r_2)$ and $set(w) \subseteq tr(r_2)$, p_1 has a walk which implements it.

5. *Constraints securing that when p during a system run reaches an entry point of r, the possibility that it will in the future overfill a specific private channel of p in no way depends on whether it is entering r_1 or r_2*

   (a) For every walk w of p_1 with $init(w) \in init(r_1)$ and $set(w) \subseteq tr(r_1)$ (i.e., for every partial or total traversal w of r_1) which overfills a channel c in $pch(p)$, but not until during $last(w)$ (for otherwise c is overfilled already during a proper prefix of w and the fact that w also overfills c is irrelevant), p_2 has a walk which traces w (i.e., w itself or an alternative for w which is enabled by $vp(p, q)$ whenever w is) and overfills c.
   (b) For every walk w of p_2 with $init(w) \in init(r_2)$ and $set(w) \subseteq tr(r_2)$ which overfills a channel c in $pch(p)$, but not until during $last(w)$, p_1 has a walk which traces w and overfills c.

Note that Θ is, unlike Θ_1 to Θ_6, symmetric. As proved in the Appendix, it has also the following properties, in which q_1 and q_2 denote, respectively, the old and the new version of q:

Theorem 1. *If the constraints 1(a) to 1(e), 2(a), 2(b), 3(a) and 3(b) are respected, specific system states are reachable and dead in q_2 if and only if they are in q_1.*

Theorem 2. *If the constraints 1(a) to 1(e), 2(a), 2(b), 4(a) and 4(b) are respected, q_2 can overfill specific channels in $pch(q) \setminus pch(p)$ if and only if q_1 can.*

Theorem 3. *If the constraints 1(a) to 1(e), 2(a), 2(b), 5(a) and 5(b) are respected, q_2 can overfill specific channels in $pch(p)$ if and only if q_1 can.*

DOI: 10.1002/stvr

Theorem 4. *If the constraints 1(a) to 1(e), 2(a), 2(b), 4(a) and 4(b) are respected, specific transitions of the CSMs in $cmp(q) \setminus \{p\}$ are executable in q_2 if and only if they are in q_1.*

Theorem 5. *If the constraints 1(a) to 1(e), 2(a) and 2(b) are respected, specific transitions in $tr(p_1) \setminus tr(r_1)$, i.e. in $tr(p_2) \setminus tr(r_2)$, are executable in q_2 if and only if they are in q_1.*

Theorem 6. *If the constraints 1(a) to 1(e), 2(a) and 2(b) are respected, the following is true: (1) If a specific walk w of p_1 with $init(w) \in init(r_1)$ and $set(w) \subseteq tr(r_1)$ is executable in q_1, every walk of p_2 which traces it is executable in q_2. (2) If a specific walk w of p_2 with $init(w) \in init(r_2)$ and $set(w) \subseteq tr(r_2)$ is executable in q_2, every walk of p_1 which traces it is executable in q_1.*

4.5. The simple transformations as specializations of the generic one

Θ_3 (see the definition in Section 3.4), of which Θ_1 is a specialization, can be interpreted as a specialization of Θ as follows:

1. $tr(r_1) = in(s) \cup out(s)$.
2. $st(r_1) = \cup_{t \in tr(r_1)} \{init(t), fin(t)\}$.
3. $tr(r_2) = \{t_{tt'} | (t \in in(s)) \wedge (t' \in out(s))\}$.
4. $st(r_2) = st(r_1) \setminus \{s\}$.
5. $init(r_1) = init(r_2) = fin(r_1) = fin(r_2) = st(r_2)$.
6. For every transition t in $in(s)$, the walk $w = t$ of p_1 is implemented and traced by every walk $w' = t_{tt'}$ of p_2 with t' a non-blocking transition in $out(s)$, with w' overfilling at least those channels in $pch(p)$ which w does and with $lab(t)|_{ev^+(c)} = lab(t_{tt'})|_{ev^+(c)}$ for every channel c in $sch^+(p)$.
7. For every transition t in $in(s)$ and transition t' in $out(s)$, the walk $t_{tt'}$ of p_2 is simulated, implemented and traced by the walk tt' of p_1, and vice versa, with the two walks overfilling the same channels in $pch(p)$.

Θ_5 (see the definition in Section 3.6), of which Θ_2 is a specialization, can be interpreted as a specialization of Θ as follows:

1. $tr(r_1) = \{t\}$.
2. $tr(r_2) = \emptyset$.
3. $st(r_1) = st(r_2) = init(r_1) = init(r_2) = fin(r_1) = fin(r_2) = \{init(t), fin(t)\}$.
4. The walk t of p_1 is simulated and implemented by the walk w of p_2 from the first assumption, with $lab(first(w))|_{ev^+(c)} \leq lab(t)|_{ev^+(c)}$ for every channel c in $sch^+(p)$.
5. The walk t of p_1 is for every c in $pch(p)$ which it overfills traced by the walk w_c of p_2 from the second assumption, which also overfills c.

Θ_4 (see the definition in Section 3.5) can be interpreted as a specialization of Θ as follows:

1. $tr(r_1) = T \cup out(s)$.
2. $st(r_1) = st(r_2) = init(r_1) = init(r_2) = \cup_{t \in tr(r_1)} \{init(t), fin(t)\}$.
3. $tr(r_2) = out(s) \cup \{t_{tt'} | (t \in T) \wedge (t' \in out(s))\}$.
4. $fin(r_1) = fin(r_2) = st(r_1) \setminus \{s\}$.

DOI: 10.1002/stvr

5. For every transition t in T, the walk $w = t$ of p_1 is implemented and traced by every walk $w' = t_{tt'}$ of p_2 with t' a non-blocking transition in $out(s)$, with w' overfilling at least those channels in $pch(p)$ which w does and with $lab(t)|_{ev^+(c)} = lab(t_{tt'})|_{ev^+(c)}$ for every channel c in $sch^+(p)$.
6. For every transition t in $out(s)$, the walk t of p_1 is simulated, implemented and traced by the walk t of p_2, and vice versa, with the two walks overfilling the same channels in $pch(p)$.
7. For every transition t in T and transition t' in $out(s)$, the walk $t_{tt'}$ of p_2 is simulated, implemented and traced by the walk tt' of p_1, and vice versa, with the two walks overfilling the same channels in $pch(p)$.

Θ_6 (see the definition in Section 3.7) can be interpreted as a specialization of Θ as follows, with t' denoting the new version of the target transition:

1. $tr(r_1) = \{t\}$.
2. $tr(r_2) = \{t'\}$.
3. $st(r_1) = st(r_2) = init(r_1) = init(r_2) = fin(r_1) = fin(r_2) = \{init(t), fin(t)\}$.
4. The walk t of p_1 is simulated and implemented by the walk t' of p_2, and vice versa, with $lab(t)|_{ev^+(c)} = lab(t')|_{ev^+(c)}$ for every channel c in $sch^+(p)$.
5. For every channel c' in $pch(p)$ which both the walk t of p_1 and t' overfill, the walk is traced by the walk t' of p_2 which also overfills c.
6. If the walk t of p_1 overfills c and t' does not, the walk is traced by the walk w from the assumption, which also overfills c.
7. The walk t' of p_2 is for every channel c' in $pch(p)$ which it overfills traced by the walk t of p_1 which also overfills c'.

5. DISCUSSION AND CONCLUSIONS

5.1. The art of advanced reachability analysis as the art of SEPTs

The final recommendation of a recent review and evaluation of techniques for fighting state space explosion in the reachability analysis [11] is to use many different simple techniques and combine them, because individual techniques can seldom bring dramatic improvements. Pelánek [11] classifies techniques into four main groups: (1) reducing the number of the states that need to be explored, (2) reducing the memory requirements needed for storing the explored states, (3) using parallelism or a distributed environment and (4) giving up the requirement on completeness and explore only a part of the reachable part of the system RDG.

This paper, like Duan and Chen [3], takes the view that the four groups correspond, respectively, to the following aspects of the reachability analysis: (1) the reachable part of *which derivative of the target system RDG to explore* (up to the first error encountered), (2) how to make the exploration memory-efficient, (3) how to make the exploration time-efficient, and (4) how much of the reachable part of the target derivative will typically remain unexplored when the memory or time limit is reached. This view conveniently exposes the first aspect as *the only verification-specific concern*. Once a verification expert suggests which derivative to explore, (the assessment of the feasibility of) the exploration of its reachable part can be considered the responsibility of experts for algorithm design and implementation.

DOI: 10.1002/stvr

In the brute-force reachability analysis of a system q, the target derivative of $rdg(q)$ is the RDG of q. In the advanced reachability analysis, the derivative is (a more or less explicit encoding of) some sufficiently informative sub-RDG of $rdg(q)$, i.e., virtually the RDG of some other system, one obtained from q by an SEPT. With this view, the art of advanced reachability analysis reduces to the art of SEPTs, with every new elementary or composite concrete SEPT defining a new method and every new elementary or composite abstract SEPT defining a new class of methods.

The convenience of characterizing a reachability analysis method by its SEPT is in that this reduces its proof to the proof that the transformation is indeed an SEPT. Moreover, if the SEPT has been conceived as a specialization of another, it suffices that the abstract SEPT has been proven. With Θ, the paper demonstrated that there exist SEPTs which are very general and for which the relation between the old and the new system version can be specified with a small set of very intuitive constraints. Such an SEPT is not difficult to prove and is an excellent template for new concrete SEPTs, for which the only concern is then their *efficiency*, i.e., the extent to which for a typical target system q, with its new version a q', $rch(q')$ can be in the target computational environment generated (up to the first error encountered) more easily than $rch(q)$.

5.2. The art of SEPTs as the art of error-preserving (quasi-)local transformations

Pelánek [11] classifies techniques for reducing the number of the states (and transitions) that need to be explored into three groups: (1) reductions relying on (relative) equivalence of states, (2) reductions relying on (relative) equivalence of walks, and (3) compositional methods. In the view of the paper, the first group represents application of Θ_{mes}, whose inclusion into an SEPT is typically the only reason why the SEPT is just semantically error-preserving. By compositional methods, Pelánek means all methods able to exploit the structuring of the target system into components. In the view of the paper, this would be the methods whose SEPT is specified as a composition of SEPTs including instances of Θ_{csm}. A reasonable plan for building non-elementary SEPTs is, hence, to build them from instances of Θ_{mes}, instances of Θ_{csm} and instances of various elementary error-preserving (quasi-)local transformations (EP(Q)LTs), where the assumed role of the latter is to implement reductions relying on (relative) equivalence of walks, the only yet uncovered point in the above classification. The primary interest is, hence, in EP(Q)LTs of precisely this kind.

5.3. Θ and its specializations as EP(Q)LTs exploiting (relative) equivalence of walks

According to Pelánek [11], typical techniques exploiting (relative) equivalence of walks are transition merging [12, 13], POR [5, 6, 14–19], τ-confluence-based reduction (TCR) [20] and SRA [4, 7].

The essence of transition merging is to regard a multi-transition system walk w whose intermediate states are irrelevant for the considered class of errors as a compound transition, thereby potentially completely avoiding further consideration of the states. For every component p of the system, one thereby regards the walk $cw(w, p)$ as a compound transition, a t_p. For the default class of errors, the practice so far has been to assume that $lab(t_p)$ comprises at most one reception on a shared channel of p. Even Duan and Chen [3] never merge transitions of a system component into a multi-reception transition, but Θ does and so do its specializations Θ_3 and Θ_4. With the new transformations, one can, hence, potentially *ignore more states*.

DOI: 10.1002/stvr

The essence of POR is to consider in a system only a representative set of the interleavings in which specific concurrently executable component walks can appear in a system walk, thereby potentially ignoring some system walks. It is, however, impossible to thereby for a rooted walk w of a component p ignore *every* system walk w' with $cw(w', p) = w$. On the other hand, Θ or its specialization Θ_5 might be able to delete from $tr(p)$ a transition in w, so that the system will not even have such walks w'. With the new transformations, one can, hence, potentially *ignore more walks*.

The essence of SRA is to pretend that a set of concurrently executable component walks is virtually executed simultaneously. This is equivalent to assuming that the walks are executed in a specific, though unspecified interleaving, meaning that SRA is just a form of POR.

The essence of TCR is to pretend that an externally unobservable transition of a system has priority over a transition with which it commutes for the considered class of errors, i.e., to ignore every system walk not respecting the priority. Sometimes the priority can be implemented simply as priority in one of the involved system constituents, a participant of both transitions, by application of a transformation removing those walks of the constituent which violate the local priority. Lang and Mateescu [21] recently demonstrated, for a class of systems including CSM systems, that compositional detection and exploitation of commutation is sometimes possible even if each of the two transitions involves multiple system constituents (in a closed CSM system, every transition is externally unobservable and requires cooperation of the component executing its events on some channels and the channels). For a CSM system, with channel specifications assumed to be fixed, constituent-level specification of priority of a system transition over another is possible only if the transitions belong to the same component and, hence, takes the form of an EP(Q)LT. Lang and Mateescu [21], however, consider compositional detection and exploitation only for commutation evident as (strict) strong confluence. For CSM systems, the transformations which they suggest are specializations of Θ operating on regions comprising at most four transitions and no loops. With the general Θ, and even with its specializations Θ_3 to Θ_5, it is possible to *prioritize events for a much more general kind of commutation*. Besides, Lang and Mateescu [21] neglect the fact that some of the system constituents might be channels for which one wants to preserve overfillings.

5.4. Efficiency issues of system pre-processing

Pre-processing of a system q with an SEPT Θ_p before applying a reachability analysis method characterized by an SEPT Θ_r is considered useful if to find the system $q' = \Theta_p(q)$ and then generate (up to the first error encountered) $rch(\Theta_r(q'))$ is (in the worst or the average case) easier than to generate (up to the first error encountered) $rch(\Theta_r(q))$. If $rch(\Theta_r(\Theta_p(q)))$ has less states and not more transitions (or less transitions and not more states) than $rch(\Theta_r(q))$, this is definitely a good sign. With the pre-processing, it should, hence, be possible to ignore more non-erroneous states and/or walks, e.g. by prioritizing more events. As discussed above, a Θ-based Θ_p can potentially achieve that for *any* of the currently popular reachability analysis methods.

As an example, consider again the X.25 system $sys(M, N)$ from Section 3. Its first version $sys(M_1, N_1)$ belongs to the system class for which Gouda and Yu [14] developed an advanced reachability analysis method called *maximal progress state exploration* (MPSE) [14]. The class comprises every closed two-component CFSM system whose only channels are a channel from the first component to the second and a reverse channel. For such systems, MPSE detects every reachable dead state and channel overfilling. Here is an abstractly specified adaptation of the algorithm to systems

DOI: 10.1002/stvr

which, like the later versions of $sys(M, N)$, comprise also multi-event, possibly even multi-reception transitions (for a proof, see the Appendix):

Algorithm 1. *For each of the two components, perform, possibly in parallel, on the system RDG an exploration differing from the brute-force exploration in that the component is in individual generated system states allowed to progress only if (1) the current contents of its outgoing channel can be extended in a way enabling a transition which is in the particular state enabled by the other component, but not by the system, or (2) the other component is in the system state unable to progress.*

Suppose that the channels in $sys(M, N)$ have nominal capacity 6, so that the system can be considered error-free. If the algorithm is applied to $sys(M_1, N_1)$, i.e., to the system $sys(M, N)$ without pre-processing, it generates 110 states and 222 transitions. If it is applied to $sys(M_2, N_2)$, i.e., to $sys(M, N)$ exhaustively pre-processed with Θ_1 and Θ_2, it generates 103 states and 215 transitions. If it is applied to $sys(M_9, N_9)$, i.e., to $sys(M, N)$ exhaustively pre-processed with Θ_1 to Θ_4, it generates 83 (i.e. 20% less) states and 204 (i.e. 5% less) transitions.

Alternatively, suppose that Θ_r does not denote Algorithm 1, but the brute-force reachability analysis. For the non-erroneous $sys(M, N)$, this means that every reachable state and transition is generated. For $sys(M_9, N_9)$, the number of the generated states is again 83 and the number of the generated transitions is again 204, meaning that for $sys(M, N)$, exhaustive pre-processing with Θ_1 to Θ_4 makes the POR technique of Algorithm 1 unnecessary. Remember also that for $sys(M_8, N_8)$, the numbers with brute-force analysis were 90 and 188, meaning that in the last pre-processing step on $sys(M, N)$, states were traded for transitions. One might, hence, want to limit the application of Θ_1 to Θ_4 (or any other specializations of Θ) to forms and combinations not increasing the number of the transitions which individual components have. In this aspect, the two steps transforming $sys(M_6, N_6)$ into $sys(M_8, N_8)$ were unproblematic in combination, but the first of them individually was not.

A wider experiment with Θ_r denoting Algorithm 1 considered 10,000 randomly generated systems with the architecture that assumed by the algorithm. In each of the systems, each of the components had 7 states and only one-event transitions. After generating the sending component transitions randomly, for two kinds of messages, 75% of the applicable receiving component transitions were introduced. For the nominal channel capacity 3, the number of the states and the number of the transitions which Algorithm 1 generates in the worst case were computed. On the average, the numbers were 133 and 180. Exhaustive pre-processing with the previously proposed EPLTs Θ_1 and Θ_2 decreased the numbers, respectively, by 29.9% and 25.0% on the average and by 98.5% and 98.4% in the best case. When the pre-processing included also every possible application of Θ_3 which did not increase the number of the component's transitions and every possible application of Θ_4 which generated at most one transition that could not be immediately eliminated with Θ_1, the respective reductions were 35.0% and 30.1% on the average and 98.7% and 98.7% in the best case. For an average system, the benefit of pre-processing also with Θ_3 and Θ_4 was, hence, relatively low, just 5.1% additional reduction for states and transitions. The experiment, however, detected also systems for which Θ_3 and Θ_4 meant 82% additional reduction for states and 96% additional reduction for transitions.

To be useful, Θ_p must, of course, consist exclusively of easily applicable instances of Θ. A specialization of Θ intended for direct application would be characterized by (1) the pattern which the target region has to match, (2) the additional constraints which the target RDG has to satisfy for the region and (3) the modification performed. The simpler the pattern, the easier it is to identify regions matching it. One may also search for them in many different parts of the target RDG simultaneously.

DOI: 10.1002/stvr

Θ_1 to Θ_6 all have a very simple characterization pattern. The modification specified would typically also be easy to perform. As for the additional constraints, they all require existence of specific kinds of alternative walks, where one is free to decide how short a walk must be to count as a candidate alternative. A lower upper bound means less work, but also less chances for finding an opportunity for applying the EPLT. Alternatively, one could decide to look for the necessary alternative walks only in the target region, i.e., to replace the additional constraints of the EPLT with additional specifics of its characterization pattern. A simple algorithm for systematically performing a selection of specializations of Θ on a CSM would be the following (for a proof, see the Appendix):

Algorithm 2. *For the target CSM p, the target EPLT set Q and variables $Open$ and $Closed$:*

1. $Open := \{init(p)\}; Closed := \emptyset$;
2. *While* $Open \neq \emptyset$*:*
   (a) *Move a state s from $Open$ to $Closed$.*
   (b) *While the RDG of p has a region r with $s \in init(r)$ to which an EPLT Θ' in Q is applicable:*
      i. *While the RDG of p has a region r' with $init(r') \subseteq (Closed \setminus \{s\})$ to which an EPLT in Q is currently not applicable, but will become applicable if Θ' is performed on r, move a member of $init(r')$ from $Closed$ to $Open$.*
      ii. *Apply Θ' to r.*
   (c) *For every transition t in $tr(p)$ with $init(t) = s$ and $fin(t) \notin (Open \cup Closed)$, add $fin(t)$ to $Open$.*

5.5. Concluding remarks

The experiments conducted with Θ_3 and Θ_4 on two-component systems confirmed the findings of Lang and Mateescu [21] that in spite of the relatively low benefit for an average system, pre-processing with specific EP(Q)LTs can be extremely beneficial for systems in which extensive application of the EP(Q)LTs strongly simplifies the component RDGs. Because of the multiplicative nature of CSM composition, for multi-component systems the benefit of such simplifications can be even higher.

Obviously, it is important to have a wide range of easily applicable EP(Q)LTs, so that one can efficiently simplify a wide class of RDGs. So after introducing, for motivation, some new ready-to-use simple EPLTs, the paper defined a generic EPLT from which further EPLTs can be derived simply by specialization. More precisely, it defined very weak constraints sufficient for a region modification to be error-preserving. It also took care that the constraints determining what exactly the transformation preserves were clearly separated from the general ones and grouped according to their purpose.

Some subjects deserving further study are further generalization of the generic EPLT, adaptation of the EPLT for preservation of alternative kinds of errors, identification of further easily applicable and useful specializations of the EPLT, assessment of the benefits which the derived EPLTs would have in typical verification scenarios, and integration of the EPLTs into advanced reachability analysis methods and advanced CSM composition operators.

REFERENCES

DOI: 10.1002/stvr

1. Bochmann G. Finite state description of communication protocols. *Computer Networks* 1978; **2**(4-5):361-372. DOI:10.1016/0376-5075(78)90015-6.
2. Krimm JP, Mounier L. Compositional state space generation with partial order reductions for asynchronous communicating systems. *Proceeding of TACAS 2000* (*Lecture Notes in Computer Science*, vol. 1785). Springer: Berlin, 2000; 266-282. DOI:10.1007/3-540-46419-0_19.
3. Duan L, Chen J. Error-preserving reductions on communication protocols. *Software Testing, Verification and Reliability* 2008; **18**(1):51-68. DOI:10.1002/stvr.385.
4. Ye W, Lei Y, Kung D. A blocking-based approach to protocol validation. *Computer Journal* 2006; **42**(5):541-553. DOI:10.1093/comjnl/bxl017.
5. Godefroid P. *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem* (*Lecture Notes in Computer Science*, vol. 1032). Springer: Berlin, 1996. DOI:10.1007/3-540-60761-7.
6. Clarke EM, Grumberg O, Minea M, Peled D. State space reduction using partial order reduction. *International Journal on Software Tools for Technology Transfer* 1999, **2**(3):279-287. DOI:10.1007/s100090050035.
7. Ozdemir K, Ural H. Protocol validation by simultaneous reachability analysis. *Computer Communications* 1997; **20**(9):772-788. DOI:10.1016/S0140-3664(97)00075-3.
8. Karacali B, Tai K. Model checking based on simultaneous reachability analysis. *Proceedings of the 7th International SPIN Workshop on Model Checking of Software* (*Lecture Notes in Computer Science*, vol.1885), Springer: Berlin, 2000; 34-53. DOI:10.1.1.24.5595.
9. Gouda MG, Yu YT. Synthesis of communicating finite-state machines with guaranteed progress. *IEEE Transactions on Communications* 1984; **32**(7):779-788.
10. Hogrefe D. *OSI Formal Specification Case Study: The InRes Protocol and Service, revised*. Technical Report No. IAM-91-012, University of Bern, Update May 1992.
11. Pelánek R. Fighting state space explosion: Review and evaluation. *Proceedings of FMICS 2008* (*Lecture Notes in Computer Science*, vol. 5596). Springer: Berlin, 2009; 37-52. DOI:10.1007/978-3-642-03240-0_7.
12. Dong Y, Ramakrishnan CR, An optimizing compiler for efficient model checking. *Proceedings of FORTE XII/PSTV XIX*. Kluwer, B.V., 1999, 241-256.
13. Kurshan RP, Levin V, Yenigün H. Compressing transitions for model checking. *Proceedings of CAV 2002* (*Lecture Notes in Computer Science*, vol. 2404). Springer: Berlin, 2002; 569-581. DOI:10.1007/3-540-45657-0_48.
14. Gouda MG, Yu YT. Protocol validation by maximal progress state exploration. *IEEE Transactions on Communications* 1984; **32**(1):94-97.
15. Peled D. Combining partial order reductions with on-the-fly model checking. *Proceedings of CAV 1994* (*Lecture Notes in Computer Science*, vol. 818). Springer: Berlin, 1994; 377-390. DOI:10.1007/3-540-58179-0_69.
16. Holzmann GJ, Peled D. An improvement in formal verification. *Proceedings of FORTE VII*, Chapman&Hall: London, 1995; 197-211.
17. Liu H, Miller RE. Partial-order validation for multi-process protocols modeled as communicating finite state machines. *Proceedings, of ICNP'96*, IEEE Computer Society Press: Washington, 1996; 76-83.
18. Penczek W, Szreter M, Gerth R, Kuiper R. Improving partial order reductions for universal branching time properties. *Fundamenta Informaticae* 2000, **43**(1-4):245-267.
19. Gueta G, Flangan C, Yahav E, Sagiv M. Cartesian partial-order reduction. *Proceedings of SPIN Workshop* (*Lecture Notes in Computer Science*, vol. 4595). Springer: Berlin, 2007; 95-122. DOI:10.1007/978-3-540-73370-6_8.
20. Blom S, van de Pol J. State-space reduction by proving confluence. *Proceedings of CAV 2002* (*Lecture Notes in Computer Science*, vol. 2404). Springer: Berlin, 2002; 596-609. DOI:10.1007/3-540-45657-0_50.
21. Lang F., Mateescu R. Partial order reductions using compositional confluence detection. *Proceedings of FM 2009* (*Lecture Notes in Computer Science*, vol. 5850). Springer: Berlin, 2009; 157-172. DOI:10.1007/978-3-642-05089-3_11.

Appendix: Proofs

Lemma 1. *If q_1 has a rooted walk w and p_2 has a rooted walk w' implementing $cw(w,p)$, q_2 has a rooted walk w'' with $cw(w'',p) = w'$ and $cw(w'',p') = cw(w,p')$ for every CSM p' in $cmp(q) \setminus \{p\}$.*

Proof: If p instead of $cw(w,p)$ tries to execute w', the changes in its external behaviour are that it perhaps transmits some additional messages, executes some transmissions belonging to different channels in a different order, receives less of the incoming messages and/or delays some receptions. Only the first change might be observable for the virtual partner $vp(p,q)$ through the intervening

channels, but only if it enables additional receptions. Hence, if its walk remains the same, it can still execute it to completion, thereby providing a context in which p can complete w'. □

Lemma 2. *If the constraints 1(a) to 1(e), 2(a) and 2(b) are respected, q_2 for every rooted walk w of q_1 with $(cw(w,p) = \varepsilon) \vee (last(cw(w,p)) \notin tr(r_1)) \vee (fin(cw(w,p)) \in fin(r_1))$ has a rooted walk w' with $cw(w',p') = cw(w,p')$ for every CSM p' in $cmp(q) \setminus \{p\}$ and with $cw(w',p)$ simulating $cw(w,p)$.*

Proof: For such a w, $cw(w,p)$ can be represented as a $w_1 \cdot w'_1 \cdot w_2 \cdot \ldots \cdot w'_{k-1} \cdot w_k$ with $set(w_i) \subseteq tr(p_1) \setminus tr(r_1)$ for $1 \leq i \leq k$ and with $init(w'_i) \in init(r_1)$, $set(w'_i) \subseteq tr(r_1)$ and $fin(w'_i) \in fin(r_1)$ for $1 \leq i \leq k-1$. p_2 has a walk $w'' = w_1 \cdot w''_1 \cdot w_2 \cdot \ldots \cdot w''_{k-1} \cdot w_k$ in which for every $1 \leq i \leq k-1$, w''_i simulates w'_i. w'' simulates and, hence, implements $cw(w,p)$. This, by Lemma 1, implies that q_2 has a rooted walk w' with $cw(w',p) = w''$ and $cw(w',p') = cw(w,p')$ for every CSM p' in $cmp(q) \setminus \{p\}$. □

Proof of Theorem 1. As the transformation is symmetric, it suffices to prove the "if" part.

If a state s is reachable and dead in q_1, $s|_p \in (st(p_1) \setminus st(r_1)) \cup fin(r_1)$, because every state in $st(r_1) \setminus fin(r_1)$ is unstable, and q_1 has a rooted walk w with $fin(w) = s$. If $cw(w,p) \neq \varepsilon$, then $last(cw(w,p)) \notin tr(r_1)$ or $fin(cw(w,p)) \in fin(r_1)$. Hence, $s|_p \in (st(p_2) \setminus st(r_2)) \cup fin(r_2)$ and, by Lemma 2, q_2 has a rooted walk w' with $fin(w') = s$.

Suppose that in q_2, s is not dead. As it is dead in q_1, it must be that $tr(p_2) \setminus tr(p_1)$, and hence $tr(r_2)$, comprises a transition t with $init(t) = s|_p$ for which $tr(p_1)$ comprises no transition t' with $init(t') = init(t)$ and $lab(t')|_{ev^+(c)} \leq lab(t)|_{ev^+(c)}$ for every channel c in $sch^+(p)$. This is, however, impossible, as $init(t)$, i.e. $s|_p$, is in $st(r_2)$ and, hence, in $fin(r_2)$. □

Proof of Theorem 5. As the transformation is symmetric, it suffices to prove the "if" part.

If a specific transition in $tr(p_1) \setminus tr(r_1)$ is executable in q_1, q_1 has a rooted walk w with t executable in $fin(w)$. If $cw(w,p) \neq \varepsilon$, then $last(cw(w,p)) \notin tr(r_1)$ or $fin(cw(w,p)) \in fin(r_1)$. Hence, by Lemma 2, q_2 has a rooted walk w' with $fin(w') = fin(w)$, meaning that it can reach $fin(w)$ and then, by t in $tr(p_2)$, execute t. □

Lemma 3. *If the constraints 1(a) to 1(e), 2(a), 2(b), 4(a) and 4(b) are respected, q_2 for every rooted walk w of q_1 has a rooted walk w' with $cw(w',p') = cw(w,p')$ for every CSM p' in $cmp(q) \setminus \{p\}$ and with $cw(w',p)$ implementing $cw(w,p)$.*

Proof: For such a w, $cw(w,p)$ can be represented as a $w_1 \cdot w'_1 \cdot w_2 \cdot \ldots \cdot w_k \cdot w'_k$ with $set(w_i) \subseteq tr(p_1) \setminus tr(r_1)$ and $set(w'_i) \subseteq tr(r_1)$ for $1 \leq i \leq k$, with $init(w'_i) \in init(r_1)$ and $fin(w'_i) \in fin(r_1)$ for $1 \leq i \leq k-1$, and with $w'_k = \emptyset$ or $init(w'_k) \in init(r_1)$. p_2 has a walk $w'' = w_1 \cdot w''_1 \cdot w_2 \cdot \ldots \cdot w_k \cdot w''_k$ with w''_i simulating w'_i for $1 \leq i \leq k-1$ and with w''_k implementing w'_k. w'' implements $cw(w,p)$. This, by Lemma 1, implies that q_2 has a rooted walk w' with $cw(w',p) = w''$ and $cw(w',p') = cw(w,p')$ for every CSM p' in $cmp(q) \setminus \{p\}$. □

Proof of Theorem 4. As the transformation is symmetric, it suffices to prove the "if" part.

If a specific transition t of a CSM p' in $cmp(q) \setminus \{p\}$ is executable in q_1, q_1 has a rooted walk w with t executable in $fin(w)$. Then, by Lemma 3, q_2 has a rooted walk w' with $cw(w',p'') = cw(w,p'')$ for every CSM p'' in $cmp(q) \setminus \{p\}$ and with $cw(w',p)$ implementing $cw(w,p)$, i.e., with $cw(w',p') = cw(w,p')$ and $es(cw(w,csm^-(c)))|_{ev^-(c)} \leq es(cw(w',csm^-(c)))|_{ev^-(c)}$ for every

DOI: 10.1002/stvr

channel c in $sch^+(p')$. Hence, $fin(w')|_{p'} = fin(w)|_{p'}$ and $fin(w)|_c \leq fin(w')|_c$ for every channel c in $sch^+(p')$, meaning that $fin(w')$ is a reachable state of q_2 in which t is executable. □

Proof of Theorem 2. As the transformation is symmetric, it suffices to prove the "if" part.

If q_1 can overfill a specific channel c in $pch(p')$ of a CSMs p' in $cmp(q) \setminus \{p\}$, p' has a transition t which overfills c and is executable in q_1. By Theorem 4, t is executable also in q_2.

If q_1 can overfill a specific channel c in $pch(q) \setminus \cup_{p' \in cmp(q)} pch(p')$, it has a rooted walk w with $fin(w)$ a state in which c is overfilled. Then, by Lemma 3, q_2 has a rooted walk w' with $cw(w', p') = cw(w, p')$ for every CSM p' in $cmp(q) \setminus \{p\}$ and with $cw(w', p)$ implementing $cw(w, p)$, i.e., a walk in which the CSM $csm^-(c)$ executes at least as many transmissions on c as in w and the CSM $csm^+(c)$ executes at most as many receptions on c as in w. $fin(w')|_c$ is, hence, at least as long as $fin(w)|_c$. □

Lemma 4. *If q_1 has a rooted walk w and p_2 has a rooted walk w' tracing $cw(w, p)$, q_2 has a rooted walk w'' with $cw(w'', p) = w'$.*

Proof: If w' traces $cw(w, p)$, then w' is rooted and $swp(\eta, es(cw(w, p))|_{sev(p)})$, $\eta' \leq \eta$ and $swp(es(w')|_{sev(p)}, \eta' \cdot \eta'')$ for some sequences η and η' of events in $sev(p)$ and a sequence η'' of events in $sev^-(p)$. If p_2 had a rooted walk w_1' with $es(w_1')|_{sev(p)} = \eta$, q_2 would, by w_1' implementing $cw(w, p)$ and by Lemma 1, have a rooted walk w_1'' with $cw(w_1'', p) = w_1'$. Hence, if p_2 had a rooted walk w_2' with $es(w_2')|_{sev(p)} = \eta'$, q_2 would, by $\eta' \leq \eta$, have a rooted walk w_2'' with $cw(w_2'', p) = w_2'$. Hence, if p_2 had a rooted walk w_3' with $es(w_3')|_{sev(p)} = \eta' \cdot \eta''$, q_2 would, by $set(\eta'') \subseteq sev^-(p)$, have a rooted walk w_3'' with $cw(w_3'', p) = w_3'$. Hence, if p_2 has the rooted walk w', q_2 has, by w' implementing w_3' and by Lemma 1, a rooted walk w'' with $cw(w'', p) = w'$. □

Proof of Theorem 6. As the transformation is symmetric, it suffices to prove (1).

If such a w is executable in q_1, q_1 has a rooted walk w^1 with $cw(w^1, p)$ a $w_1 \cdot w_1' \cdot w_2 \cdot \ldots \cdot w_{k-1}' \cdot w_k \cdot w$ with $set(w_i) \subseteq tr(p_1) \setminus tr(r_1)$ for $1 \leq i \leq k$ and with $init(w_i') \in init(r_1)$, $set(w_i') \subseteq tr(r_1)$ and $fin(w_i') \in fin(r_1)$ for $1 \leq i \leq k-1$. For every w' tracing w, p_2 has a walk $w'' = w_1 \cdot w_1'' \cdot w_2 \cdot \ldots \cdot w_{k-1}'' \cdot w_k \cdot w'$ with w_i'' simulating w_i' for $1 \leq i \leq k-1$. w'' traces $cw(w^1, p)$. This, by Lemma 4, implies that q_2 has a rooted walk w^2 with $cw(w^2, p) = w''$, meaning that the suffix w' of w'' is executable in q_2. □

Proof of Theorem 3. As the transformation is symmetric, it suffices to prove the "if" part.

If q_1 can overfill a specific channel c in $pch(p)$, it has a rooted walk w in which $cw(w, p)$ overfills c, but not until during the transition $t = last(cw(w, p))$. If $t \in tr(p_1) \setminus tr(r_1)$, t is, by Theorem 5, executable also in q_2, meaning that q_2 can overfill c. If $t \in tr(r_1)$, $cw(w, p)$ is a $w_1 \cdot w_2$ with $init(w_2) \in init(r_1)$ and $set(w_2) \subseteq tr(r_1)$. Hence, p_2 has a walk w_2' which traces w_2 and overfills c. By Theorem 6, w_2' is executable in q_2, meaning that q_2 can overfill c. □

Proof of Algorithm 1. Any generated state is a reachable state of the system and any generated transition is a system transition from such a state. The exploration, hence, can neither detect a channel overfilling which the system does not have nor misinterpret a dead state as a progress state.

For any progress state generated, at least one transition emanating from the state is also generated, meaning that no progress state is misinterpreted as a dead state.

If the system can reach, by a walk w, a dead state s, that part of the exploration in which p progresses normally generates a walk w' with $cp(w', p) = cp(w, p)$ and $cp(w', p') \leq cp(w, p')$. If

DOI: 10.1002/stvr

$cp(w', p') \neq cp(w, p')$, $fin(w')$ is a progress state of the system. As p cannot progress in the state, the exploration allows p' to complete the execution of $cp(w)$. This extends w' into a walk w'' with $fin(w'') = fin(w)$, meaning that the exploration generates every reachable dead state of the system.

If the system can reach, by a walk w, a state s in which the channel, a $c_{p,p'}$, from a component p to the other component, a p', is overfilled, that part of the exploration in which p progresses normally generates a walk w' with $cp(w', p) = cp(w, p)$ and $cp(w', p') \leq cp(w, p')$. $fin(w')|_{c_{p,p'}}$ is, hence, an $\eta \cdot s|_{c_{p,p'}}$, meaning that the overfilling in s is detected as overfilling of $c_{p,p'}$ in $fin(w')$ or an earlier state. □

Proof of Algorithm 2. The algorithm basically visits every state in $st(p)$ and performs the required EPLTs on the regions emanating from the state. It might, however, happen that an EPLT in Q becomes applicable to a region r' after every state in $init(r')$ has already been considered and consequently moved to $Closed$. The problem is solved by moving a member of $init(r')$ back to $Open$, thereby securing reconsideration of r'. □

DOI: 10.1002/stvr