A Semantic Theory for Heterogeneous System Design*

Rance Cleaveland[1] and Gerald Lüttgen[2]

[1] Department of Computer Science, State University of New York at Stony Brook, Stony Brook, New York 11794–4400, USA, rance@cs.sunysb.edu

[2] Department of Computer Science, Sheffield University, 211 Portobello Street, Sheffield S1 4DP, England, g.luettgen@dcs.shef.ac.uk

Abstract. This paper extends DeNicola and Hennessy's testing theory from labeled transition system to Büchi processes and establishes a tight connection between the resulting Büchi must–preorder and satisfaction of linear–time temporal logic (LTL) formulas. An example dealing with the design of a communications protocol testifies to the utility of the theory for heterogeneous system design, in which some components are specified as labeled transition systems and others are given as LTL formulas.

1 Introduction

Approaches to formally verifying reactive systems typically follow one of two paradigms. The first paradigm is founded on notions of *refinement* and is employed in process algebra [2]. In such approaches one formulates specifications and implementations in the same notation and then proves that the latter refine the former. The underling semantics is usually given operationally, and refinement relations are formalized as preorders. *Testing/failure preorders* [4, 8] have attracted particular attention because of their intuitive formulations in terms of responses a system exhibits to tests. Their strength is their support for *compositional reasoning*, i.e., one may refine part of a system design independently of others, and their *full abstractness* with respect to trace inclusion [18].

The other paradigm relies on the use of *temporal logics* [22] to formulate specifications, with implementations being given in an operational notation. One then verifies a system by establishing that it is a model of its specification; *model checkers* [5] automate this task for finite–state systems. Temporal logics support the definition of properties that constrain single aspects of expected system behavior and, thus, allow a "property–at–a–time" approach. Such logics also have connections with automata over infinite words. For example, *linear–time temporal logic* (LTL) specifications may be translated into *Büchi automata* [27] which allow semantic constraints on infinite behavior to be expressed.

* Research support was provided under NASA Contract No. NAS1–97046 and by NSF grant CCR–9988489. The first author was also supported by AFOSR Grant F49620–95–1–0508, ARO Grant P–38682–MA, and NSF Grants CCR–9505562, CCR–9996086, and INT–9996095.

The objective of this paper is to develop a semantic framework that seamlessly unifies testing–based refinement and LTL, thereby enabling the development of design formalisms that provide support for both styles of verification. Using Büchi automata and the testing framework of DeNicola and Hennessy [8] as starting points, we approach this task by developing *Büchi may–* and *must–preorders* that relate *Büchi processes* on the basis of their responses to *Büchi tests*. Alternative characterizations are provided and employed for proving conservative–extension results regarding DeNicola and Hennessy's testing theory. We then apply this framework to defining a semantics for heterogeneous design notations, where systems are specified using a mixture of labeled transition systems and LTL formulas. This is done in two steps: first, we show that our Büchi must–preorder is compositional for *parallel composition* and *scoping* operators that are inspired by CCS [19]. Second, we establish that the Büchi must–preorder reduces to a variant of reverse trace inclusion when its first argument is purely nondeterministic. Consequently, the Büchi must–preorder permits a uniform treatment of traditional notions of process refinement and LTL satisfaction. The utility of our new theory is illustrated by means of a small example featuring the heterogeneous design of a generic communications protocol.

2 Büchi Testing

We extend the *testing theory* of DeNicola and Hennessy [8], which was developed for labeled transition systems in a process–algebraic setting, to *Büchi automata*. Traditional testing relates labeled transition systems via two preorders, the *may–* and *must–preorders*, which distinguish systems on the basis of the tests they might be able to, or are necessarily able to, pass. Büchi automata generalize labeled transition systems by means of an acceptance condition for infinite traces. However, the classical Büchi semantics, which identifies automata having the same infinite languages, is in general not compositional with respect to parallel composition operators, since it is insensitive to the potential for deadlock. Our testing semantics is intended to overcome this problem. In the sequel, we refer to Büchi automata as *Büchi processes* to emphasize that we are equipping Büchi automata with a different semantics than the traditional one.

Basic Definitions. Our semantic framework is defined relative to some *alphabet* $\mathcal{A}$, i.e., a countable set of *actions* which does not include the distinguished *unobservable, internal action* τ. In the remainder, we let $a, b, \ldots$ range over $\mathcal{A}$ and $\alpha, \beta, \ldots$ over $\mathcal{A} \cup \{\tau\}$. Büchi processes are distinguished from labeled transition systems in their treatment of infinite traces. Whereas in labeled transition systems all infinite traces are typically deemed possible, in Büchi processes only those infinite traces that go through designated *Büchi states* infinitely often are considered actual executions.

Definition 1 (Büchi process). *A* Büchi process *is a tuple* $\langle P, \longrightarrow, \surd, p\rangle$*, where* P *is a countable set of* states*,* $\longrightarrow \subseteq P \times (\mathcal{A} \cup \{\tau\}) \times P$ *is the* transition relation*,* $\surd \subseteq S$ *is the* Büchi set*, and* $p \in P$ *is the* start state*. If* $\surd = P$ *we refer to the Büchi process as a* labeled transition system*.*

For convenience, we often write (i) $p' \stackrel{\alpha}{\longrightarrow} p''$ instead of $\langle p', \alpha, p'' \rangle \in \longrightarrow$, (ii) $p' \stackrel{\alpha}{\longrightarrow}$ for $\exists p'' \in P.\, p' \stackrel{\alpha}{\longrightarrow} p''$, (iii) $p' \longrightarrow$ for $\exists \alpha \in \mathcal{A} \cup \{\tau\}, p'' \in P.\ p' \stackrel{\alpha}{\longrightarrow} p''$, and (iv) $p' \surd$ for $p' \in \surd$. If no confusion arises, we abbreviate the Büchi process $\langle P, \longrightarrow, \surd, p \rangle$ by its start state p and refer to its transition relation and Büchi set as $\longrightarrow_p$ and $\surd_p$, respectively. Moreover, we denote the set of all Büchi processes by $\mathcal{P}$. Note that we do not require Büchi processes to be finite–state.

Definition 2 (Path & trace). *Let $\langle P, \longrightarrow, \surd, p \rangle$ be a Büchi process. A* path *π starting from state $p' \in P$ is a potentially infinite sequence $(\langle p_{i-1}, \alpha_i, p_i \rangle)_{0<i\leq k}$, where $k \in \mathbb{N} \cup \{\infty\}$, such that $k = 0$, or $p_0 = p'$ and $p_{i-1} \stackrel{\alpha_i}{\longrightarrow} p_i$, for all $0 < i \leq k$. We use $|\pi|$ to refer to k, the length of π. If $|\pi| = \infty$, we say that π is* infinite*; otherwise, π is* finite*. If $|\pi| \in \mathbb{N}$ and $p_{|\pi|} \not\longrightarrow$, i.e., $p_{|\pi|}$ is a* deadlock *state, path π is called* maximal*. Path π is referred to as a* Büchi path *if $|\pi| = \infty$ and $|\{i \in \mathbb{N} \,|\, p_i \surd\}| = \infty$. The (visible)* trace *$\mathsf{trace}(\pi)$ of π is defined as the sequence $(\alpha_i)_{i \in I_\pi} \in \mathcal{A}^* \cup \mathcal{A}^\infty$, where $I_\pi =_{df} \{0 < i \leq |\pi| \,|\, \alpha_i \neq \tau\}$.*

We denote the sets of all finite paths, all maximal paths, and all Büchi paths starting from state $p' \in P$ by $\Pi_{\mathsf{fin}}(p')$, $\Pi_{\mathsf{max}}(p')$, and $\Pi_{\mathsf{B}}(p')$, respectively. The empty path π with $|\pi| = 0$ is symbolized by $()$ and its trace by ϵ. We sometimes write α for the empty or single-element sequence trace (α) and use the notation $p' \stackrel{w}{\Longrightarrow}_p p''$ to indicate that state p' of Büchi process p may evolve to state p'' when observing trace w for some path $\pi \in \Pi_{\mathsf{fin}}(p')$. Formally, $p' \stackrel{w}{\Longrightarrow}_p p''$ if $\exists \pi = (\langle p_{i-1}, \alpha_i, p_i \rangle)_{0<i\leq k} \in \Pi_{\mathsf{fin}}(p).\, p_0 = p',\ p_k = p''$, and $\mathsf{trace}(\pi) = w$. Moreover, $\mathcal{I}_p(p') =_{\mathrm{df}} \{a \in \mathcal{A} \,|\, \exists p''.\, p' \stackrel{a}{\Longrightarrow}_p p''\}$ is the set of initial actions of p in state $p' \in P$. We may also introduce different languages for Büchi process p.

$$
\begin{array}{llr}
\mathcal{L}_{\mathsf{fin}}(p) =_{\mathrm{df}} \{\mathsf{trace}(\pi) \,|\, \pi \in \Pi_{\mathsf{fin}}(p)\} & \subseteq \mathcal{A}^* & \textit{finite–trace language of } p \\
\mathcal{L}_{\mathsf{max}}(p) =_{\mathrm{df}} \{\mathsf{trace}(\pi) \,|\, \pi \in \Pi_{\mathsf{max}}(p)\} & \subseteq \mathcal{A}^* & \textit{maximal–trace language of } p \\
\mathcal{L}_{\mathsf{B}}(p) =_{\mathrm{df}} \{\mathsf{trace}(\pi) \,|\, \pi \in \Pi_{\mathsf{B}}(p)\} & \subseteq \mathcal{A}^* \cup \mathcal{A}^\infty & \textit{Büchi–trace language of } p
\end{array}
$$

A key notion in testing–based semantics is *divergence*, i.e., a system's ability to engage in an infinite internal computation. In this paper, we use adaptations of the traditional notions of DeNicola and Hennessy [8]; more sophisticated definitions may be found elsewhere in the literature [3, 21, 23] but are not considered here. We say that state p' of Büchi process p is (*Büchi*) *divergent*, in symbols $p' \Uparrow_p$, if $\exists \pi \in \Pi_{\mathsf{B}}(p').\ \mathsf{trace}(\pi) = \epsilon$. State p' is called *w–divergent* for some $w = (a_i)_{0<i\leq k} \in \mathcal{A}^* \cup \mathcal{A}^\infty$, in symbols $p' \Uparrow_p w$, if one can reach a divergent state starting from p' when executing a finite prefix of w, i.e., if $\exists l \leq k,\ p'' \in P, w' \in \mathcal{A}^*.\ w' = (a_i)_{0<i\leq l}, p' \stackrel{w'}{\Longrightarrow} p''$, and $p'' \Uparrow_p$. For convenience, we write $\mathcal{L}_{\mathsf{div}}(p')$ for the *divergent–trace language* of p', i.e., $\mathcal{L}_{\mathsf{div}}(p') =_{\mathrm{df}} \{w \in \mathcal{A}^* \cup \mathcal{A}^\infty \,|\, p' \Uparrow_p w\}$. State p' is convergent or w–convergent, in symbols $p' \Downarrow_p$ and $p' \Downarrow_p w$, if not $p' \Uparrow_p$ and not $p' \Uparrow_p w$, respectively. Note that a finite trace $w \in \mathcal{L}_{\mathsf{B}}(p)$ indicates that p is divergent exactly after executing w. In the following, we often omit the indices of the divergence and convergence predicates, as well as of the transition relations, whenever these are obvious from the context. Finally, we write $w \cdot w'$ for the *concatenation* of finite trace $w \in \mathcal{A}^*$ with the finite or infinite trace $w' \in \mathcal{A}^* \cup \mathcal{A}^\infty$.

Testing Theory. The testing framework of DeNicola and Hennessy defines behavioral preorders that relate labeled transition systems with respect to their responses to *tests* [8]. Tests are employed to witness the interactions a system may have with its environment. In our setting, a test is a Büchi process in which certain states are designated as *successful*. In order to determine whether a system passes a test, one has to examine the finite and infinite *computations* that result when the test runs in lock–step with the system under consideration.

Definition 3 (Test, computation, success). *A Büchi test $\langle T, \longrightarrow, \surd, t, \mathsf{Suc}\rangle$ is a Büchi process $\langle T, \longrightarrow, \surd, t\rangle$ together with a set $\mathsf{Suc} \subseteq T$ of* success states*. If $\surd = \emptyset$, we call the test* classical*. The set of all Büchi tests is denoted by $\mathcal{T}$.*

A potential computation *c with respect to a Büchi process p and a Büchi test t is a potentially infinite sequence $(\langle p_{i-1}, t_{i-1}\rangle \xmapsto{\alpha_i}_{r_i} \langle p_i, t_i\rangle)_{0<i\leq k}$, where $k \in \mathbb{N} \cup \{\infty\}$, such that (1) $p_i \in P$ and $t_i \in T$, for all $0 \leq i \leq k$, and (2) $\alpha_i \in \mathcal{A} \cup \{\tau\}$ and $r_i \in \{\blacktriangleleft, \blacktriangleright, \blacklozenge\}$, for all $0 < i \leq k$. The relation $\longmapsto$ is defined by:*

- $\langle p_{i-1}, t_{i-1}\rangle \xmapsto{\alpha_i}_{\blacktriangleleft} \langle p_i, t_i\rangle$ *if* $\alpha_i = \tau,\ t_{i-1} = t_i, \quad p_{i-1} \xrightarrow{\tau}_p p_i,\ \&\ t_{i-1} \notin \mathsf{Suc}$.
- $\langle p_{i-1}, t_{i-1}\rangle \xmapsto{\alpha_i}_{\blacktriangleright} \langle p_i, t_i\rangle$ *if* $\alpha_i = \tau,\ p_{i-1} = p_i, \quad t_{i-1} \xrightarrow{\tau}_t t_i,\ \&\ t_{i-1} \notin \mathsf{Suc}$.
- $\langle p_{i-1}, t_{i-1}\rangle \xmapsto{\alpha_i}_{\blacklozenge} \langle p_i, t_i\rangle$ *if* $\alpha_i \in \mathcal{A},\ p_{i-1} \xrightarrow{\alpha_i}_p p_i,\ t_{i-1} \xrightarrow{\alpha_i}_t t_i,\ \&\ t_{i-1} \notin \mathsf{Suc}$.

The potential computation c is finite *if $|c| < \infty$ and* infinite *if $|c| = \infty$. The projection $\mathsf{proj}_p(c)$ of c on p is defined as $(\langle p_{i-1}, \alpha_i, p_i\rangle)_{i \in I_p^c} \in \Pi(p)$, where $I_p^c =_{df} \{0 < i \leq k \mid r_i \in \{\blacktriangleleft, \blacklozenge\}\}$, and the projection $\mathsf{proj}_t(c)$ of c on t as $(\langle t_{i-1}, \alpha_i, t_i\rangle)_{i \in I_t^c} \in \Pi(p)$, where $I_t^c =_{df} \{0 < i \leq k \mid r_i \in \{\blacktriangleright, \blacklozenge\}\}$. A potential computation c is called a* computation *if it satisfies the following properties: (1) c is maximal, i.e. if $|c| < \infty$ then $\langle p_{|c|}, t_{|c|}\rangle \not\xmapsto{\alpha}_r$ for any α and r; and (2) if $|c| = \infty$ then $\mathsf{proj}_p(c) \in \Pi_B(p)$. The set of all computations of p and t is denoted by $\mathcal{C}(p,t)$.*

Computation c is called successful *if $t_{|c|} \in \mathsf{Suc}$, in case $|c| < \infty$, or if $\mathsf{proj}_t(c) \in \Pi_B(t)$, in case $|c| = \infty$. We say that p* may pass *t, in symbols $p\ \mathsf{may}_{\mathrm{CL}}\ t$, if there exists a successful computation $c \in \mathcal{C}(p,t)$. Analogously, p* must pass *t, in symbols $p\ \mathsf{must}_{\mathrm{CL}}\ t$, if every computation $c \in \mathcal{C}(p,t)$ is successful.*

Intuitively, an infinite computation of process p and test t differs from an infinite potential computation in that in the former the process is required to enter a Büchi state infinitely often. An infinite computation is then successful if the test also passes through a Büchi state infinitely often. Hence, in contrast with the original theory of DeNicola and Hennessy, some infinite computations can be successful in our setting. Also, since Büchi processes and Büchi tests potentially exhibit nondeterministic behavior, one may distinguish between the *possibility* and *inevitability* of success. This is captured in the following definitions of the Büchi *may*– and *must*–preorders.

Definition 4 (Büchi Preorders). *For Büchi processes p and q we define:*

- $p \sqsubseteq_{\mathrm{CL}}^{\mathsf{may}} q$ *if* $\forall t \in \mathcal{T}.\ p\ \mathsf{may}_{\mathrm{CL}}\ t$ *implies* $q\ \mathsf{may}_{\mathrm{CL}}\ t$.
- $p \sqsubseteq_{\mathrm{CL}}^{\mathsf{must}} q$ *if* $\forall t \in \mathcal{T}.\ p\ \mathsf{must}_{\mathrm{CL}}\ t$ *implies* $q\ \mathsf{must}_{\mathrm{CL}}\ t$.

It is easy to check that $\sqsubseteq_{\mathrm{CL}}^{\mathsf{may}}$ and $\sqsubseteq_{\mathrm{CL}}^{\mathsf{must}}$ are preorders. The classical may– and must–preorders of DeNicola and Hennessy are defined analogously, but with respect to transition systems and classical tests [8]. Note that in this paper we consider the Büchi may–preorder only for the sake of completing the Büchi testing theory; it is not used in our semantic framework for heterogeneous system specification.

3 Alternative Characterizations and Conservativity

We now present characterizations of our Büchi preorders and use these characterizations as a basis for comparing DeNicola-Hennessy testing theory [8] ours.

Theorem 1. *Let p and q be Büchi processes. Then*

1. $p \sqsubseteq_{\mathrm{CL}}^{\mathit{may}} q$ *if and only if* $\mathcal{L}_{\mathit{fin}}(p) \subseteq \mathcal{L}_{\mathit{fin}}(q)$ *and* $\mathcal{L}_{B}(p) \subseteq \mathcal{L}_{B}(q)$.
2. $p \sqsubseteq_{\mathrm{CL}}^{\mathit{must}} q$ *if and only if for all* $w \in \mathcal{A}^* \cup \mathcal{A}^\infty$ *such that* $p \Downarrow w$ *:*
   (a) $q \Downarrow w$
   (b) $\underline{|w| < \infty}$*:* $\forall q'.\, q \stackrel{w}{\Longrightarrow} q'$ *implies* $\exists p'.\, p \stackrel{w}{\Longrightarrow} p'$ *and* $\mathcal{I}_p(p') \subseteq \mathcal{I}_q(q')$.
   $\underline{|w| = \infty}$*:* $w \in \mathcal{L}_B(q)$ *implies* $w \in \mathcal{L}_B(p)$.

With respect to finite traces, the characterizations are virtually the same as the ones of DeNicola and Hennessy's preorders [8]. However, we need to refine the classical characterizations in order to capture the sensitivity of Büchi may– and must–testing to infinite behavior. The proof of this theorem relies on the properties of several specific Büchi tests. Some of them are standard [8]; the other ones are depicted in Fig. 1, where (i) $w = (a_i)_{0<i\leq k} \in \mathcal{A}^*$ for tests $t_w^{\mathsf{may,div}}$ and $t_w^{\mathsf{must,max}}$ and (ii) $w = (a_i)_{i\in\mathbb{N}} \in \mathcal{A}^\infty$ for tests $t_w^{\mathsf{may},\infty}$, $t_w^{\Downarrow}$, and $t_w^{\mathsf{must},\infty}$. In Fig. 1, Büchi states are marked by the symbol $\surd$, and success states are distinguished by thick borders.

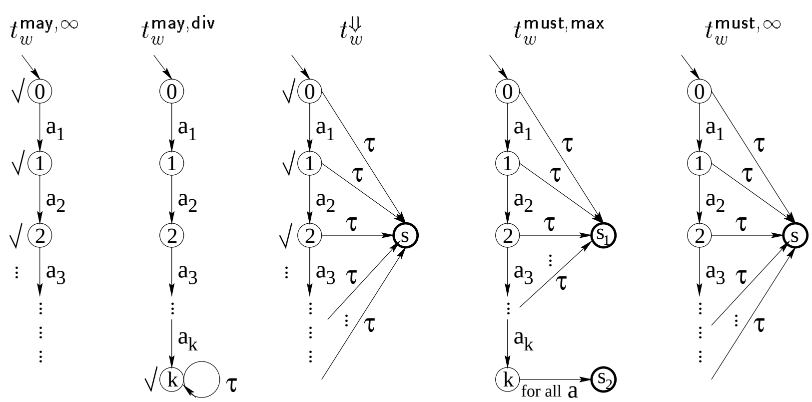


Fig. 1. Büchi tests used for characterizing the Büchi may– and must–preorders.

Intuitively, while Büchi test $t_w^{\mathsf{may},\infty}$ tests for the presence of Büchi trace w, Büchi tests $t_w^{\mathsf{may},\mathsf{div}}$ and $t_w^{\Downarrow}$ are capable of detecting divergent behavior when executing trace w. Moreover, Büchi tests $t_w^{\mathsf{must},\mathsf{max}}$ and $t_w^{\mathsf{must},\infty}$ are concerned with the absence of maximal trace and Büchi trace w, respectively. These intuitions are made precise by the following properties, which hold for any Büchi process p.

1. Let $w \in \mathcal{A}^\infty$. Then $w \in \mathcal{L}_{\mathsf{B}}(p)$ if and only if $p\,\mathsf{may}_{\mathrm{CL}}\, t_w^{\mathsf{may},\infty}$.
2. Let $w \in \mathcal{A}^*$. Then $w \in \mathcal{L}_{\mathsf{B}}(p)$ if and only if $p\,\mathsf{may}_{\mathrm{CL}}\, t_w^{\mathsf{may},\mathsf{div}}$.
3. Let $w \in \mathcal{A}^\infty$. Then $p \Downarrow w$ if and only if $p\,\mathsf{must}_{\mathrm{CL}}\, t_w^{\Downarrow}$.
4. Let $w \in \mathcal{A}^*$ s.t. $p \Downarrow w$. Then $w \notin \mathcal{L}_{\mathsf{max}}(p)$ if and only if $p\,\mathsf{must}_{\mathrm{CL}}\, t_w^{\mathsf{must},\mathsf{max}}$.
5. Let $w \in \mathcal{A}^\infty$ s.t. $p \Downarrow w$. Then $w \notin \mathcal{L}_{\mathsf{B}}(p)$ if and only if $p\,\mathsf{must}_{\mathrm{CL}}\, t_w^{\mathsf{must},\infty}$.

The proof of Thm. 1, which can be found in [7], relies on these properties of Büchi tests. Specifically, it uses the *infinite–state* tests $t_w^{\mathsf{may},\infty}$, $t_w^{\Downarrow}$, and $t_w^{\mathsf{must},\infty}$. The employment of infinite–state tests — even when relating finite–state Büchi processes — is justified by our view that Büchi tests represent the arbitrary, potentially irregular behavior of the unknown system environment.

Using the above characterizations, we investigate the relation of our Büchi preorders to the corresponding classical preorders, $\sqsubseteq_{\mathrm{DH}}^{\mathsf{may}}$ and $\sqsubseteq_{\mathrm{DH}}^{\mathsf{must}}$, respectively, as defined by DeNicola and Hennessy [8]. It should be noted that their framework is restricted to *image–finite* labeled transition systems and classical, image–finite tests; a labeled transition system or Büchi process is called image–finite if every state has only a finite number of outgoing transitions for any action.

Theorem 2. *Let p and q be image–finite labeled transition systems.*

1. *If p and q are convergent, then $p \sqsubseteq_{\mathrm{CL}}^{\mathsf{may}} q$ if and only if $p \sqsubseteq_{\mathrm{DH}}^{\mathsf{may}} q$.*
2. *$p \sqsubseteq_{\mathrm{CL}}^{\mathsf{must}} q$ if and only if $p \sqsubseteq_{\mathrm{DH}}^{\mathsf{must}} q$.*

We refer the reader to [7] for the proof of this theorem. In a nutshell, the second part follows by inspection of the alternative characterizations of $\sqsubseteq_{\mathrm{CL}}^{\mathsf{must}}$ and $\sqsubseteq_{\mathrm{DH}}^{\mathsf{must}}$. Thm. 2(1) is invalid if one allows *divergent* labeled transition systems. As a counterexample consider the transition systems $\langle \{p\}, \{\langle p, \tau, p\rangle\}, \{p\}, p\rangle$ and $\langle \{q\}, \emptyset, \{q\}, q\rangle$, as well as the Büchi test $\langle \{t\}, \{\langle t, \tau, t\rangle\}, \{t\}, t, \emptyset\rangle$. Then, $p \sqsubseteq_{\mathrm{DH}}^{\mathsf{may}} q$ since $\mathcal{L}_{\mathsf{fin}}(p) = \mathcal{L}_{\mathsf{fin}}(q) = \{\epsilon\}$, but $p \not\sqsubseteq_{\mathrm{CL}}^{\mathsf{may}} q$ since $p\,\mathsf{may}_{\mathrm{CL}}\, t$ and $q\,\not{\mathsf{may}}_{\mathrm{CL}}\, t$.

4 Büchi Testing and Heterogeneous System Design

In this section we investigate the utility of our theory as a semantic framework for heterogeneous design notations that mix labeled transition systems and formulas in LTL. The design methodology which we wish to support is *component–based*, where a system designer starts off with a system architecture, with components given either as automata or, more abstractly, as LTL formulas. Then the system is refined by successively implementing each component as a labeled transition system satisfying its specification. To support such a methodology mathematically, one needs a *refinement preorder* which satisfies at least two properties.

First, it must be compositional for key operators of such design languages. Second, it must be "compatible" with the LTL satisfaction relation. We show that our Büchi must–preorder obeys both properties.

Büchi Testing and Compositionality. In the component–based design framework we wish to study, two operators are central: (i) *parallel composition*, for connecting concurrent components and allowing them to interact via system channels, and (ii) *restriction*, for restricting access to channels to certain system components. In the following, we introduce two such operators that allow us to give the reader hints about the application of the semantic theory developed so far. While other operators are of course possible, the ones considered here suffice for the purposes of the example in the next section.

Our parallel composition operator "$|$" and the restriction operator $\backslash A$, where $A \subseteq \mathcal{A}$, are inspired by the ones in the process algebra CCS [19]. We assume that alphabet $\mathcal{A}$ is composed of two sets $\mathcal{A}!$ and $\mathcal{A}?$, representing *sending* and *receiving actions*, such that for every $a! \in \mathcal{A}!$ there exists a corresponding $a? \in \mathcal{A}?$, and vice versa. Here, a should be interpreted as a *channel name*. The intuition for parallel composition in CCS is that a process willing to send a message on channel a and another one able to receive a message on a can do so by performing the actions $a!$ and $a?$ in synchrony with each other. This *handshake* is invisible to an external observer, i.e., it results in the distinguished, unobservable action τ. When adapting the CCS parallel operator to our framework of Büchi processes, the question that naturally arises concerns the interpretation of Büchi traces. We adopt the following point of view: intuitively, "fair merges" of Büchi traces of p and q should also be Büchi traces of $p|q$. Moreover, a Büchi trace of one process, when merged with a finite trace of the other process, should also result in a Büchi trace of $p|q$.

Formally, our parallel composition of Büchi processes $\langle P, \longrightarrow_p, \surd_p, p\rangle$ and $\langle Q, \longrightarrow_q, \surd_q, q\rangle$ is defined as the Büchi process $\langle P|Q, \longrightarrow_{p|q}, \surd_{p|q}, p|q\rangle$, where $P|Q =_{\mathrm{df}} \{p'|q' \mid p' \in P, q' \in Q\} \cup \{q'|p' \mid p' \in P, q' \in Q\}$ and where $\longrightarrow_{p|q}$ is the least relation such that:

(1)	$p' \xrightarrow{\alpha}_p p''$	implies	$p'\|q' \xrightarrow{\alpha}_{p\|q} q'\|p''$	if $p'\surd_p$
(2)	$p' \xrightarrow{\alpha}_p p''$	implies	$p'\|q' \xrightarrow{\alpha}_{p\|q} p''\|q'$	if not $p'\surd_p$
(3)	$q' \xrightarrow{\alpha}_q q''$	implies	$p'\|q' \xrightarrow{\alpha}_{p\|q} q''\|p'$	
(4)	$p' \xrightarrow{a!}_p p''$ and $q' \xrightarrow{a?}_q q''$	implies	$p'\|q' \xrightarrow{\tau}_{p\|q} q''\|p''$	if $p'\surd_p$
(5)	$p' \xrightarrow{a!}_p p''$ and $q' \xrightarrow{a?}_q q''$	implies	$p'\|q' \xrightarrow{\tau}_{p\|q} p''\|q''$	if not $p'\surd_p$
(6)	$p' \xrightarrow{a?}_p p''$ and $q' \xrightarrow{a!}_q q''$	implies	$p'\|q' \xrightarrow{\tau}_{p\|q} q''\|p''$	if $p'\surd_p$
(7)	$p' \xrightarrow{a?}_p p''$ and $q' \xrightarrow{a!}_q q''$	implies	$p'\|q' \xrightarrow{\tau}_{p\|q} p''\|q''$	if not $p'\surd_p$

These rules are in accordance with our above–mentioned intuition of system behavior. The "switching" of the states of p and q in Rules (1), (3), (4), and (6) allows us to fairly merge "Büchi traces with Büchi traces" and "Büchi traces with finite traces" of the argument Büchi processes. Finally, the Büchi predicate $\surd_{p|q}$ is defined by $p'|q'\surd_{p|q}$ if $p'\surd_p$, for any $p' \in P$ and $q' \in Q$. The unary *restriction* operator $\backslash A$, for $A \subseteq \mathcal{A}$, essentially is a *scoping mechanism* on channel names.

Intuitively, $p \setminus A$ is defined as the Büchi process p, except that all transitions labeled by actions $a!$ and $a?$, where $a \in A$, are eliminated.

By referring to the characterizations of the Büchi may– and must–preorders one can establish the desired compositionality results: the Büchi may– and must–preorders are substitutive under parallel composition and restriction.

Büchi Must–testing and LTL Satisfaction. We now show that the Büchi must–preorder is compatible with the LTL satisfaction relation $\models$, which relates labeled transition systems and LTL formulas [22]. By "compatible" we mean that, for every LTL formula ϕ, there exists a Büchi process B_ϕ such that the following holds for any labeled transition system p: $p \models \phi$ if and only if $B_\phi \sqsubseteq^{\mathsf{must}}_{\mathrm{CL}} p$, i.e., the 'implementation' p *refines* the 'specification' ϕ.

To achieve this goal, we characterize the Büchi must–preorder for a certain class of Büchi processes by means of trace inclusion. We call a Büchi process p *purely nondeterministic*, if for all $p' \in P$: (i) $p' \xrightarrow{\tau}_p$ implies $p' \not\xrightarrow{a}_p$, for all $a \in \mathcal{A}$, and (ii) $|\{\langle a, p'' \rangle \in \mathcal{A} \times P \,|\, p' \xrightarrow{a}_p p''\}| = 1$. Note that a Büchi process p can be transformed to a purely nondeterministic Büchi process p', such that $\mathcal{L}_{\mathsf{div}}(p) = \mathcal{L}_{\mathsf{div}}(p')$, $\mathcal{L}_{\mathsf{fin}}(p) = \mathcal{L}_{\mathsf{fin}}(p')$, $\mathcal{L}_{\mathsf{max}}(p) = \mathcal{L}_{\mathsf{max}}(p')$, and $\mathcal{L}_{\mathsf{B}}(p) = \mathcal{L}_{\mathsf{B}}(p')$, by splitting every transition $p' \xrightarrow{a}_p p''$ into two transitions $p' \xrightarrow{\tau}_p p_{\langle p',a,p''\rangle} \xrightarrow{a}_p p''$, where $p_{\langle p',a,p''\rangle} \notin P$ is a new, distinguished state.

Theorem 3. *Let p and q be Büchi processes and p be purely nondeterministic. Then, $p \sqsubseteq^{\mathsf{must}}_{\mathrm{CL}} q$ if and only if (i) $\mathcal{L}_{\mathsf{div}}(q) \subseteq \mathcal{L}_{\mathsf{div}}(p)$, (ii) $\mathcal{L}_{\mathsf{fin}}(q) \setminus \mathcal{L}_{\mathsf{div}}(p) \subseteq \mathcal{L}_{\mathsf{fin}}(p)$, (iii) $\mathcal{L}_{\mathsf{max}}(q) \setminus \mathcal{L}_{\mathsf{div}}(p) \subseteq \mathcal{L}_{\mathsf{max}}(p)$, and (iv) $\mathcal{L}_{\mathsf{B}}(q) \setminus \mathcal{L}_{\mathsf{div}}(p) \subseteq \mathcal{L}_{\mathsf{B}}(p)$.*

The necessity of the premise of this theorem, whose proof is in [7], may be demonstrated by Büchi processes $p =_{\mathrm{df}} \langle \{p_1, p_2\}, \{\langle p_1, a, p_1\rangle, \langle p_1, b, p_2\rangle\}, \emptyset, p_1\rangle$ and $q =_{\mathrm{df}} \langle \{q_1, q_2\}, \{\langle q_1, b, q_2\rangle\}, \emptyset, q_1\rangle$. Then p is *not* purely nondeterministic and Inclusions (i)–(iv) obviously hold, but $p \not\sqsubseteq^{\mathsf{must}}_{\mathrm{CL}} q$ since $p \;\mathsf{must}_{\mathrm{CL}}\; t$ and $q \;\not\mathsf{must}_{\mathrm{CL}}\; t$, for the Büchi test $t =_{\mathrm{df}} \langle \{t_1, t_2\}, \{\langle t_1, a, t_2\rangle\}, \emptyset, t_1, \{t_2\}\rangle$.

The above theorem is the key for establishing the desired connection between the Büchi must–preorder $\sqsubseteq^{\mathsf{must}}_{\mathrm{CL}}$ and the satisfaction relation $\models$ for LTL. In particular, well–known constructions — starting with the seminal work of Vardi and Wolper [27] — exist for converting LTL formulas into Büchi automata whose languages consist precisely of the models of the corresponding formulas. These constructions may be adapted to yield purely nondeterministic Büchi processes. However, there are a few subtleties of our setting compared to the traditional one on which we need to comment. First of all, our framework is concerned with labeled transition systems, so we must be able to interpret LTL formulas with respect to sequences of actions rather than states. Also, our framework is not only concerned with Büchi traces but also with finite traces (i.e., deadlocks) and divergent traces. The syntax and semantics of LTL may be modified to cope with these new phenomena; the details are not difficult and are omitted. The classical constructions of Büchi automata from LTL formulas may then be adapted to cope with the modifications to the logic. Whereas the adaptation for deadlock is well–known [17], the handling of divergence requires some attention. Intuitively, in a Büchi process a divergent state may engage in arbitrary behavior; this is reflected in its divergence language, which is $\mathcal{A}^* \cup \mathcal{A}^\infty$ (cf. Sec. 2). The only LTL formulas satisfied by arbitrary behavior are tautologies. Hence, in the Büchi process construction for LTL formulas, every state which corresponds to a tautology needs to be made divergent. Having these twists in mind, one may obtain the following variant of the key theorem for automata–based LTL model checking (cf. [27]), where $\hat{B}_\phi$ denotes the Büchi process constructed for LTL formula ϕ.

Theorem 4. *Let p be a labeled transition system and ϕ be an* LTL *formula. Then, $p \models \phi$ if and only if Inclusions (i)–(iv) in Thm. 3 hold for $\hat{B}_\phi$ and p (i.e., replace p in Inclusions (i)–(iv) by $\hat{B}_\phi$ and q by p).*

Note that the "$\Longrightarrow$"–direction of Thm. 4 is invalid if p is allowed to be an arbitrary Büchi process rather than a labeled transition system. As a counterexample consider $p =_{\text{df}} \langle \{p_1, p_2, p_3\}, \{\langle p_1, a, p_2\rangle, \langle p_1, b, p_3\rangle, \langle p_3, b, p_3\rangle\}, \emptyset, p_1\rangle$ and $\phi =_{\text{df}} a$. Then $p \models a$ as $b^\infty \notin \mathcal{L}_{\mathsf{B}}(p)$ and $b \in \mathcal{L}_{\mathsf{fin}}(p) \setminus \mathcal{L}_{\mathsf{div}}(\hat{B}_\phi)$. But obviously $b \notin \mathcal{L}_{\mathsf{fin}}(\hat{B}_\phi)$. When transforming $\hat{B}_\phi$ to a purely nondeterministic Büchi process B_ϕ as outlined above, we may combine Thms. 3 and 4 to obtain our desired result.

Corollary 1 (Büchi must–testing and LTL). *Let p be a labeled transition system and ϕ be an* LTL *formula. Then, $p \models \phi$ if and only if $B_\phi \sqsubseteq^{\mathit{must}}_{\text{CL}} p$.*

Hence, our notion of Büchi must–testing not only extends DeNicola and Hennessy's [8] and Narayan Kumar et al.'s [20] must–preorders to (arbitrary) Büchi processes, but is also compatible with the LTL satisfaction relation.

5 Example

As an example for the utility of our theory for heterogeneous system design, consider the design of a very simple communications protocol given in Fig. 2.

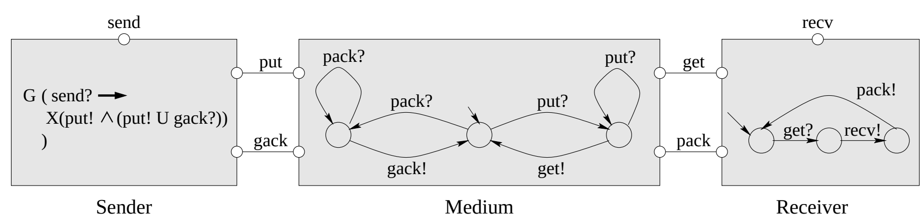


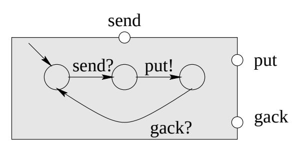
Fig. 2. A simple communications protocol.

The architecture of the protocol has already been fixed by the system designer and consists of a sender `Sender`, a medium `Medium`, and a receiver `Receiver`. The components communicate with the protocol's environment and among themselves via *channels*. In case of component `Sender`, these are the channels `send`, `put`, and `gack` (*get acknowledgment*). Each component in turn has its own specification. `Receiver` and `Medium` are given as labeled transition systems, reflecting the fact that their designs are relatively advanced. `Sender`, in contrast, is specified as an LTL formula stating that whenever a `send?` action occurs during an execution sequence of the sender, the remainder of the execution must begin with a sequence of `put!` actions followed by a `gack?` action. Finally, the overall specification of the protocol's required behavior may be given by the LTL formula $\texttt{Spec} =_{\text{df}} \texttt{G (send?} \rightarrow \texttt{(F recv!))}$. This formula dictates that in any sequence of actions which the system performs, whenever a `send?` action occurs, a `recv!` action eventually follows. An obvious question that a designer would be interested in is whether the specification of the sender is "strong enough" to ensure that the protocol satisfies `Spec`. The theory developed in this paper provides the semantic framework for answering this question. To do so, we first construct the purely nondeterministic Büchi process $B_{\texttt{Spec}}$ for LTL formula `Spec`, as well as Büchi process $B_{\texttt{Sender}}$ for LTL formula $\phi_{\texttt{sender}}$. Next we assemble the overall system by employing our parallel composition and restriction operators.

$$\texttt{System} =_{\text{df}} (B_{\texttt{Sender}} \,|\, \texttt{Medium} \,|\, \texttt{Receiver}) \setminus \{\texttt{put}, \texttt{get}, \texttt{pack}, \texttt{gack}\}$$

Finally, we determine whether or not $B_{\texttt{Spec}} \sqsubseteq^{\mathsf{must}}_{\text{CL}} \texttt{System}$; this indeed holds.

The development of an efficient algorithm for automatically determining whether two Büchi processes are related by $\sqsubseteq^{\mathsf{must}}_{\text{CL}}$ is future work. However, the alternative characterization of $\sqsubseteq^{\mathsf{must}}_{\text{CL}}$ (cf. Thm. 1) already provides some hints about how this can be done. Due to the compositionality of the Büchi must–preorder, our positive answer is preserved when replacing $B_{\texttt{Sender}}$ with any Büchi process p such that $B_{\texttt{Sender}} \sqsubseteq^{\mathsf{must}}_{\text{CL}} p$. If p is a labeled transition system and $B_{\texttt{Sender}}$ is made purely nondeterministic, then $B_{\texttt{Sender}} \sqsubseteq^{\mathsf{must}}_{\text{CL}} p$ holds exactly when $p \models \phi_{\texttt{Sender}}$, according to Cor. 1. One such p is depicted to the right.



6 Related Work

Starting with the same motivation we did, Abadi and Lamport have developed ideas for heterogeneous specification for shared–memory systems [1]. Their technical setting is the logical framework of TLA [15], in which processes and temporal formulas are indistinguishable and logical implication serves as the refinement relation. TLA refinement coincides in some sense with trace inclusion in our testing scenario and is therefore insensitive to deadlock and divergence. Such issues are not of concern in the shared–memory world but must be dealt with in our setting, which is targeted towards specifying *distributed* systems in which components can interact directly, rather than indirectly via the shared memory.

Of direct relevance to this paper is the work of Kurshan [14], who developed a theory of ω*–word automata* that includes notions of synchronous and asynchronous composition. However, his underlying semantic model maps processes to their *maximal (infinite) traces*, and the associated notion of refinement is (reverse) trace inclusion. In theories of concurrency such as CCS [19] and CSP [4], in which deadlock is possible, maximal trace inclusion is not compositional [18]. In contrast, our must–preorder is compositional, at least for the operators presented here. Other work [6, 10, 13] investigated *modular* and *compositional model–checking* in similar non–deadlock environments.

Relatively more work has been devoted to analyzing relationships *between* refinement and logical approaches. One line of study relates temporal–logic specifications to refinement–based ones by establishing that one system refines another if and only if both satisfy the same properties. Results along these lines were pioneered by Hennessy and Milner [11] for *bisimulation* equivalence and a modal logic of their devising [19]. Similar ideas were also adapted regarding other behavioral equivalences and preorders and other temporal logics [9, 19, 25]. Congruences preserving "next-time–less" LTL have been studied by Kaivola and Valmari in [12]; the results have subsequently been extended to handle deadlock [26] and livelock [23]. Our work differs from theirs in that we want to have LTL formulas *embedded* in specifications.

Another line of research involves the encoding of labeled transition systems as logical formulas, and vice versa. Steffen and Ingólfsdóttir [24] defined an algorithm for converting finite–state labeled transition systems into formulas in the *mu–calculus*, while Larsen [16] demonstrated that certain mu–calculus formulas can be encoded as bisimulation–based *implicit specifications*. Finally, traditional testing has also been enriched with notions of *fairness* [3, 21] in order to constrain infinite computations in labeled transition systems.

7 Conclusions and Future Work

We conservatively extended the testing theories of DeNicola and Hennessy [8] and Narayan Kumar et al. [20] to Büchi processes. We then studied the derived Büchi may– and must–preorders, developed alternative characterizations for them, argued that the preorders are substitutive for several operators necessary for component–based system design, and showed that the Büchi must–preorder degrades to a variant of reverse trace inclusion when its first argument is purely nondeterministic. Using the latter result, we illustrated that Büchi must–testing provides a uniform basis for analyzing heterogeneous system designs given as a mixture of labeled transition systems and LTL formulas.

Regarding future work, we plan to develop specification languages mixing process algebras and LTL, which are given a semantics in terms of Büchi testing. We also intend to explore algorithms for computing our Büchi must–preorder.

References

[1] M. Abadi and L. Lamport. Composing specifications. *TOPLAS*, 15(1):73–132, 1993. See also: Conjoining Specifications, *TOPLAS*, 17(3):507–534, 1995.

[2] J.A. Bergstra, A. Ponse, and S.A. Smolka. *Handbook of Process Algebra*. Elsevier Science, 2000.

[3] E. Brinksma, A. Rensink, and W. Vogler. Fair testing. In *CONCUR '95*, volume 962 of *LNCS*, pages 313–328. Springer-Verlag, 1995.

[4] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *J. of the ACM*, 31(3):560–599, 1984.

[5] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[6] E.M. Clarke, D.E. Long, and K.L. McMillan. Compositional model checking. In *LICS '89*, pages 353–362. IEEE Computer Society Press, 1989.

[7] R. Cleaveland and G. Lüttgen. Model checking is refinement: Relating Büchi testing and linear-time temporal logic. Technical Report 2000-14, Institute for Computer Applications in Science and Engineering, March 2000.

[8] R. DeNicola and M.C.B. Hennessy. Testing equivalences for processes. *TCS*, 34:83–133, 1983.

[9] R. DeNicola and F. Vaandrager. Three logics for branching bisimulation. *J. of the ACM*, 42(2):458–487, 1995.

[10] O. Grumberg and D.E. Long. Model checking and modular verification. *TOPLAS*, 16(3):843–871, 1994.

[11] M.C.B. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *J. of the ACM*, 32(1):137–161, 1985.

[12] R. Kaivola and A. Valmari. The weakest compositional semantic equivalence preserving nexttime-less linear temporal logic. In *CONCUR '92*, volume 630 of *LNCS*, pages 207–221. Springer-Verlag, 1992.

[13] O. Kupferman and M.Y. Vardi. Modular model checking. In *Compositionality: The Significant Difference*, volume 1536 of *LNCS*. Springer-Verlag, 1997.

[14] R.P. Kurshan. *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach*. Princeton University Press, 1994.

[15] L. Lamport. The temporal logic of actions. *TOPLAS*, 16(3):872–923, 1994.

[16] K.G. Larsen. The expressive power of implicit specifications. *TCS*, 114(1):119–147, 1993.

[17] O. Lichtenstein, A. Pnueli, and L. Zuck. The glory of the past. In *Workshop on Logics of Programs*, volume 193 of *LNCS*, pages 196–218. Springer-Verlag, 1985.

[18] M.G. Main. Trace, failure and testing equivalences for communicating processes. *J. of Par. Comp.*, 16(5):383–400, 1987.

[19] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[20] K. Narayan Kumar, R. Cleaveland, and S.A. Smolka. Infinite probabilistic and nonprobabilistic testing. In *FSTTCS '98*, volume 1530 of *LNCS*, pages 209–220. Springer-Verlag, 1998.

[21] V. Natarajan and R. Cleaveland. Divergence and fair testing. In *ICALP '95*, volume 944 of *LNCS*, pages 684–695. Springer-Verlag, 1995.

[22] A. Pnueli. The temporal logic of programs. In *FOCS '77*, pages 46–57. IEEE Computer Society Press, 1977.

[23] A. Puhakka and A. Valmari. Weakest-congruence results for livelock-preserving equivalences. In *CONCUR '99*, volume 1664 of *LNCS*, pages 510–524. Springer-Verlag, 1999.

[24] B. Steffen and A. Ingólfsdóttir. Characteristic formulae for CCS with divergence. *Inform. & Comp.*, 110(1):149–163, 1994.

[25] C. Stirling. Modal logics for communicating systems. *TCS*, 49:311–347, 1987.

[26] A. Valmari and M. Tiernari. Compositional failure-based semantics models for basic LOTOS. *FAC*, 7(4):440–468, 1995.

[27] M. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *LICS '86*, pages 332–344. IEEE Computer Society Press, 1986.