Basic Research in Computer Science

Compositional Safety Logics

Jørgen H. Andersen
Kim G. Larsen

BRICS Report Series **RS-97-13**

ISSN 0909-0878 **June 1997**

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/97/13/`

COMPOSITIONAL SAFETY LOGICS

J.H. Andersen and K.G. Larsen
BRICS, Aalborg University, Denmark

June 17, 1997

In this paper we present a generalisation of a promising compositional model–checking technique introduced for finite–state systems by Andersen in [And95] and extended to networks of timed automata by Larsen et al in [LPY95a, LL95, LPY95b, KLL+97a]. In our generalized setting, programs are modelled as arbitrary (possibly infinite–state) transition systems and verified with respect to properties of a basic safety logic. As the fundamental prerequisite of the compositional technique, it is shown how logical properties of a parallel program may be transformed into necessary and sufficient properties of components of the program. Finally, a set of axiomatic laws are provided useful for simplifying formulae and complete with respect to validity and unsatisfiability.

1 Introduction

It is well–known that the major problem in applying automatic verification techniques to analyze concurrent systems is the potential combinatorial explosion of the state space arising from parallel composition. During the last decade, various techniques have been developed to avoid this explosion problem, either by *symbolic* representation of the state space using BDD [BCM+90, GW91], by application of *partial order* methods [GW91, Val90] which suppresses unnecessary interleavings of transitions, or by application of *abstractions* and *symmetries* [BCM+90, CFJ93, CGL92].

A recently introduced [And95] and very promising technique is a *compositional* technique, which avoids global state–space construction and – exploration by gradually moving components of a concurrent system from

the system description into the specification. Consider the following verification problem

$$(P_1 \mid \ldots \mid P_n) \models \varphi \tag{1.1}$$

where $(P_1 \mid \ldots \mid P_n)$ is the concurrent system to be verified and φ is a formula specifying a desired property. The compositional technique allows a component (P_n say) to be removed from the network and instead added to the specification φ. This results in a *quotient* formula φ/P_n expressing the sufficient and necessary property of $(P_1 \mid \ldots \mid P_{n-1})$ in order that (1.1) holds. That is, the original verification problem reduces to

$$(P_1 \mid \ldots \mid P_{n-1}) \models \varphi/P_n \tag{1.2}$$

Now repeating this process of quotienting iteratively yields equivalent verification problems with decreasing numbers of components and hence decreasing state–space sizes. However, this idea alone is clearly not enough to solve the explosion–problem as the explosion may now show up as an exponential growth in the sizes of the quotient formulas instead. The crucial observation in [And95] is that each quotienting should be followed by a *minimization* of the formula based on a small collection of efficiently implementable strategies.

The three parameters to the compositional method are the following: 1) the components P_i and their semantic modelling; 2) the notion of parallel composition, and 3) the logic for specifying properties. In order for the compositional method to be applicable the combination of the three parameters must obviously satisfy the following two criteria:

1. The logic must be expressive enough that the quotient formulas can be expressed.

2. There must be heuristics for simplifying formulas of the logic.

In [And95] the compositional method is developed for a setting of finite–state systems with parallel composition being that of CCS [Mil89]. As specification formalism is used the modal μ-calculus [Koz82]. Using a prototype implementation experimental evidence is given that the technique may improve results obtained using BDDs.

In [LPY95a, LL95, LPY95b] the compositional method has been extended to deal with real–time systems modelled as networks of timed automata [AD90, AD92], with a real–time version of the modal μ–calculus [LLW95]

used as specification formalism. In [KLL+97b] the real–time extension of the technique is applied to the verification of a mutual exclusion protocol; experimental results obtained using a tool implementation of the method gives further evidence of the potential of technique.

In this paper we show how to satisfy the basic requirements of the compositional method in a *generalized setting*, where the components are modelled as arbitrary (possibly infinite–state) transition systems The notion of parallel composition is that of interleaving, and as specification formalism is considered a minimal safety logic.

To meet the two criteria of the compositional method, the safety logic is extended in a minimal way to be closed with respect to quotienting. Possible strategies for simplifying formulas are suggested by a set of axiomatic laws for formula equivalence. The laws are shown complete with respect to validity and unsatisfiability of formulas.

The paper is organized as follows: In the next section we describe our general setting for modelling parallel programs. For this setting we introduce in Section 3 a minimal modal for expressing safety properties. A necessary extension of this logic allowing for quotienting is then provided in Section 4. In Section 5 we present an axiomatization allowing formulae to be simplified and being complete with respect to validity and unsatisfiability. Finally, we apply the quotient construction and the provided axiomatization to the verification of a (simple) shared variable program in Section 6.

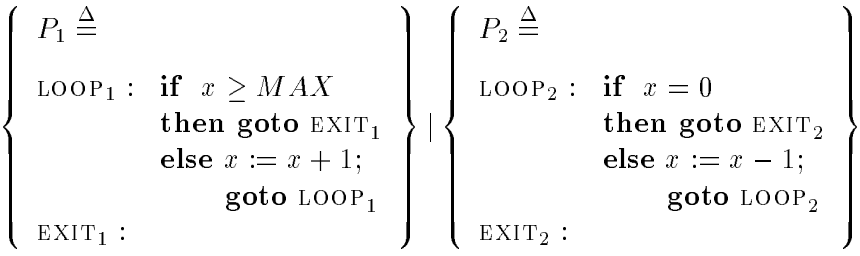

Figure 1: A parallel counter.

2 Programs and Transition Systems

Component programs as well as their composition are modelled as arbitrary transition systems.

Definition 1 (Transition system) *A transition system is a tuple $\langle \mathcal{S}, s_0, \Sigma \rangle$ where: $\mathcal{S}$ is a set of states, $s_0 \in \mathcal{S}$ is a start state and $\Sigma \subset \mathcal{S} \times \mathcal{S}$ is a transition relation.* $\diamond$

Parallel composition is equally general: given two transition systems $\mathcal{T}_1 = \langle \mathcal{S}, s_0, \Sigma_1 \rangle$ and $\mathcal{T}_2 = \langle \mathcal{S}, s_0, \Sigma_2 \rangle$ over the same set of states and with the same start state we define the composition of $\mathcal{T}_1$ and $\mathcal{T}_2$ as the transition system $\mathcal{T} = \mathcal{T}_1 \,|\, \mathcal{T}_2 = \langle \mathcal{S}, s_0, \Sigma_1 \cup \Sigma_2 \rangle$. That is we model parallel composition as pure interleaving.

In typical applications, the state–space $\mathcal{S}$ of a composition of programs $P = P_1 \,|\, \cdots \,|\, P_n$ over a shared data–domain D will be described as the product of the local state-spaces $\mathcal{S}_i$ of the individual programs and D, that is $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n \times D$. To ensure that $\mathcal{S}_i$ acts as a local state–space for the component P_i, one may impose the following constraint on the transition relation Σ_i modelling the behaviour of P_i: whenever $(\langle s_1, \ldots, s_n, d \rangle, \langle s'_1, \ldots, s'_n, d' \rangle) \in \Sigma_i$ then $s_j = s'_j$ for all $j \neq i$. Thus, transitions of P_i may depend on the global state but can only effect the local state of P_i and the shared data.

Example 1 [Parallel counter] Consider the program of Figure 1. Both component programs P_1 and P_2 have two local control–states. In the LOOP state P_1 repeatedly increments the value of the shared variable x until it reaches the value MAX, at which point P_1 goes to the terminating EXIT

control state. Dually, in the LOOP control state, P_2 repeatedly decrements x unless it has reached the value 0, at which point P_2 goes to the terminating EXIT control state. In the interleaved parallel composition of P_1 and P_2, the value of x will in most states be incremented or decremented non–deterministically. Thus, the termination properties of the combined program is less obvious than those of the component programs. Formally, we describe P_1 and P_2 in terms of transition systems:

$$\mathcal{T}_i = \langle \mathcal{S}_1 \times \mathcal{S}_2 \times \mathbb{N}, \langle \text{LOOP}_1, \text{LOOP}_2, 0 \rangle, \Sigma_i \rangle, \quad i \in \{1, 2\}$$

where $\mathcal{S}_i = \{\text{LOOP}_i, \text{EXIT}_i\}$ and:

$$\begin{aligned}
\Sigma_1 \;=\; & \Big\{ (\langle \text{LOOP}_1, l_2, n \rangle, \langle \text{LOOP}_1, l_2, n+1 \rangle) \mid n < MAX \text{ and } l_2 \in \mathcal{S}_2 \Big\} \cup \\
& \Big\{ (\langle \text{LOOP}_1, l_2, n \rangle, \langle \text{EXIT}_1, l_2, n \rangle) \mid n \geq MAX \text{ and } l_2 \in \mathcal{S}_2 \Big\} \\
\Sigma_2 \;=\; & \Big\{ (\langle l_1, \text{LOOP}_2, n \rangle, \langle l_1, \text{LOOP}_2, n-1 \rangle) \mid n > 0 \text{ and } l_1 \in \mathcal{S}_1 \Big\} \cup \\
& \Big\{ (\langle l_1, \text{LOOP}_2, n \rangle, \langle l_1, \text{EXIT}_2, n \rangle) \mid n \leq 0 \text{ and } l_1 \in \mathcal{S}_1 \Big\}
\end{aligned}$$

The composition of P_1 and P_2 is then described by the transition system:

$$\mathcal{T} = \langle \mathcal{S}_1 \times \mathcal{S}_2 \times \mathbb{N}, \langle \text{LOOP}_1, \text{LOOP}_2, 0 \rangle, \Sigma_1 \cup \Sigma_2 \rangle$$

$\diamond$

3 Safety Logic

We first introduce a minimal modal logic, $\mathcal{L}_{\mathcal{S}}^{\mathsf{safe}}$, for expressing safety properties of programs over a given set of states $\mathcal{S}$. Formulae in $\mathcal{L}_{\mathcal{S}}^{\mathsf{safe}}$ are given by the following BNF:

$$\varphi ::= B \;\Big|\; \varphi \wedge \varphi \;\Big|\; \Box\varphi \;\Big|\; X$$

where $B \subseteq \mathcal{S}$ is a condition over a set of states and X is a formula identifier of a finite set of identifiers Id. X is declared by some declaration $\mathcal{D} \in Id \to \mathcal{L}_{\mathcal{S}}^{\mathsf{safe}}$. Note that $\mathcal{L}_{\mathcal{S}}^{\mathsf{safe}}$ is parameterized with the set of states over which conditions are expressed.

A formula φ of $\mathcal{L}_{\mathcal{S}}^{\mathsf{safe}}$ is interpreted with respect to a declaration and a transition relation in the following obvious sense:

Definition 2 *Let $\mathcal{S}$ be a set of states and $\Sigma \subseteq \mathcal{S} \times \mathcal{S}$ a transition relation over $\mathcal{S}$. Let $\mathcal{D} \in Id \to \mathcal{L}_{\mathcal{S}}^{\mathsf{safe}}$ be a declaration. The satisfaction relation $\models_{\mathcal{D},\Sigma}$ is the maximal relation satisfying the following set of implications:*

$$\begin{array}{lcl} s \models_{\mathcal{D},\Sigma} \varphi \wedge \psi & \Rightarrow & s \models_{\mathcal{D},\Sigma} \varphi \text{ and } s \models_{\mathcal{D},\Sigma} \psi \\ s \models_{\mathcal{D},\Sigma} \Box\varphi & \Rightarrow & \text{whenever } (s,s') \in \Sigma \text{ then } s' \models_{\mathcal{D},\Sigma} \varphi \\ s \models_{\mathcal{D},\Sigma} B & \Rightarrow & s \in B \\ s \models_{\mathcal{D},\Sigma} X & \Rightarrow & s \models_{\mathcal{D},\Sigma} \mathcal{D}(X) \end{array}$$

$\diamond$

Any relation satisfying the above implications is a *satisfiability* relation. It follows from standard fixpoint theory [Tar55] that $\models_{\mathcal{D},\Sigma}$ is the union of all satisfiability relations and that the above implications are biimplications for $\models_{\mathcal{D},\Sigma}$.

Example 2 Continuing Example 1, we may wonder whether there are executions of $P_1 \,|\, P_2$ in which P_2 will terminate. Let $\neg\text{EXIT}_2$ denote the set of all states $\langle l_1, l_2, n\rangle$ where $l_2 \neq \text{EXIT}_2$. Then the property that no execution will bring P_2 to termination may be expressed as follows [1]

$$X \stackrel{\Delta}{=} \neg\text{EXIT}_2 \ \wedge \ \Box X$$

$\diamond$

4 Quotienting and $\mathcal{L}_{\mathcal{S}}$

Now let P_1 and P_2 be component programs modelled as transition relations Σ_1 and Σ_2 over some common global state space $\mathcal{S}$. Verification problems for $P_1 \,|\, P_2$ then becomes assertions of the form $s \models_{\mathcal{D},\Sigma_1\cup\Sigma_2} \varphi$.

To meet the requirements of the compositional method, we are required to describe a technique for moving parts of $\Sigma_1 \cup \Sigma_2$ (say Σ_2 which corresponds to the component P_2) into φ. More precisely, we offer a construction for *quotienting* a formula φ with respect to a transition relation Σ_2 such that:

$$s \models_{\mathcal{D},\Sigma_1\cup\Sigma_2} \varphi \ \text{ iff } \ s \models_{\mathcal{D},\Sigma_1} \varphi/_{\Sigma_2}$$

[1] Clearly $P_1 \,|\, P_2$ does *not* satisfy this property and later (in Section 6) we shall provide a proof of this fact using the compositional proof technique.

In order to express quotient formulae we extend $\mathcal{L}_{\mathcal{S}}^{\mathsf{safe}}$ with a new construction $\Delta \twoheadrightarrow \varphi$, where $\Delta \subseteq \mathcal{S} \times \mathcal{S}$ (the so–called *step–operator*). Formulae of the resulting extended logic, simply called $\mathcal{L}_{\mathcal{S}}$, is given by the following BNF:

$$\varphi ::= B \;\Big|\; \varphi \wedge \varphi \;\Big|\; \Box\varphi \;\Big|\; X \;\Big|\; \Delta \twoheadrightarrow \varphi$$

where $\Delta \subseteq \mathcal{S} \times \mathcal{S}$. The interpretation of $\mathcal{L}_{\mathcal{S}}$ extends that of $\mathcal{L}_{\mathcal{S}}^{\mathsf{safe}}$ (Definition 2) with the following implication:

$$s \models_{\mathcal{D},\Sigma} \Delta \twoheadrightarrow \varphi \quad \Rightarrow \quad \text{whenever } (s,s') \in \Delta \text{ then } s' \models_{\mathcal{D},\Sigma} \varphi$$

To see how $\Delta \twoheadrightarrow \varphi$ is used in quotienting consider a simple assertion of the form

$$s \models_{\mathcal{D},\Sigma_1 \cup \Sigma_2} \Box B \tag{4.1}$$

Now clearly (4.1) holds if $s' \in B$ whenever $(s,s') \in \Sigma_1$ and $s'' \in B$ whenever $(s,s'') \in \Sigma_2$. More concisely, this becomes $s \models_{\mathcal{D},\Sigma_1} \Box B$ and $s \models_{\mathcal{D},\Sigma_1} \Sigma_2 \twoheadrightarrow B$. Thus the quotient formula ${}^{\Box B}/_{\Sigma_2}$ is precisely $\Box B \wedge \Sigma_2 \twoheadrightarrow B$.

Generally, the quotient–construction is given by the following definition:

Definition 3 (Quotienting) *Let ϕ be an $\mathcal{L}_{\mathcal{S}}$-formula and let $\Sigma \subseteq \mathcal{S} \times \mathcal{S}$ be a transition relation. Then the quotient formula $({}^{\phi}/_{\Sigma})$ is defined inductively over ϕ as follows*

$$\begin{aligned}
{}^{\varphi \wedge \psi}/_{\Sigma} &= {}^{\varphi}/_{\Sigma} \wedge {}^{\psi}/_{\Sigma} \\
{}^{\Box\varphi}/_{\Sigma} &= \Box\, {}^{\varphi}/_{\Sigma} \wedge \Sigma \twoheadrightarrow {}^{\varphi}/_{\Sigma} \\
{}^{\Delta \twoheadrightarrow \varphi}/_{\Sigma} &= \Delta \twoheadrightarrow {}^{\varphi}/_{\Sigma} \\
{}^{B}/_{\Sigma} &= B \\
{}^{X}/_{\Sigma} &= X^{\Sigma}
\end{aligned}$$

Where X^{Σ} is a new identifier. For $\mathcal{D}$ a declaration over Id and Σ a transition relation, $\mathcal{D}^{\Sigma}$ is the declaration over $Id^{\Sigma} = \{X^{\Sigma} \mid X \in Id\}$ given by $\mathcal{D}^{\Sigma} = \{X^{\Sigma} \triangleq {}^{\varphi}/_{\Sigma} \mid X \triangleq \varphi \in \mathcal{D}\}$. ◇

Theorem 1 (Correctness of Quotienting) *Let Σ_1 and Σ_2 be transition relations, and let $\mathcal{D}$ be a declaration. Then*

$$s \models_{\mathcal{D},\Sigma_1 \cup \Sigma_2} \varphi \quad \Leftrightarrow \quad s \models_{\mathcal{D}^{\Sigma_2},\Sigma_1} {}^{\varphi}/_{\Sigma_2}$$

Proof: See Appendix. ◇

5 Axiomatization

Having introduced the logic $\mathcal{L}_{\mathcal{S}}$ the question of axiomatizing equivalence between and validity of formulae naturally arises. Besides providing fundamental theoretical insight these axiomatizations are crucial in simplifying quotient formulae and hence making the compositional approach to model checking feasible.

By $\models_{\mathcal{D},\Sigma} \varphi \simeq \psi$ we denote that φ and ψ are *equivalent* with respect to the declaration $\mathcal{D}$ and the transition relation Σ in the obvious sense, that is:

$$\models_{\mathcal{D},\Sigma} \varphi \simeq \psi \quad \text{iff} \quad \forall s \in \mathcal{S}.s \models_{\mathcal{D},\Sigma} \varphi \Leftrightarrow s \models_{\mathcal{D},\Sigma} \psi$$

By $\models \varphi \simeq \psi$ we denote that $\models_{\mathcal{D},\Sigma} \varphi \simeq \psi$ holds for all $\mathcal{D}$ and Σ, and by we denote that $\models_{\Sigma} \varphi \simeq \psi$ holds for all $\mathcal{D}$.

A formula φ is *valid* whenever $\models \varphi \simeq \mathcal{S}$ and we denote this by $\models \varphi$. A formula is *unsatisfiable* if $\models \varphi \simeq \emptyset$ which we denote[2] $\models \neg\varphi$. Now — due to the single universal modality $\Box$ of $\mathcal{L}_{\mathcal{S}}$ — it is not hard to see that validity (unsatisfiability) of formulae reduces to validity (unsatisfiability) with respect to the total (empty) transition relation, that is:

Lemma 1 *Let φ be a formula of $\mathcal{L}_{\mathcal{S}}$. Then $\models \varphi$ if and only if $\models_{S\times S} \varphi \simeq \mathcal{S}$. Moreover $\models \neg\varphi$ if and only if $\models_{\emptyset} \varphi \simeq \emptyset$.* $\diamond$

Note that for $\Box$-free formulae φ and ψ equivalence is independent of the interpreting transition relation. That is for any Σ:

$$\models_{\Sigma} \varphi \simeq \psi \text{ if and only if } \models \varphi \simeq \psi$$

Let φ be a recursion-free formula. Then by replacing all occurrences of subformulae of the form $\Box\eta$ with $\Sigma \twoheadrightarrow \eta$ we obtain a Σ-equivalent formula φ^{Σ}, i.e. $\models_{\Sigma} \varphi \simeq \varphi^{\Sigma}$. It follows that:

$$\models_{\Sigma} \varphi \simeq \psi \text{ if and only if } \models \varphi^{\Sigma} \simeq \psi^{\Sigma}$$

Applied to the special case of validity and unsatisfiability this yields:

Lemma 2 *Let φ be a recursion-free formula of $\mathcal{L}_{\mathcal{S}}$. Then $\models \varphi$ if and only if $\models \varphi^{\mathcal{S}\times\mathcal{S}} \simeq \mathcal{S}$. Moreover $\models \neg\varphi$ if and only if $\models \varphi^{\emptyset} \simeq \emptyset$* $\diamond$

[2]The notation $\neg\varphi$ does not imply that $\neg\varphi$ is added to the logic as a construct

$$\vdash B_1 \wedge B_2 \;\simeq\; B_1 \cap B_2 \tag{5.1}$$

$$\vdash \emptyset \twoheadrightarrow \varphi \;\simeq\; \mathcal{S} \tag{5.2}$$

$$\vdash \emptyset \wedge \varphi \;\simeq\; \emptyset \tag{5.3}$$

$$\vdash \mathcal{S} \wedge \varphi \;\simeq\; \varphi \tag{5.4}$$

$$\vdash \Box\varphi \wedge \Box\psi \;\simeq\; \Box(\varphi \wedge \psi) \tag{5.5}$$

$$\vdash \Delta \twoheadrightarrow \varphi \wedge \nabla \twoheadrightarrow \varphi \;\simeq\; (\Delta \cup \nabla) \twoheadrightarrow \varphi \tag{5.6}$$

$$\vdash \Delta \twoheadrightarrow \nabla \twoheadrightarrow \varphi \;\simeq\; \Delta \circ \nabla \twoheadrightarrow \varphi \tag{5.7}$$

$$\vdash \Delta \twoheadrightarrow \varphi \wedge \Delta \twoheadrightarrow \psi \;\simeq\; \Delta \twoheadrightarrow (\varphi \wedge \psi) \tag{5.8}$$

$$\vdash \Delta \twoheadrightarrow B \;\simeq\; \Delta^{-1}(B) \tag{5.9}$$

$$\vdash B \wedge \Delta \twoheadrightarrow \varphi \;\simeq\; B \wedge \Delta \twoheadrightarrow (\Delta(B) \wedge \varphi) \tag{5.10}$$

$$\vdash B \wedge \Delta \twoheadrightarrow \varphi \;\simeq\; B \wedge \Delta\downarrow_B \twoheadrightarrow \varphi \tag{5.11}$$

$$\vdash \Delta \twoheadrightarrow \varphi \wedge \nabla \twoheadrightarrow \psi \;\simeq\; \Delta\backslash\nabla \twoheadrightarrow \varphi \wedge \nabla\backslash\Delta \twoheadrightarrow \psi \wedge \Delta \cap \nabla \twoheadrightarrow (\varphi \wedge \psi) \tag{5.12}$$

Table 1: Equivalence laws of $\mathcal{L}_{\mathcal{S}}$.

In Table 1 we state a number of laws for equivalence between formulae. To ease notation we introduce a couple of operators on sets of transitions.

$$\Delta(B) = \{s' \in \mathcal{S} \mid (s,s') \in \Delta \wedge s \in B\}$$
$$\Delta^{-1}(B) = \{s \in \mathcal{S} \mid (s,s') \in \Delta \Rightarrow s' \in B\}$$
$$\Delta \circ \nabla = \{(s,s'') \in \mathcal{S} \times \mathcal{S} \mid \exists s' \in \mathcal{S}.\, (s,s') \in \Delta \wedge (s',s'') \in \nabla\}$$
$$\Delta\downarrow_B = \{(s,s'') \in \Delta \mid s \in B\}$$

The laws are sound in the following sense:

Theorem 2 (Soundness) *Whenever* $\vdash \varphi \simeq \psi$ *then* $\models \varphi \simeq \psi$ $\diamond$

The laws enable all recursion-free formulae to be transformed into the following strong normal-form:

Definition 4 *A formula φ is in strong normal-form if φ is of the form:*

$$\varphi \equiv B \wedge \Box\varphi' \wedge \bigwedge_{i \in I} \Delta_i \twoheadrightarrow \varphi_i$$

where φ' and φ_i $(i \in I)$ are themselves in strong normal-form and:

$$\begin{array}{llll} i) & i \neq j \rightarrow \Delta_i \cap \Delta_j = \emptyset & ii) & \Delta_i = \Delta_i \downarrow_B \\ iii) & B_{\varphi_i} = \Delta_i(B) & iv) & B = \bigcap_{i \in I} \Delta_i^{-1}(B_{\varphi_i}) \end{array}$$

where for a formula φ in strong normal-form B_φ refers to the conjunct B of φ. $\diamond$

Theorem 3 *For any recursion-free formula φ of $\mathcal{L}_\mathcal{S}$ there exists a strong normal-form formula φ' such that $\vdash \varphi \simeq \varphi'$.*

Proof: By using the laws (5.1) through (5.8) iteratively we can bring any formula φ into a formula φ' of the required (syntactic) form. To meet the requirements *i)* through *iv)* we can apply the remaining laws (5.9) through (5.13). $\diamond$

For recursion-free formulae the laws of Table 1 are complete with respect to validity and unsatisfiability as stated below:

Theorem 4 (Completeness) *Let φ be a recursion-free formula. Then:*

$$\textit{Whenever } \models \varphi \textit{ then } \vdash \varphi^{\mathcal{S}\times\mathcal{S}} \simeq \mathcal{S} \quad \textit{and} \quad \textit{whenever } \models \neg\varphi \textit{ then } \vdash \varphi^{\emptyset} \simeq \emptyset$$

$\diamond$

It still is an open question as to whether the laws of Table 1 are complete w.r.t. to general formula equivalence — though we conjecture this to be the case.

6 Example

Let us now apply the compositional quotient technique to the program described in Example 1 and the property given in Example 2. For simplicity we consider the following simplified version of the program of Example 1:

$$\left\{\begin{array}{l} P_1 \triangleq \\ \\ \textbf{while } true \textbf{ do} \\ \quad \textbf{if } x \geq MAX \textbf{ then } \ x := x+1 \end{array}\right\} \mid \left\{\begin{array}{l} P_2 \triangleq \\ \\ \textbf{while } true \textbf{ do} \\ \quad \textbf{if } x > 0 \textbf{ then } \ x := x-1 \end{array}\right\}$$

In this simplified version both P_1 and P_2 have a single control location. The state-space therefore degenerates to $\mathbb{N}$ (i.e. the value of the shared variable x). The transition relation given by the components P_1 and P_2 are now:

$$\Sigma_1 = \{(n, n+1) \mid n \in \mathbb{N}\}$$
$$\Sigma_2 = \{(n+1, n) \mid n \in \mathbb{N}\}$$

The property of divergence of P_2 can be expressed simply as: the value of x is invariantly different from 0, i.e:

$$X \triangleq \{0\}^{\complement} \wedge \Box X$$

Now, we claim that for no state (i.e. no $k \in \mathbb{N}$) the above program $P_1 \mid P_2$ will have this property. That is:

Claim 1 *For any* $k \in \mathbb{N}$: $\quad k \not\models_{\Sigma_1 \cup \Sigma_2} X$ $\quad \diamond$

In fact – which should be intuitively clear – component P_2 is entirely to blame for this; no matter what component P_1 does the composite program $P_1 \mid P_2$ will always fail having the property X for all states. That is:

Claim 2 *For any* $\Sigma \subseteq \mathbb{N} \times \mathbb{N}$ *and any* $k \in \mathbb{N}$: $\quad k \not\models_{\Sigma} X/\Sigma_2$ $\quad \diamond$

Note that Claim 1 follows from Claim 2 by taking $\Sigma = \Sigma_1$ and applying the quotient theorem (Theorem 1). To justify Claim 2 for a particular $k \in \mathbb{N}$ we actually only need a certain initial part of Σ_2 ($\Sigma_2^{\leq k}$). For $k \in \mathbb{N}$ define:

$$\Sigma_2^{\leq k} = \{(n+1, n) \mid n < k\}$$
$$\Sigma_2^{> k} = \Sigma_2 \backslash \Sigma^{\leq k}$$

Now Claim 2 follows from the following theorem by instantiating it to $\Sigma \cup \Sigma_2^{>k}$ and using the fact that $\varphi/\Sigma_1/\Sigma_2 \simeq \varphi/(\Sigma_1 \cup \Sigma_2)$.

Theorem 5 *For any $\Sigma \subseteq \mathbb{N} \times \mathbb{N}$ and for any $k \in \mathbb{N}$:* $\quad k \not\models_\Sigma X/\Sigma_2^{\leq k}$

Proof: Let $Y^{\leq k}$ be defined by the following equation:

$$Y^{\leq k} \triangleq \{0 \ldots k\}^{\complement} \wedge \Box Y^{\leq k}$$

We prove by induction on k that $X/\Sigma_2^{\leq k} \simeq Y^{\leq k}$.

BASIS $k = 0$: Trivial, as $\Sigma_2^{\leq 0} = \emptyset$. It is then easy to see that $X/\emptyset \simeq X$ and as the equation for $Y^{\leq 0}$ is exactly that of X, it follows that $X/\Sigma_2^{\leq 0} \simeq_\Sigma Y^{\leq 0}$.

STEP: By definition $\Sigma_2^{\leq k+1} = \Sigma_2^{\leq k} \cup \{(k+1,k)\}$. Again, by using the fact that $\varphi/\Sigma_1/\Sigma_2 \simeq \varphi/(\Sigma_1 \cup \Sigma_2)$ and by induction hypothesis we have:

$$X/\Sigma_2^{\leq k+1} \;\simeq\; (X/\Sigma_2^{\leq k})/\{(k+1,k)\} \;\overset{IH}{\simeq}\; Y^{\leq k}/\{(k+1,k)\}$$

Now expanding the definition of $Y^{\leq k}$ yields:

$$\begin{array}{rcl} Y^{\leq k}/\{(k+1,k)\} & \triangleq & \{0 \ldots k\}^{\complement} \wedge \Box(Y^{\leq k}/\{(k+1,k)\}) \wedge \\ & & \{(k+1,k)\} \twoheadrightarrow (Y^{\leq k}/\{(k+1,k)\}) \\ & \simeq & \{0 \ldots k\}^{\complement} \wedge \Box(Y^{\leq k}/\{(k+1,k)\}) \wedge \\ & & \{(k+1,k)\} \twoheadrightarrow \Big(\{0 \ldots k\}^{\complement} \wedge (Y^{\leq k}/\{(k+1,k)\})\Big) \end{array}$$

Now, by using the laws of Table 1 we can simplify $Y^{\leq k}/\{(k+1,k)\}$ in the following manner[3]:

$$\begin{array}{rcl} \{0 \ldots k\}^{\complement} \wedge \Box Z \wedge \{(k+1,k)\} \twoheadrightarrow \Big(\{0 \ldots k\}^{\complement} \wedge Z\Big) & & \\ \simeq & & \text{by (5.8)} \\ \{0 \ldots k\}^{\complement} \wedge \Box Z \;\wedge\; \{(k+1,k)\} \twoheadrightarrow \{0 \ldots k\}^{\complement} \;\wedge\; \{(k+1,k)\} \twoheadrightarrow Z & & \\ \simeq & & \text{by (5.9)} \\ \{0 \ldots k\}^{\complement} \wedge \Box Z \;\wedge\; \{0 \ldots k+1\}^{\complement} \;\wedge\; \{(k+1,k)\} \twoheadrightarrow Z & & \\ \simeq & & \text{by (5.11)} \\ \{0 \ldots k+1\}^{\complement} \wedge \Box Z \;\wedge\; \{\} \twoheadrightarrow Z & & \\ \simeq & & \text{by (5.2)} \\ \{0 \ldots k+1\}^{\complement} \wedge \Box Z & & \end{array}$$

That is, $Y^{\leq k}/\{(k+1,k)\}$ – and hence $X/\Sigma_2^{\leq k+1}$ – satisfies precisely the defining equation for $Y^{\leq k+1}$. $\diamond$

[3]For better readability we substitute Z for $Y^{\leq k}/\{(k+1,k)\}$

7 Concluding Remarks

We have presented a generalisation of the promising compositional model–checking technique presented in [And95] and extended in [LPY95a, LL95, LPY95b] to programs modelled as arbitrary (possibly infinite–state) transition systems.

Both our quotient construction and laws of equivalence between formulae may be applied to the more specialized settings of [And95, LPY95a, LL95, LPY95b] but also to other settings such as shared variable models as demonstrated in the Example of the paper and studied in [ASM96]. In fact, we believe that the framework offered by our paper may serve as a guideline when instantiating the compositional model–checking technique to a variety of settings.

In a practical implementation care must of course be taken in choosing a suitable and compact representation of the sets of states and the relation on states occurring in formulae. In addition the representation should lead to efficiency in manipulation using the laws of Table 1. For finite state programs BDD's are obvious candidates [ASM96] and for networks of timed automata a variety of techniques for representing subsets of the Euclidean space exists.

Appendix

Theorem 6 (Correctness of Quotienting) *Let Σ_1 and Σ_2 be transition relations, and let $\mathcal{D}$ be a declaration. Then*

$$s \models_{\mathcal{D},\Sigma_1 \cup \Sigma_2} \varphi \quad \Leftrightarrow \quad s \models_{\mathcal{D}^{\Sigma_2},\Sigma_1} \varphi/_{\Sigma_2}$$

$\diamond$

Proof ($\Leftarrow$):: Let $S = \{(s,\varphi) \mid s \models \varphi/_{\Sigma_2}\}$. We prove $S^+ = S \cup \models_{\mathcal{D},\Sigma_1 \cup \Sigma_2}$ is a satisfiability relation under $\mathcal{D}, \Sigma_1 \cup \Sigma_2$ according to the extended Definition 2.

Case $\varphi = \Box\psi$: Assume $(s, \Box\psi) \in S^+$. We must prove:

$$(s,s') \in \Sigma_1 \cup \Sigma_2 \;\Rightarrow\; (s',\psi) \in S^+$$

If $(s,\Box\psi) \in \models_{\mathcal{D},\Sigma_1 \cup \Sigma_2}$ this holds trivially. Hence assume $(s,\Box\psi) \in S$ and $(s,s') \in \Sigma_1 \cup \Sigma_2$:

By definition of S we have $s \models_{\mathcal{D},\Sigma_1} \Box(\psi/_{\Sigma_2}) \wedge \Sigma_2 \rightarrow \psi/_{\Sigma_2}$. We now have two cases: $(s,s') \in \Sigma_1$ and $(s,s') \in \Sigma_2$.

- If $(s,s') \in \Sigma_1$ then $s \models_{\mathcal{D},\Sigma_1} \Box({}^{\psi}/_{\Sigma_2})$ implies $s' \models_{\mathcal{D}^+,\Sigma_1} {}^{\psi}/_{\Sigma_2}$. By definition of S we then have $(s',\psi) \in S^+$.

- If $(s,s') \in \Sigma_2$ then $s \models_{\mathcal{D},\Sigma_1} \Sigma_2 \twoheadrightarrow {}^{\psi}/_{\Sigma_2}$ implies $s' \models_{\mathcal{D}^+,\Sigma_1} {}^{\psi}/_{\Sigma_2}$. By definition of S we then have $(s',\psi) \in S^+$.

Thus, we have established: $(s',\psi) \in S^+$

$(\Rightarrow)$:: Let $S = \{(s, {}^{\varphi}/_{\Sigma_2}) \mid s \models_{\mathcal{D},\Sigma_1 \cup \Sigma_2} \varphi\}$. We prove $S^+ = S \cup \models_{\mathcal{D},\Sigma_1}$ is a satisfiability relation under $\mathcal{D}^+, \Sigma_1$ according to the extended Definition 2.

Case $\varphi = \Box\psi$*:* Assume $(s, {}^{\Box\psi}/_{\Sigma_2}) \in S^+$. By definition of ${}^{\varphi}/_{\Sigma}$ we have ${}^{\Box\varphi}/_{\Sigma_2} = \Box({}^{\psi}/_{\Sigma_2}) \wedge \Sigma_2 \twoheadrightarrow {}^{\varphi}/_{\Sigma_2}$. We must then prove:

- $(s,s') \in \Sigma_1 \;\Rightarrow\; (s', {}^{\psi}/_{\Sigma_2}) \in S^+$

- $(s,s'') \in \Sigma_2 \;\Rightarrow\; (s'', {}^{\psi}/_{\Sigma_2}) \in S^+$

If $(s, \Box({}^{\psi}/_{\Sigma_2}) \wedge \Sigma_2 \twoheadrightarrow {}^{\psi}/_{\Sigma_2}) \in \models_{\mathcal{D}^+,\Sigma_1}$ this holds trivially. Hence assume $(s,\Box\psi) \in S$, $(s,s') \in \Sigma_1$ and $(s,s'') \in \Sigma_2$:

By definition of S we have $s \models_{\mathcal{D}^+,\Sigma_1} \Box({}^{\psi}/_{\Sigma_2}) \wedge \Sigma_2 \twoheadrightarrow {}^{\psi}/_{\Sigma_2}$. We now have two cases: $(s,s') \in \Sigma_1$ and $(s,s') \in \Sigma_2$.

- If $(s,s') \in \Sigma_1$ then $s \models_{\mathcal{D},\Sigma_1} \Box({}^{\psi}/_{\Sigma_2})$ implies $s' \models_{\mathcal{D}^+,\Sigma_1} {}^{\psi}/_{\Sigma_2}$. By definition of S we then have $(s',\psi) \in S^+$.

- If $(s,s'') \in \Sigma_2$ then $s \models_{\mathcal{D},\Sigma_1} \Sigma_2 \twoheadrightarrow {}^{\psi}/_{\Sigma_2}$ implies $s'' \models_{\mathcal{D}^+,\Sigma_1} {}^{\psi}/_{\Sigma_2}$. By definition of S we then have $(s'',\psi) \in S^+$.

Thus, we have established both $(s',\psi) \in S^+$ and $(s'',\psi) \in S^+$. $\diamond$

References

[AD90] R. Alur and D.L. Dill. Automata for modeling real-time systems. In M. Paterson, editor, *Proceedings* 17^{th} *ICALP,* Warwick, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer-Verlag, July 1990.

[AD92] R. Alur and D.L. Dill. The theory of timed automata. In J.W. de Bakker, C. Huizing, W.P. de Roever, and G. Rozenberg, editors, *Proceedings REX Workshop on Real-Time: Theory in*

Practice, Mook, The Netherlands, June 1991, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer-Verlag, 1992.

[And95] H.R. Andersen. Partiel model checking. In *Proceedings of Logics in Computer Science, LICS'95*, 95.

[ASM96] Henrik Reif Andersen, Jørgen Staunstrup, and Niels Maretti. Partial model checking with robdds. In *To appear in Proceedings of TACAS'97*, 1996.

[BCM+90] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking 10^{20} states and beyond. In *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science,* Philadelphia, USA, pages 428–439. IEEE Computer Society Press, 1990.

[CFJ93] E. M. Clarke, T. Filkorn, and S. Jha. Exploiting Symmetry in Temporal Logic Model Checking. 697, 1993. In Proc. of CAV'93.

[CGL92] E. M. Clarke, O. Grümberg, and D. E. Long. Model Checking and Abstraction. *Principles of Programming Languages*, 1992.

[GW91] P. Godefroid and P. Wolper. A partial approach to model checking. In *Proceedings* 6^{th} *Annual Symposium on Logic in Computer Science,* Amsterdam, pages 406–416. IEEE Computer Society Press, 1991.

[KLL+97a] K. Kristoffersen, F. Laroussinie, K.G. Larsen, P. Petterson, and Wang Yi. A compositional proof of a real-time mutual exclusion protocol. In *Proceedings of TAPSOFT97*, 1997.

[KLL+97b] K. Kristoffersen, F. Laroussinie, K.G. Larsen, P. Petterson, and Wang Yi. A compositional proof of a real-time mutual exclusion protocol. In *Proceedings of TAPSOFT97*, 1997.

[Koz82] D. Kozen. Results on the propositional mu–calculus. *Lecture Notes in Computer Science*, 140, 1982. in Proc. of International Colloquium on Algorithms, Languages and Programming 1982.

[LL95] F. Laroussinie and K.G. Larsen. Compositional Model Checking of Real Time Systems. In *Proc. of CONCUR '95*, Lecture Notes in Computer Science. Springer–Verlag, 1995.

[LLW95] François Laroussinie, Kim G. Larsen, and Carsten Weise. From timed automata to logic - and back. *Lecture Notes in Computer Science*, 969:pages 529–539, 1995. In Proceedings of MFCS95.

[LPY95a] Kim G. Larsen, Paul Pettersson, and Wang Yi. Compositional and Symbolic Model-Checking of Real-Time Systems. In *Proc. of the 16*th *IEEE Real-Time Systems Symposium*, pages 76–87, December 1995.

[LPY95b] Kim G. Larsen, Paul Pettersson, and Wang Yi. Model-Checking for Real-Time Systems. In *Proc. of Fundamentals of Computation Theory*, volume 965 of *Lecture Notes in Computer Science*, pages 62–88, 1995.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall International, 1989.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 1955.

[Val90] A. Valmari. A stubborn attack on state explosion. *Theoretical Computer Science*, 3, 1990.

Recent BRICS Report Series Publications

RS-97-13 **Jørgen H. Andersen and Kim G. Larsen.** ***Compositional Safety Logics.*** **June 1997. 16 pp.**

RS-97-12 **Andrej Brodnik, Peter Bro Miltersen, and J. Ian Munro.** ***Trans-Dichotomous Algorithms without Multiplication — some Upper and Lower Bounds.*** **May 1997. 19 pp. Appears in Dehne, Rau-Chaulin, Sack and Tamassio, editors,** ***Algorithms and Data Structures: 5th International Workshop,*** **WADS '97 Proceedings, LNCS 1272, 1997, pages 426–439.**

RS-97-11 **Kārlis Čerāns, Jens Chr. Godskesen, and Kim G. Larsen.** ***Timed Modal Specification — Theory and Tools.*** **April 1997. 32 pp.**

RS-97-10 **Thomas Troels Hildebrandt and Vladimiro Sassone.** ***Transition Systems with Independence and Multi-Arcs.*** **April 1997. 20 pp. Appears in Peled, Pratt and Holzmann, editors,** ***DIMACS Workshop on Partial Order Methods in Verification,*** **POMIV '96, pages 273–288.**

RS-97-9 **Jesper G. Henriksen and P. S. Thiagarajan.** ***A Product Version of Dynamic Linear Time Temporal Logic.*** **April 1997. 18 pp. Appears in Mazurkiewicz and Winkowski, editors,** ***Concurrency Theory: 8th International Conference,*** **CONCUR '97 Proceedings, LNCS 1243, 1997, pages 45–58.**

RS-97-8 **Jesper G. Henriksen and P. S. Thiagarajan.** ***Dynamic Linear Time Temporal Logic.*** **April 1997. 33 pp.**

RS-97-7 **John Hatcliff and Olivier Danvy.** ***Thunks and the λ-Calculus (Extended Version).*** **March 1997. 55 pp. Extended version of article to appear in the** ***Journal of Functional Programming.***

RS-97-6 **Olivier Danvy and Ulrik P. Schultz.** ***Lambda-Dropping: Transforming Recursive Equations into Programs with Block Structure.*** **March 1997. 53 pp. Extended version of an article to appear in the 1997 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation (PEPM '97), Amsterdam, The Netherlands, June 1997.**