BRICS RS-98-50 Aceto & Ingólfsdóttir: Testing Hennessy-Milner Logic with Recursion

BRICS

Basic Research in Computer Science

Testing Hennessy-Milner Logic with Recursion

Luca Aceto
Anna Ingólfsdóttir

BRICS Report Series **RS-98-50**

ISSN 0909-0878 **December 1998**

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/98/50/`

Testing Hennessy-Milner Logic with Recursion*

Luca Aceto** and Anna Ingólfsdóttir***

BRICS†, Department of Computer Science, Aalborg University,
Fredrik Bajers Vej 7-E, DK-9220 Aalborg Ø, Denmark.

Abstract. This study offers a characterization of the collection of properties expressible in Hennessy-Milner Logic (HML) with recursion that can be tested using finite LTSs. In addition to actions used to probe the behaviour of the tested system, the LTSs that we use as tests will be able to perform a distinguished action `nok` to signal their dissatisfaction during the interaction with the tested process. A process s *passes the test* T iff T does not perform the action `nok` when it interacts with s. A test T tests for a property ϕ in HML with recursion iff it is passed by exactly the states that satisfy ϕ. The paper gives an expressive completeness result offering a characterization of the collection of properties in HML with recursion that are testable in the above sense.

1 Introduction

Observational semantics for concurrent processes are based upon the general idea that two processes should be equated, unless they behave differently, in some precise sense, when they are made to interact with some distinguishing environment. Such an idea is, in arguably its purest form, the foundation of the theory of the *testing equivalences* of De Nicola and Hennessy [4, 6]. In the theory of testing equivalence, two processes, described abstractly as *labelled transition systems* (LTSs) [8], are deemed to be equivalent iff they pass exactly the same tests. A *test* is itself an LTS — i.e., a process — which may perform a distinguished action to signal that it is (un)happy with the outcome of its interaction with the tested process. Intuitively, the purpose of submitting a process to a test is to discover whether it enjoys some distinguished property or not. Testing equivalence then stipulates that two processes that enjoy the same properties for which tests can be devised are to be considered equivalent. The main aim of this study is to present a characterization of the collection of properties of concurrent processes that can be tested using LTSs. Of course, in order to be able to even attempt such a characterization (let alone provide it), we need to precisely define a formalism for the description of properties of LTSs, single out a collection of LTSs as tests, and describe the testing process and when an LTS passes or fails a test.

As our specification formalism for properties of processes, we use Hennessy-Milner Logic (HML) with recursion [10]. This is a very expressive property

* The work reported in this paper was mostly carried out during the authors' stay at the Dipartimento di Sistemi ed Informatica, Università di Firenze, Italy.

** Partially supported by a grant from the CNR, Gruppo Nazionale per l'Informatica Matematica (GNIM). Email:`luca@cs.auc.dk`.

*** Supported by the Danish Research Council. Email: `annai@cs.auc.dk`.

† **B**asic **R**esearch **i**n **C**omputer **S**cience, Centre of the Danish National Research Foundation.

language which results from the addition of least and greatest fixed points to the logic considered by Hennessy and Milner in their seminal study [7]. The resulting property language is indeed just a reformulation of the modal μ-calculus [10]. Following the idea of using test automata to check whether processes enjoy properties described by formulae in such a language [2, 1], we use finite LTSs as property testers. In addition to actions used to probe the behaviour of the tested system, the LTSs that we use as tests will be able to perform a distinguished action $\mathtt{nok}$ (read 'not okay') to signal their dissatisfaction during the interaction with the tested process. As in the approach underlying the testing equivalences, a test interacts with a process by communicating with it, and, in keeping with the aforementioned references, the interaction between processes and tests will be described using the (derived) operation of restricted parallel composition from CCS [13].

We say that a process s *fails the test* T iff T can perform the action $\mathtt{nok}$ when it interacts with s. Otherwise s *passes* T. A test T tests for a property ϕ in HML with recursion iff it is passed by exactly the states that satisfy ϕ. The main result of the paper is an expressive completeness result offering a characterization of the collection of properties in HML with recursion that are testable in the above sense. We refer to this language as SHML (for 'safety HML'). More precisely we show that:

- every property ϕ of SHML is testable, in the sense that there exists a test T_ϕ such that s satisfies ϕ if and only if s passes T_ϕ, for every process s; and
- every test T is expressible in SHML, in the sense that there exists a formula ϕ_T of SHML such that, for every process s, the agent s passes T if and only if s satisfies ϕ_T.

This expressive completeness result will be obtained as a corollary of a stronger result pertaining to the compositionality of the property language SHML. A property language is *compositional* if checking whether a composite system $s\|T$ satisfies a property ϕ can be reduced to deciding whether the component s has a corresponding property ϕ/T. As the property ϕ/T is required to be expressible in the property language under consideration, compositionality clearly puts a demand on its expressive power. Let $\mathcal{L}_{\mathtt{nok}}$ be the property language that only contains the simple safety property $[\mathtt{nok}]\mathtt{ff}$, expressing that the $\mathtt{nok}$ action cannot be performed. We prove that SHML is the least expressive, compositional extension of the language $\mathcal{L}_{\mathtt{nok}}$ (Thm. 3.19). This yields the desired expressive completeness result because any compositional property language that can express the property $[\mathtt{nok}]\mathtt{ff}$ is expressive complete with respect to tests (Propn. 3.13). Any increase in expressiveness for the language SHML can only be obtained at the loss of testability.

The paper is organized as follows. After reviewing the model of labelled transition systems and HML with recursion (Sect. 2), we introduce tests and describe how they can be used to test for properties of processes (Sect. 3). We then proceed to argue that not every formula in HML with recursion is testable (Propn. 3.4), but that its sub-language SHML is (Sect. 3.1). Our main results on the compositionality and completeness of SHML are presented in Sect. 3.2.

2 Preliminaries

We begin by briefly reviewing the basic notions from process theory that will be needed in this study. The interested reader is referred to, e.g., [7, 10, 13] for more details.

Labelled Transition Systems Let Act be a set of *actions*, and let a, b range over it. We assume that Act comes equipped with a mapping $\bar{\cdot} : \mathsf{Act} \to \mathsf{Act}$ such that $\bar{\bar{a}} = a$, for every $a \in \mathsf{Act}$. Action $\bar{a}$ is said to be the *complement* of a. We let Act_τ (ranged over by μ) stand for $\mathsf{Act} \cup \{\tau\}$, where τ is a symbol not occurring in Act. Following Milner [13], the symbol τ will stand for an internal action of a system; such actions will typically arise from the synchronization of complementary actions (cf. the rules for the operation of parallel composition in Defn. 2.2).

Definition 2.1. A *labelled transition system (LTS)* over the set of actions Act_τ is a triple $\mathcal{T} = \langle \mathcal{S}, \mathsf{Act}_\tau, \longrightarrow \rangle$ where $\mathcal{S}$ is a set of *states*, and $\longrightarrow \subseteq \mathcal{S} \times \mathsf{Act}_\tau \times \mathcal{S}$ is a *transition relation*. An LTS is *finite* iff its set of states and its transition relation are both finite. It is *rooted* if a distinguished state $\mathsf{root}(\mathcal{T}) \in \mathcal{S}$ is singled out as its start state.

As it is standard practice in process theory, we use the more suggestive notation $s \xrightarrow{\mu} s'$ in lieu of $(s, \mu, s') \in \longrightarrow$. We also write $s \stackrel{\mu}{\not\to}$ if there is no state s' such that $s \xrightarrow{\mu} s'$. Following [13], we now proceed to define versions of the transition relations that abstract from the internal evolution of states as follows:

$$s \stackrel{\varepsilon}{\Rightarrow} s' \quad \text{iff} \quad s \xrightarrow{\tau}{}^{*} s'$$

$$s \stackrel{\mu}{\Rightarrow} s' \quad \text{iff} \quad \exists s_1, s_2. \quad s \stackrel{\varepsilon}{\Rightarrow} s_1 \xrightarrow{\mu} s_2 \stackrel{\varepsilon}{\Rightarrow} s'$$

where we use $\xrightarrow{\tau}{}^{*}$ to stand for the reflexive, transitive closure of $\xrightarrow{\tau}$.

Definition 2.2 (Operations on LTSs).

- Let $\mathcal{T}_i = \langle \mathcal{S}_i, \mathsf{Act}_\tau, \longrightarrow_i \rangle$ $(i \in \{1, 2\})$ be two LTSs. The parallel composition of $\mathcal{T}_1$ and $\mathcal{T}_2$ is the LTS $\mathcal{T}_1 \parallel \mathcal{T}_2 = \langle \mathcal{S}_1 \times \mathcal{S}_2, \mathsf{Act}_\tau, \longrightarrow \rangle$, where the transition relation $\longrightarrow$ is defined by the rules ($\mu \in \mathsf{Act}_\tau$, $a \in \mathsf{Act}$):

$$\frac{s_1 \xrightarrow{\mu}_1 s_1'}{s_1 \| s_2 \xrightarrow{\mu} s_1' \| s_2} \qquad \frac{s_2 \xrightarrow{\mu}_2 s_2'}{s_1 \| s_2 \xrightarrow{\mu} s_1 \| s_2'} \qquad \frac{s_1 \xrightarrow{a}_1 s_1' \qquad s_2 \xrightarrow{\bar{a}}_2 s_2'}{s_1 \| s_2 \xrightarrow{\tau} s_1' \| s_2'}$$

  In the rules above, and in the remainder of the paper, we use the more suggestive notation $s \parallel s'$ in lieu of (s, s').
- Let $\mathcal{T} = \langle \mathcal{S}, \mathsf{Act}_\tau, \to \rangle$ be an LTS and let $L \subseteq \mathsf{Act}$ be a set of actions. The restriction of $\mathcal{T}$ over L is the LTS $\mathcal{T} \backslash L = \langle \mathcal{S} \backslash L, \mathsf{Act}_\tau, \leadsto \rangle$, where $\mathcal{S} \backslash L = \{s \backslash L \mid s \in \mathcal{S}\}$ and the transition relation $\leadsto$ is defined by the rules:

$$\frac{s \xrightarrow{\tau} s'}{s \backslash L \stackrel{\tau}{\leadsto} s' \backslash L} \qquad \frac{s \xrightarrow{a} s'}{s \backslash L \stackrel{a}{\leadsto} s' \backslash L}$$

  where $a, \bar{a} \notin L$.

The reader familiar with [13] may have noticed that the above definitions of parallel composition and restriction are precisely those of CCS. We refer the interested reader to *op. cit.* for more details on these operations.

Hennessy-Milner Logic with Recursion In their seminal study [7], Hennessy and Milner gave a logical characterization of bisimulation equivalence [14] (over states of image-finite LTSs) in terms of a (multi-)modal logic which has since then been referred to as *Hennessy-Milner Logic* (HML). For the sake of completeness and clarity, we now briefly review a variation of this property language for concurrent processes which contains operations for the recursive definition of formulae — a feature that dramatically increases its expressive power. The interested reader is referred to, e.g., [10] for more details.

Definition 2.3. Let Var be a countably infinite set of formula variables, and let $\mathtt{nok}$ denote an action symbol not contained in Act. The collection $\mathsf{HML}(\mathsf{Var})$ of formulae over Var and $\mathsf{Act} \cup \{\mathtt{nok}\}$ is given by the following grammar:

$$\phi ::= \mathtt{tt} \mid \mathtt{ff} \mid \phi \vee \phi \mid \phi \wedge \phi \mid \langle \alpha \rangle \phi \mid [\alpha]\phi \mid X \mid \mathtt{min}(X, \phi) \mid \mathtt{max}(X, \phi)$$

where $\alpha \in \mathsf{Act} \cup \{\mathtt{nok}\}$, X is a formula variable and $\mathtt{min}(X, \phi)$ (respectively, $\mathtt{max}(X, \varphi)$) stands for the least (respectively, largest) solution of the recursion equation $X = \varphi$.

We use $\mathsf{SHML}(\mathsf{Var})$ (for 'safety HML') to stand for the collection of formulae in $\mathsf{HML}(\mathsf{Var})$ that do not contain occurrences of $\vee$, $\langle \alpha \rangle$ and $\mathtt{min}(X, \phi)$.

A *closed recursive formula* of $\mathsf{HML}(\mathsf{Var})$ is a formula in which every formula variable X is *bound*, i.e., every occurrence of X appears within the scope of some $\mathtt{min}(X, \phi)$ or $\mathtt{max}(X, \phi)$ construct. A variable X is *free* in the formula ϕ if some occurrence of it in ϕ is not bound. For example, the formula $\mathtt{max}(X, X)$ is closed, but $\mathtt{min}(X, [a]Y)$ is not because Y is free in it. The collection of closed formulae contained in $\mathsf{HML}(\mathsf{Var})$ (respectively, $\mathsf{SHML}(\mathsf{Var})$) will be written HML (resp. SHML). In the remainder of this paper, every formula will be closed, unless specified otherwise, and we shall identify formulae that only differ in the names of their bound variables. For formulae ϕ and ψ, and a variable X, we write $\phi\{\psi/X\}$ for the formula obtained by replacing every free occurrence of X in ϕ with ψ. The details of such an operation in the presence of binders are standard (see, e.g., [15]), and are omitted here.

Given an LTS $\mathcal{T} = \langle \mathcal{S}, \mathsf{Act}_\tau, \longrightarrow \rangle$, an *environment* is a mapping $\rho : \mathsf{Var} \to 2^{\mathcal{S}}$. For an environment ρ, variable X and subset of states S, we write $\rho[X \mapsto S]$ for the environment mapping X to S, and acting like ρ on all the other variables.

Definition 2.4 (Satisfaction Relation). Let $\mathcal{T} = \langle \mathcal{S}, \mathsf{Act}_\tau, \longrightarrow \rangle$ be an LTS. For every environment ρ and formula φ contained in $\mathsf{HML}(\mathsf{Var})$, the collection $[\![\varphi]\!]\rho$ of states in $\mathcal{S}$ satisfying the formula φ with respect to ρ is defined by structural recursion on φ thus:

$$
\begin{aligned}
[\![\mathtt{tt}]\!]\rho &\stackrel{\mathrm{def}}{=} \mathcal{S} \\
[\![\mathtt{ff}]\!]\rho &\stackrel{\mathrm{def}}{=} \varnothing \\
[\![\varphi_1 \vee \varphi_2]\!] &\stackrel{\mathrm{def}}{=} [\![\varphi_1]\!]\rho \cup [\![\varphi_2]\!]\rho \\
[\![\varphi_1 \wedge \varphi_2]\!] &\stackrel{\mathrm{def}}{=} [\![\varphi_1]\!]\rho \cap [\![\varphi_2]\!]\rho \\
[\![\langle\alpha\rangle\varphi]\!] &\stackrel{\mathrm{def}}{=} \left\{ s \mid s \stackrel{\alpha}{\Rightarrow} s' \text{ for some } s' \in [\![\varphi]\!]\rho \right\} \\
[\![[\alpha]\varphi]\!] &\stackrel{\mathrm{def}}{=} \left\{ s \mid \text{for every } s',\ s \stackrel{\alpha}{\Rightarrow} s' \text{ implies } s' \in [\![\varphi]\!]\rho \right\} \\
[\![X]\!]\rho &\stackrel{\mathrm{def}}{=} \rho(X) \\
[\![\mathtt{min}(X,\varphi)]\!]\rho &\stackrel{\mathrm{def}}{=} \bigcap \{ S \mid [\![\varphi]\!]\rho[X \mapsto S] \subseteq S \} \\
[\![\mathtt{max}(X,\varphi)]\!]\rho &\stackrel{\mathrm{def}}{=} \bigcup \{ S \mid S \subseteq [\![\varphi]\!]\rho[X \mapsto S] \} \ .
\end{aligned}
$$

The interested reader will find more details on this definition in, e.g., [10]. Here we just confine ourselves to remarking that, as the interpretation of each formula ϕ containing at most X free induces a monotone mapping $[\![\phi]\!] : 2^{\mathcal{S}} \to 2^{\mathcal{S}}$, the closed formulae $\mathtt{min}(X,\phi)$ and $\mathtt{max}(X,\phi)$ are indeed interpreted as the least and largest solutions, respectively, of the equation $X = \phi$. If φ is a closed formula, then the collection of states satisfying it is independent of the environment ρ, and will be written $[\![\varphi]\!]$. In the sequel, for every state s and closed formula φ, we shall write $s \models \varphi$ (read 's satisfies φ') in lieu of $s \in [\![\varphi]\!]$.

When restricted to SHML, the satisfaction relation $\models$ is the largest relation included in $\mathcal{S} \times \mathsf{SHML}$ satisfying the implications in Table 1. A relation satisfying the defining implications for $\models$ will be called a *satisfiability relation*. It follows from standard fixed-point theory [16] that, over $\mathcal{S} \times \mathsf{HML}$, the relation $\models$ is the union of all satisfiability relations and that the above implications are in fact biimplications for $\models$.

Remark. Since $\mathtt{nok}$ is not contained in Act, every state of an LTS trivially satisfies formulae of the form $[\mathtt{nok}]\phi$. The role played by these formulae in the developments of this paper will become clear in Sect. 3.2. Dually, no state of an LTS satisfies formulae of the form $\langle\mathtt{nok}\rangle\varphi$.

Formulae ϕ and ψ are *logically equivalent* (with respect to $\models$) iff they are satisfied by the same states. We say that a formula is *satisfiable* iff it is satisfied by at least one state in some LTS, otherwise we say that it is *unsatisfiable*.

3 Testing Formulae

As mentioned in Sect. 1, the main aim of this paper is to present a complete characterization of the class of testable properties of states of LTSs that can be expressed in the language HML. In this section we define the collection of *tests* and the notion of *property testing* used in this study. Informally, testing involves the parallel composition of the tested state with a test. Following the spirit of the classic approach of De Nicola and Hennessy [4, 6], we say that the

$$
\begin{array}{rcl}
s \models \mathtt{tt} & \Rightarrow & \mathit{true} \\
s \models \mathtt{ff} & \Rightarrow & \mathit{false} \\
s \models \varphi_1 \wedge \varphi_2 & \Rightarrow & s \models \varphi_1 \text{ and } s \models \varphi_2 \\
s \models [\alpha]\varphi & \Rightarrow & \forall s'.\ s \stackrel{\alpha}{\Rightarrow} s' \text{ implies } s' \models \varphi \\
s \models \mathtt{max}(X, \varphi) & \Rightarrow & s \models \varphi\{\mathtt{max}(X, \varphi)/X\}
\end{array}
$$

Table 1. Satisfaction implications

tested state fails a test if the distinguished reject action $\mathtt{nok}$ can be performed by the test while it interacts with it, and passes otherwise. The formal definition of testing then involves the definition of what a test is, how interaction takes place and when the test has failed or succeeded. We now proceed to make these notions precise.

Definition 3.1 (Tests). A *test* is a finite, rooted LTS over the set of actions $\mathsf{Act}_\tau \cup \{\mathtt{nok}\}$.

In the remainder of this study, tests will often be concisely described using the regular fragment of Milner's CCS [13] given by the following grammar:

$$T ::= \mathbf{0} \mid \alpha.T \mid T + T \mid X \mid \mathrm{fix}(X = T)$$

where $\alpha \in \mathsf{Act}_\tau \cup \{\mathtt{nok}\}$, and X ranges over Var. As usual, we shall only be concerned with the closed expressions generated by the above grammar, with $\mathrm{fix}(X = T)$ as the binding construct, and we shall identify expressions that only differ in the names of their bound variables. In the sequel, the symbol $\equiv$ will be used to denote syntactic equality up to renaming of bound variables. The operation of substitution over the set of expressions given above is defined exactly as for formulae in $\mathsf{HML}(\mathsf{Var})$. The operational semantics of the expressions generated by the above grammar is given by the classic rules for CCS. These are reported below for the sake of clarity:

$$
\frac{}{\alpha.T \stackrel{\alpha}{\to} T}
\qquad
\frac{T_1 \stackrel{\alpha}{\to} T_1'}{T_1 + T_2 \stackrel{\alpha}{\to} T_1'}
\qquad
\frac{T_2 \stackrel{\alpha}{\to} T_2'}{T_1 + T_2 \stackrel{\alpha}{\to} T_2'}
\qquad
\frac{T\{\mathrm{fix}(X = T)/X\} \stackrel{\alpha}{\to} T'}{\mathrm{fix}(X = T) \stackrel{\alpha}{\to} T'}
$$

where α is either $\mathtt{nok}$ or an action in Act_τ. The intention is that the term T stands for the test whose start state is T itself, whose transitions are precisely those that are provable using the above inference rules, and whose set of states is the collection of expressions reachable from T by performing zero or more transitions. We refer the reader to [13] for more information on the operational semantics of CCS.

Definition 3.2 (Testing Properties). Let φ be a formula in HML, and let T be a test.

– A state s of an LTS passes the test T iff $(s \| \mathsf{root}(T)) \backslash \mathsf{Act} \stackrel{\mathtt{nok}}{\not\Rightarrow}$. Otherwise we say that s fails the test T.

– We say that the test T *tests* for the formula φ (and that φ is *testable*) iff for every LTS $\mathcal{T}$ and every state s of $\mathcal{T}$, $s \models \varphi$ iff s passes the test T.
– Let $\mathcal{L}$ be a collection of formulae in HML. We say that $\mathcal{L}$ is testable iff each of the formulae in $\mathcal{L}$ is.

Example 3.3. The formula $[a]\mathtt{ff}$ states that a process does not afford a $\stackrel{a}{\Rightarrow}$-transition. We therefore expect that a suitable test for such a property is $T \equiv \bar{a}.\mathtt{nok}.\mathbf{0}$. Indeed, the reader will easily realize that $(s \| T) \backslash \mathsf{Act} \stackrel{\mathtt{nok}}{\not\Rightarrow}$ iff $s \stackrel{a}{\not\Rightarrow}$, for every state s. The formula $[a]\mathtt{ff}$ is thus testable, in the sense of this paper.

The formula $\mathtt{max}(X, [a]\mathtt{ff} \wedge [b]X)$ is satisfied by those states which cannot perform a $\stackrel{a}{\Rightarrow}$-transition, no matter how they engage in a sequence of $\stackrel{b}{\Rightarrow}$-transitions. A suitable test for such a property is $\mathrm{fix}(X = \bar{a}.\mathtt{nok}.\mathbf{0} + \bar{b}.X)$, and the formula $\mathtt{max}(X, [a]\mathtt{ff} \wedge [b]X)$ is thus testable.

As already stated, our main aim in this paper is to present a characterization of the collection of HML-properties that are testable in the sense of Defn. 3.2. To this end, we begin by providing evidence to the effect that not every property expressible in HML is testable.

Proposition 3.4 (Two Negative Results).

1. *Let ϕ be a formula in HML. Suppose that ϕ is satisfiable. Then, for every action a in Act, the formula $\langle a \rangle \phi$ is not testable.*
2. *Let a and b be two distinct actions in Act. Then the formula $[a]\mathtt{ff} \vee [b]\mathtt{ff}$ is not testable.*

Remark. If φ is unsatisfiable, then the formula $\langle a \rangle \varphi$ is logically equivalent to $\mathtt{ff}$. Since $\mathtt{ff}$ is testable using the test $\mathtt{nok}.\mathbf{0}$, the requirement on φ is necessary for Propn. 3.4(1) to hold. Note moreover that, as previously remarked, both the formulae $[a]\mathtt{ff}$ and $[b]\mathtt{ff}$ are testable, but their disjunction is not (Propn. 3.4(2)).

Our aim in the remainder of this paper is to show that the collection of testable properties is precisely SHML. This is formalized by the following result.

Theorem 3.5. *The collection of formulae SHML is testable. Moreover, every testable property in HML can be expressed in SHML.*

The remainder of this paper will be devoted to a proof of the above theorem. In the process of developing such a proof, we shall also establish some results pertaining to the expressive power of SHML which may be of independent interest.

3.1 Testability of SHML

We begin our proof of Thm. 3.5 by showing that the language SHML is testable. To this end, we define, for every open formula ϕ in the language $\mathsf{SHML}(\mathsf{Var})$, a regular CCS expression T_ϕ by structural recursion thus:

$$\begin{array}{rlrl} T_{\mathtt{tt}} & \stackrel{\mathrm{def}}{=} \mathbf{0} & T_{[a]\phi} & \stackrel{\mathrm{def}}{=} \bar{a}.T_\phi \\ T_{\mathtt{ff}} & \stackrel{\mathrm{def}}{=} \mathtt{nok}.\mathbf{0} & T_X & \stackrel{\mathrm{def}}{=} X \\ T_{\phi_1 \wedge \phi_2} & \stackrel{\mathrm{def}}{=} \tau.T_{\phi_1} + \tau.T_{\phi_2} & T_{\mathtt{max}(X,\phi)} & \stackrel{\mathrm{def}}{=} \mathrm{fix}(X = T_\phi) \ . \end{array}$$

For example, if $\phi \equiv \mathtt{max}(X, [a]\mathtt{ff} \wedge [b]X)$ then T_ϕ is the test $\mathrm{fix}(X = \tau.\bar{a}.\mathtt{nok}.\mathbf{0} + \tau.\bar{b}.X)$. We recall that we identify CCS descriptions of tests that only differ in the name of their bound variables since they give rise to isomorphic LTSs. Our order of business in this section will be to show the following result:

Theorem 3.6. *Let ϕ be a closed formula contained in* SHML. *Then the test T_ϕ tests for it.*

In the proof of this theorem, it will be convenient to have an alternative, novel characterization of the satisfaction relation for formulae in the language SHML. This we now proceed to present.

Definition 3.7. Let $\mathcal{T} = \langle \mathcal{S}, \mathsf{Act}_\tau, \longrightarrow \rangle$ be an LTS. The satisfaction relation $\models_\varepsilon$ is the largest relation included in $\mathcal{S} \times \mathsf{SHML}$ satisfying the following implications:

$$
\begin{array}{rcl}
s \models_\varepsilon \mathtt{tt} & \Rightarrow & \mathit{true} \\
s \models_\varepsilon \mathtt{ff} & \Rightarrow & \mathit{false} \\
s \models_\varepsilon \varphi_1 \wedge \varphi_2 & \Rightarrow & s' \models_\varepsilon \varphi_1 \text{ and } s' \models_\varepsilon \varphi_2, \text{ for every } s' \text{ such that } s \stackrel{\varepsilon}{\Rightarrow} s' \\
s \models_\varepsilon [a]\varphi & \Rightarrow & s \stackrel{a}{\Rightarrow} s' \text{ implies } s' \models_\varepsilon \varphi, \text{ for every } s' \\
s \models_\varepsilon \mathtt{max}(X, \varphi) & \Rightarrow & s' \models_\varepsilon \varphi\{\mathtt{max}(X, \varphi)/X\}, \text{ for every } s' \text{ such that } s \stackrel{\varepsilon}{\Rightarrow} s'
\end{array}
$$

A relation satisfying the above implications will be called a *weak satisfiability relation.*

The satisfaction relation $\models_\varepsilon$ is closed with respect to the relation $\stackrel{\varepsilon}{\Rightarrow}$, in the sense of the following proposition.

Proposition 3.8. *Let $\mathcal{T} = \langle \mathcal{S}, \mathsf{Act}_\tau, \longrightarrow \rangle$ be an LTS. Then, for every $s \in \mathcal{S}$ and $\varphi \in \mathsf{SHML}$, $s \models_\varepsilon \varphi$ iff $s' \models_\varepsilon \varphi$, for every s' such that $s \stackrel{\varepsilon}{\Rightarrow} s'$.*

Proof. The only interesting thing to check is that if $s \models_\varepsilon \varphi$ and $s \stackrel{\varepsilon}{\Rightarrow} s'$, then $s' \models_\varepsilon \varphi$. To this end, it is sufficient to prove that the relation $\mathcal{R}$ defined thus:

$$\mathcal{R} \stackrel{\mathrm{def}}{=} \{(s, \varphi) \mid \exists t.\ t \models_\varepsilon \varphi \text{ and } t \stackrel{\varepsilon}{\Rightarrow} s\}$$

is a weak satisfiability relation. The straightforward verification is left to the reader. □

We now proceed to establish that the relations $\models_\varepsilon$ and $\models$ coincide for formulae in SHML.

Proposition 3.9. *Let ϕ be a formula contained in* SHML. *Then, for every state s of an LTS, $s \models \phi$ iff $s \models_\varepsilon \phi$.*

In the proof of Thm. 3.6, it will be convenient to have at our disposal some further auxiliary results. For ease of reference, these are collected in the following lemma.

Lemma 3.10.

1. *Let ϕ be a formula in* SHML*. Assume that* $T_\phi \stackrel{\mathsf{nok}}{\rightarrow}$*. Then ϕ is logically equivalent to* $\mathbf{ff}$*.*
2. *Let ϕ be a formula in* SHML*. Assume that $T_\phi \stackrel{\tau}{\rightarrow} T$. Then there are formulae ϕ_1 and ϕ_2 in* SHML *such that $T \equiv T_{\phi_1}$, and ϕ is logically equivalent to $\phi_1 \wedge \phi_2$.*
3. *Let ϕ be a formula in* SHML*. Assume that $T_\phi \stackrel{\bar{a}}{\rightarrow} T$. Then there is a formula ψ in* SHML *such that $T \equiv T_\psi$, and ϕ is logically equivalent to $[a]\psi$.*

Using these results, we are now in a position to prove Thm. 3.6.

Proof of Thm. 3.6: In light of Propn. 3.9, it is sufficient to show that, for every state s of an LTS and closed formula $\phi \in \mathsf{SHML}$,

$$s \models_\varepsilon \phi \text{ iff } (s \| T_\phi)\backslash\mathsf{Act} \stackrel{\mathsf{nok}}{\not\Rightarrow} \ .$$

We prove the two implications separately.

- 'If Implication'. It is sufficient to show that the relation

$$\mathcal{R} \stackrel{\text{def}}{=} \left\{ (s,\phi) \mid (s\|T_\phi)\backslash\mathsf{Act} \stackrel{\mathsf{nok}}{\not\Rightarrow} \text{ and } \phi \in \mathsf{SHML} \right\}$$

  is a weak satisfiability relation. The details of the proof are left to the reader.
- 'Only If Implication'. We prove the contrapositive statement. To this end, assume that

$$(s\|T_\phi)\backslash\mathsf{Act} \stackrel{\varepsilon}{\Rightarrow} (s'\|T')\backslash\mathsf{Act} \stackrel{\mathsf{nok}}{\rightarrow}$$

  for some state s' and test T'. We show that $s \not\models_\varepsilon \phi$ holds by induction on the length of the computation $(s\|T_\phi)\backslash\mathsf{Act} \stackrel{\varepsilon}{\Rightarrow} (s'\|T')\backslash\mathsf{Act}$.

  - Base Case: $(s\|T_\phi)\backslash\mathsf{Act} \equiv (s'\|T')\backslash\mathsf{Act} \stackrel{\mathsf{nok}}{\rightarrow}$. In this case, we may infer that $T_\phi \stackrel{\mathsf{nok}}{\rightarrow}$. By Lemma 3.10(1), it follows that ϕ is unsatisfiable. Propn. 3.9 now yields that $s \not\models_\varepsilon \phi$, which was to be shown.
  - Inductive Step: $(s\|T_\phi)\backslash\mathsf{Act} \stackrel{\tau}{\rightarrow} (s''\|T'')\backslash\mathsf{Act} \stackrel{\varepsilon}{\Rightarrow} (s'\|T')\backslash\mathsf{Act} \stackrel{\mathsf{nok}}{\rightarrow}$, for some state s'' and test T''. We proceed by a case analysis on the form the transition

$$(s\|T_\phi)\backslash\mathsf{Act} \stackrel{\tau}{\rightarrow} (s''\|T'')\backslash\mathsf{Act}$$

    may take.
    * Case: $s \stackrel{\tau}{\rightarrow} s''$ and $T'' \equiv T_\phi$.
      In this case, we may apply the inductive hypothesis to infer that $s'' \not\models_\varepsilon \phi$. By Propn. 3.8, it follows that $s \not\models_\varepsilon \phi$, which was to be shown.
    * Case: $T_\phi \stackrel{\tau}{\rightarrow} T''$ and $s = s''$.
      By Lemma 3.10(2), it follows that ϕ is logically equivalent to $\phi_1 \wedge \phi_2$ for some formulae ϕ_1 and ϕ_2 in SHML, and that $T'' \equiv T_{\phi_1}$. By induction, we may now infer that $s \not\models_\varepsilon \phi_1$. Since ϕ is logically equivalent to $\phi_1 \wedge \phi_2$, this implies that $s \not\models_\varepsilon \phi$ (Propn. 3.9), which was to be shown.

* Case: $s \xrightarrow{a} s''$ and $T_\phi \xrightarrow{\bar{a}} T''$, for some action $a \in \mathsf{Act}$.
  By Lemma 3.10(3), it follows that ϕ is logically equivalent to $[a]\psi$ for some formula ψ in SHML, and that $T'' \equiv T_\psi$. By induction, we may now infer that $s'' \not\models_\varepsilon \psi$. Since ϕ is logically equivalent to $[a]\psi$ and $s \xrightarrow{a} s'' \not\models_\varepsilon \psi$, this implies that $s \not\models_\varepsilon \phi$ (Propn. 3.9), which was to be shown.

This completes the inductive argument, and the proof of the 'only if' implication.

The proof of the theorem is now complete. □

3.2 Expressive Completeness of SHML

We have just shown that every property φ which can be expressed in the language SHML is testable, in the sense of Defn. 3.2. We now address the problem of the expressive completeness of this property language with respect to tests. More precisely, we study whether all properties that are testable can be expressed in the property language SHML — in the sense that, for every test T, there exists a formula ψ_T in SHML such that every state of an LTS passes the test T if, and only if, it satisfies ψ_T. Our aim in this section is to complete the proof of Thm. 3.5 by arguing that the language SHML *is* expressive complete, in the sense that every test T may be expressed as a property in the language SHML in the precise technical sense outlined above. This amounts to establishing an expressive completeness result for SHML akin to classic ones presented in, e.g., [9, 5, 17]. In the proof of this expressive completeness result, we shall follow an indirect approach by focusing on the compositionality of a property language $\mathcal{L}$ with respect to tests and the parallel composition operator $\|$. As we shall see (cf. Propn. 3.13), if a property language $\mathcal{L}$, that contains the property $[\mathtt{nok}]\mathtt{ff}$, is compositional with respect to tests and $\|$ (cf. Defn. 3.12) then it is expressive complete (cf. Defn. 3.11). We shall show that SHML is compositional with respect to tests and $\|$, and obtain the expressive completeness of such a language as a corollary of this stronger result.

We begin with some preliminary definitions, introducing the key concepts of compositionality and (expressive) completeness.

Definition 3.11 (Expressive completeness). Let $\mathcal{L}$ be a collection of formulae in HML. We say that $\mathcal{L}$ is *(expressive) complete* (with respect to tests) if for every test T there exists a formula $\varphi_T \in \mathcal{L}$ such that, for every state s of an LTS, $s \models \varphi_T$ iff s passes the test T.

Compositionality, on the other hand, is formally defined as follows:

Definition 3.12 (Compositionality). Let $\mathcal{L}$ be a collection of formulae in HML. We say that $\mathcal{L}$ is *compositional* (with respect to tests and $\|$) if, for every $\varphi \in \mathcal{L}$ and every test T, there exists a formula $\varphi/T \in \mathcal{L}$ such that, for every state s of an LTS, $s \| \mathsf{root}(T) \models \varphi$ iff $s \models \varphi/T$.

Intuitively, the formula φ/T states a necessary and sufficient condition for state s to satisfy φ when it is made to interact with the test T.

Our interest in compositionality stems from the following result that links it to the notion of completeness. In the sequel, we use $\mathcal{L}_{\mathtt{nok}}$ to denote the property language that only consists of the formula $[\mathtt{nok}]\mathtt{ff}$. (Recall that $\mathtt{nok}$ is a fresh action not contained in Act.)

Proposition 3.13. *Let $\mathcal{L}$ be a collection of formulae in* HML *that includes $\mathcal{L}_{\mathtt{nok}}$. Suppose that $\mathcal{L}$ is compositional. Then $\mathcal{L}$ is complete with respect to tests.*

Proof. Consider an arbitrary test T. We aim at exhibiting a formula $\phi_T \in \mathcal{L}$ meeting the requirements in Defn. 3.11. Since $\mathcal{L}$ is compositional and contains the formula $[\mathtt{nok}]\mathtt{ff}$, we may define φ_T to be the formula $([\mathtt{nok}]\mathtt{ff})/T$. Let s be an arbitrary state of an LTS. We can now argue that s passes T iff it satisfies ϕ_T thus:

$$
\begin{aligned}
s \text{ passes the test } T &\text{ iff } (s\|\mathsf{root}(T))\backslash\mathsf{Act} \stackrel{\mathtt{nok}}{\not\Rightarrow} \\
&\text{ iff } (s\|\mathsf{root}(T))\backslash\mathsf{Act} \models [\mathtt{nok}]\mathtt{ff} \\
&\text{ iff } (s\|\mathsf{root}(T)) \models [\mathtt{nok}]\mathtt{ff} \\
&\qquad (\text{As } \mathtt{nok} \notin \mathsf{Act}) \\
&\text{ iff } s \models ([\mathtt{nok}]\mathtt{ff})/T \\
&\qquad (\text{As } \mathcal{L} \text{ is compositional}) \\
&\text{ iff } s \models \varphi_T \quad .
\end{aligned}
$$

This completes the proof. $\square$

As we shall now show, SHML is compositional with respect to tests and $\|$, and thus expressive complete with respect to tests. We begin by defining a quotient construction for formulae of SHML, in the spirit of those given for different property languages and over different models in, e.g., [12, 3, 11].

Definition 3.14 (Quotient Construction). Let T be a test, and let t be one of its states. For every formula φ SHML, we define the formula φ/t (read 'φ quotiented by t') as shown in Table 2.

$$
\begin{aligned}
\mathtt{ff}/t &\stackrel{\text{def}}{=} \mathtt{ff} \\
\mathtt{tt}/t &\stackrel{\text{def}}{=} \mathtt{tt} \\
(\phi_1 \wedge \phi_2)/t &\stackrel{\text{def}}{=} \phi_1/t \wedge \phi_2/t \\
([\alpha]\phi)/t &\stackrel{\text{def}}{=} [\alpha](\phi/t) \wedge \bigwedge_{\left\{t' \mid t \stackrel{\bar{\alpha}}{\Rightarrow} t'\right\}} (\phi/t') \wedge \bigwedge_{\left\{(b,t') \mid t \stackrel{b}{\Rightarrow} t'\right\}} [\bar{b}](([\alpha]\phi)/t') \\
\mathtt{max}(X,\phi)/t &\stackrel{\text{def}}{=} (\phi\{\mathtt{max}(X,\phi)/X\})/t
\end{aligned}
$$

Table 2. Quotient construct for SHML

Some remarks about the definition presented in Table 2 are now in order. The definition of the quotient formula φ/t presented *ibidem* should be read as yielding a finite list of recursion equations, over variables of the form ψ/t', for every formula φ and state t of a test. The quotient formula φ/t itself is the component associated with φ/t in the largest solution of the system of equations having φ/t as leading variable. For instance, if φ is the formula $[a]\mathtt{ff}$ and t is a node of a test whose only transition is $t \xrightarrow{\bar{b}} t$, then, as the reader can easily verify, φ/t is the largest solution of the recursion equation:

$$\varphi/t \stackrel{\text{def}}{=} [a]\mathtt{ff} \wedge [b](\varphi/t)$$

which corresponds to the formula $\mathtt{max}(X, [a]\mathtt{ff} \wedge [b]X)$ in the property language SHML. This formula states the, intuitively clear, fact that a state of the form $s \parallel t$ cannot perform a $\stackrel{a}{\Rightarrow}$-transition iff s cannot execute such a step no matter how it engages in a sequence of synchronizations on b with t. Note that the quotient of a recursion-free formula may be a formula involving recursion. It can be shown that this is inevitable, because the recursion-free fragment of SHML is *not* compositional. Finally, we remark that, because of our finiteness restrictions on tests, the right-hand side of the defining equation for $([\alpha]\phi)/t$ is a finite conjunction of formulae.

The following key result states the correctness of the quotient construction.

Theorem 3.15. *Let φ be a closed formula in SHML. Suppose that s is a state of an LTS, and t is a state of a test. Then $s \parallel t \models \varphi$ iff $s \models \varphi/t$.*

Proof. We prove the two implications separately.

- 'Only If Implication'. Consider the environment ρ mapping each variable φ/t in the list of equations in Table 2 to the set of states $\{s \mid s\|t \models \varphi\}$. We prove that ρ is a post-fixed point of the monotonic functional on environments associated with the equations in Table 2, i.e., that if $s \in \rho(\phi/t)$ then $s \in [\![\psi]\!]\rho$, where ψ is the right-hand side of the defining equation for ϕ/t. This we now proceed to do by a case analysis on the form the formula φ may take. We only present the details for the most interesting case in the proof.
  - Case: $\varphi \equiv [\alpha]\psi$. Assume that $s\|t \models [\alpha]\psi$. We show that state s is contained in $[\![\xi]\!]\rho$ for every conjunct ξ in the right-hand side of the defining equation for $([\alpha]\psi)/t$.
    - Case: $\xi \equiv [\alpha](\psi/t)$. To show that $s \in [\![\xi]\!]\rho$, it is sufficient to prove that $s' \in [\![\psi/t]\!]\rho$, for every s' such that $s \stackrel{\alpha}{\Rightarrow} s'$. To this end, we reason as follows:

$$\begin{array}{rl} s \stackrel{\alpha}{\Rightarrow} s' \text{ implies} & s\|t \stackrel{\alpha}{\Rightarrow} s'\|t \\ \text{implies} & s'\|t \models \psi \\ & \quad (\text{As } s\|t \models [\alpha]\psi) \\ \text{iff} & s' \in \rho(\psi/t) \\ & \quad (\text{By the definition of } \rho) \\ \text{iff} & s' \in [\![\psi/t]\!]\rho \ . \end{array}$$

∗ CASE: $\xi \equiv \psi/t'$ with $t \stackrel{\alpha}{\Rightarrow} t'$. To show that $s \in [\![\xi]\!]\rho$, it is sufficient to prove that $s \in [\![\psi/t']\!]\rho$, for every t' such that $t \stackrel{\alpha}{\Rightarrow} t'$. To this end, we reason as follows:

$$\begin{aligned}
t \stackrel{\alpha}{\Rightarrow} t' \text{ implies } & s\|t \stackrel{\alpha}{\Rightarrow} s\|t' \\
\text{implies } & s\|t' \models \psi \\
& \quad (\text{As } s\|t \models [\alpha]\psi) \\
\text{iff } \quad & s \in \rho(\psi/t') \\
& \quad (\text{By the definition of } \rho) \\
\text{iff } \quad & s \in [\![\psi/t']\!]\rho \ .
\end{aligned}$$

∗ CASE: $\xi \equiv [\bar{b}]\big(([\alpha]\psi)/t'\big)$ with $t \stackrel{b}{\Rightarrow} t'$. To show that $s \in [\![\xi]\!]\rho$, it is sufficient to prove that $s' \in [\![([\alpha]\psi)/t']\!]\rho$, for every s' such that $s \stackrel{\bar{b}}{\Rightarrow} s'$. To this end, we reason as follows:

$$\begin{aligned}
s \stackrel{\bar{b}}{\Rightarrow} s' \text{ and } t \stackrel{b}{\Rightarrow} t' \text{ imply } \ & s\|t \stackrel{\tau}{\Rightarrow} s'\|t' \\
\text{implies } & s'\|t' \models [\alpha]\psi \\
& \quad (\text{By Propns. 3.8 and 3.9, as } s\|t \models [\alpha]\psi) \\
\text{iff } \quad & s' \in \rho(([\alpha]\psi)/t') \\
& \quad (\text{By the definition of } \rho) \\
\text{iff } \quad & s' \in [\![([\alpha]\psi)/t']\!]\rho \ .
\end{aligned}$$

The proof for the case $\phi \equiv [\alpha]\psi$ is now complete.

– ‘IF IMPLICATION’. Consider the relation $\mathcal{R}$ defined thus:

$$\mathcal{R} \stackrel{\text{def}}{=} \{(s\|t, \varphi) \mid s \models \varphi/t\} \ .$$

It is not hard to show that $\mathcal{R}$ is a satisfiability relation.

The proof of the theorem is now complete. □

Corollary 3.16. *The property language* SHML *is compositional with respect to tests and the parallel composition operator* $\|$.

Proof. Given a property $\varphi \in$ SHML and a test T, define φ/T to be the formula $\varphi/\mathsf{root}(T)$ given by the quotient construction. The claim is now an immediate consequence of Thm. 3.15. □

Theorem 3.17. *The property language* SHML *is expressive complete.*

Example 3.18. Applying the construction in the proof of Propn. 3.13, and the definition of the quotient formula to the tests

$$\begin{aligned}
T_1 &\equiv \text{fix}(X = \bar{a}.\mathtt{nok}.\mathbf{0} + \bar{b}.X) \quad \text{and} \\
T_2 &\equiv \text{fix}(X = \tau.\bar{a}.\mathtt{nok}.\mathbf{0} + \tau.\bar{b}.X)
\end{aligned}$$

yields that the formula tested by both T_1 and T_2 is $\mathtt{max}(X, [a]\mathtt{ff} \wedge [b]X)$.

Collecting the results in Thms. 3.6 and 3.17, we have now finally completed the proof of Thm. 3.5. Thus, as claimed, the collection of testable properties coincides with that of the properties expressible in SHML. The following result gives another characterization of the expressive power of SHML which has some independent interest.

Theorem 3.19. *The property language* SHML *is the least expressive extension of* $\mathcal{L}_{\mathtt{nok}}$ *that is compositional with respect to tests and* $\|$.

Proof. Assume that $\mathcal{L}$ is a property language that extends $\mathcal{L}_{\mathtt{nok}}$ and is compositional. We show that every property in SHML is logically equivalent to one in $\mathcal{L}$, i.e., that $\mathcal{L}$ is at least as expressive as SHML. To this end, let φ be a property in SHML. By Thm. 3.6, there is a test T_φ such that $s \models \varphi$ iff s passes the test T_φ, for every state s. Since $\mathcal{L}$ is an extension of $\mathcal{L}_{\mathtt{nok}}$ that is compositional, Propn. 3.13 yields that $\mathcal{L}$ is complete. Thus there is a formula $\psi \in \mathcal{L}$ such that $s \models \psi$ iff s passes the test T_φ, for every state s. It follows that ψ and φ are satisfied by precisely the same states, and are therefore logically equivalent. $\square$

Acknowledgements: We thank Kim Guldstrand Larsen for previous joint work and discussions that gave us the inspiration for this study. The anonymous referees provided useful comments.

References

1. L. Aceto, P. Bouyer, A. Burgueño, and K. G. Larsen, *The power of reachability testing for timed automata*, in Proceedings of the Eighteenth Conference on the Foundations of Software Technology and Theoretical Computer Science, V. Arvind and R. Ramanujam, eds., Lecture Notes in Computer Science, Springer-Verlag, December 1998.
2. L. Aceto, A. Burgueño, and K. G. Larsen, *Model checking via reachability testing for timed automata*, in Proceedings of TACAS '98, Lisbon, B. Steffen, ed., vol. 1384 of Lecture Notes in Computer Science, Springer-Verlag, 1998, pp. 263–280.
3. H. R. Andersen, *Partial model checking (extended abstract)*, in Proceedings of the Tenth Annual IEEE Symposium on Logic in Computer Science, San Diego, California, 26–29 June 1995, IEEE Computer Society Press, pp. 398–407.
4. R. De Nicola and M. Hennessy, *Testing equivalences for processes*, Theoretical Comput. Sci., 34 (1984), pp. 83–133.
5. D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi, *On the temporal basis of fairness*, in Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, Jan. 1980, pp. 163–173.
6. M. Hennessy, *Algebraic Theory of Processes*, MIT Press, Cambridge, Massachusetts, 1988.
7. M. Hennessy and R. Milner, *Algebraic laws for nondeterminism and concurrency*, J. Assoc. Comput. Mach., 32 (1985), pp. 137–161.
8. R. Keller, *Formal verification of parallel programs*, Comm. ACM, 19 (1976), pp. 371–384.
9. S. Kleene, *Representation of events in nerve nets and finite automata*, in Automata Studies, C. Shannon and J. McCarthy, eds., Princeton University Press, 1956, pp. 3–41.
10. D. Kozen, *Results on the propositional mu-calculus*, Theoretical Comput. Sci., 27 (1983), pp. 333–354.
11. F. Laroussinie, K. G. Larsen, and C. Weise, *From timed automata to logic - and back*, in Mathematical Foundations of Computer Science 1995, 20th International Symposium, J. Wiedermann and P. Hájek, eds., vol. 969 of Lecture Notes in Computer Science, Prague, Czech Republic, 28 Aug.–1 Sept. 1995, Springer-Verlag, pp. 529–539.

12. K. G. Larsen and L. Xinxin, *Compositionality through an operational semantics of contexts*, Journal of Logic and Computation, 1 (1991), pp. 761–795.
13. R. Milner, *Communication and Concurrency*, Prentice-Hall International, Englewood Cliffs, 1989.
14. D. Park, *Concurrency and automata on infinite sequences*, in 5^{th} GI Conference, Karlsruhe, Germany, P. Deussen, ed., vol. 104 of Lecture Notes in Computer Science, Springer-Verlag, 1981, pp. 167–183.
15. A. Stoughton, *Substitution revisited*, Theoretical Comput. Sci., 59 (1988), pp. 317–325.
16. A. Tarski, *A lattice-theoretical fixpoint theorem and its applications*, Pacific Journal of Mathematics, 5 (1955), pp. 285–309.
17. M. Y. Vardi and P. Wolper, *Reasoning about infinite computations*, Information and Computation, 115 (1994), pp. 1–37.

Recent BRICS Report Series Publications

RS-98-50 **Luca Aceto and Anna Ingólfsdóttir.** ***Testing Hennessy-Milner Logic with Recursion.*** **December 1998. 15 pp. To appear in Thomas, editor,** ***Foundations of Software Science and Computation Structures: Second International Conference*****, FoSSaCS '99 Proceedings, LNCS, 1998.**

RS-98-49 **Luca Aceto, Willem Jan Fokkink, and Anna Ingólfsdóttir.** ***A Cook's Tour of Equational Axiomatizations for Prefix Iteration.*** **December 1998. 14 pp. Appears in Nivat, editor,** ***Foundations of Software Science and Computation Structures: First International Conference*****, FoSSaCS '98 Proceedings, LNCS 1378, 1998, pages 20–34.**

RS-98-48 **Luca Aceto, Patricia Bouyer, Augusto Burgueño, and Kim G. Larsen.** ***The Power of Reachability Testing for Timed Automata.*** **December 1998. 12 pp. Appears in Arvind and Ramanujam, editors,** ***Foundations of Software Technology and Theoretical Computer Science: 18th Conference*****, FST&TCS '98 Proceedings, LNCS 1530, 1998, pages 245–256.**

RS-98-47 **Gerd Behrmann, Kim G. Larsen, Justin Pearson, Carsten Weise, and Yi Wang.** ***Efficient Timed Reachability Analysis using Clock Difference Diagrams.*** **December 1998. 13 pp.**

RS-98-46 **Kim G. Larsen, Carsten Weise, Yi Wang, and Justin Pearson.** ***Clock Difference Diagrams.*** **December 1998. 18 pp.**

RS-98-45 **Morten Vadskær Jensen and Brian Nielsen.** ***Real-Time Layered Video Compression using SIMD Computation.*** **December 1998. 37 pp. Appears in Zinterhof, Vajtersic and Uhl, editors,** ***Parallel Computing: Fourth International ACPC Conference*****, ACPC '99 Proceedings, LNCS 1557, 1999.**

RS-98-44 **Brian Nielsen and Gul Agha.** ***Towards Re-usable Real-Time Objects.*** **December 1998. 36 pp. To appear in** ***The Annals of Software Engineering*****, IEEE, 7, 1999.**

RS-98-43 **Peter D. Mosses.** ***CASL: A Guided Tour of its Design.*** **December 1998. 31 pp. To appear in Fiadeiro, editor,** ***Recent Trends in Algebraic Development Techniques: 13th Workshop*****, WADT '98 Selected Papers, LNCS, 1999.**