Three Logics for Branching Bisimulation

ROCCO DE NICOLA

Università di Roma "La Sapienza", Rome, Italy

AND

FRITS VAANDRAGER

CWI, Amsterdam, The Netherlands

Abstract. Three temporal logics are introduced that induce on labeled transition systems the same identifications as branching bisimulation, a behavioral equivalence that aims at ignoring invisible transitions while preserving the branching structure of systems. The first logic is an extension of Hennessy–Milner Logic with an "until" operator. The second one is another extension of Hennessy–Milner Logic, which exploits the power of backward modalities. The third logic is CTL* without the next-time operator. A relevant side-effect of the last characterization is that it sets a bridge between the state- and action-based approaches to the semantics of concurrent systems.

Categories and Subject Descriptors: F.1.1 [**Computation by Abstract Devices**]: Models of Computation; F.3.1 [**Logics and Meanings of Programs**]: General;

General Terms: Theory, verification

Additional Key Words and Phrases: Backward modalities, branching bisimulation equivalence, concurrency, CTL*, doubly labeled transition systems, Hennessy–Milner logic, Kripke structures, labeled transition systems, reactive systems, semantics, stuttering equivalence, until operations

R. De Nicola was partially supported by Esprit Basic Research Action Program, Project 3011 CEDISYS, by CNR Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo contract number 91.00894.69.

The research of F. Vaandrager was supported by RACE Project 1040, SPECS.

Part of the research was carried out while R. De Nicola was with the Institute di Elaborazione dell'Informazione.

An extended abstract of this paper appeared as DE NICOLA, R., AND VAANDRAGER, F. W. 1990. Three logics for branching bisimulation (extended abstract). In *Proceedings of the 5th Annual Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society Press, New York, pp. 118–129; this a revised version of CWI Report CS-R9012, Amsterdam 1990.

Authors' addresses: R. De Nicola, Dipartimento di Scienze dell'Informazione, Universita di Roma, "La Sapienza", Via Salaria 113, I-00198 Rome, Italy, e-mail: DENICOLA@VM.CNUCE.CNR.IT; F. Vaandrager, CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands, e-mail: FRITSV@CWI.NL.

1. *Introduction*

The operational semantics of concurrent systems has often been described by means of labeled transition systems. However, these descriptions are frequently too concrete and do not always give the same account of systems that exhibit identical observable behavior. The addition of plausible notions of behavioral equivalences permits one to overcome these problems. These notions make it possible to relate systems described at different levels of abstraction and to verify, for example, the correctness of an implementation with respect to a more abstract specification of a given system. The interested reader is referred to De Nicola [1987] and van Glabbeek [1990] for comparative presentations of many such equivalences.

Among the best known behavioral equivalences are the *bisimulation equivalences* (also called *observational equivalences*) of Milner [1989] and Park [1989]. Intuitively, two systems are bisimulation equivalent whenever they can perform the same sequences of actions to reach bisimulation equivalent states. Bisimulation equivalences are called *strong* when all labels of transitions are considered as visible, and *weak* when they ignore some actions, considered internal and thus invisible. Bisimulation equivalences have proved of fundamental importance for working with structures used to describe nondeterministic systems. Indeed, two of the major schools of concurrency theory, that of CCS [Milner 1989] and ACP [Baeten and Weijland 1990] consider bisimulations as the basic equivalence notation, and have developed a rich and powerful theory around them. The existence of a bisimulation of some type between two structures, means that at a deep level they are very much alike. The definition of bisimulations already suggests a useful method for showing equivalence of two systems: one guesses a relation among the states of the systems, and verifies that it is a bisimulation relation. Checking this is local and involves only one or a few computation steps at the time. In the case of finite state systems, one can alternatively use one of the efficient algorithms based on partition refinement for deciding bisimulation. Finally, there are elegant complete equational axiom systems for a wide variety of bisimulation based process algebras, see Baeten and Wejland [1990] for applications. In spite of some theoretical concerns (e.g., bisimulations are too fine, capable of distinguishing systems that ought to be identified [Abramsky 1987; Bloom et al. 1989]), bisimulations are a central part of concurrency theory.

In parallel with the definition of behavioral equivalences, different attempts have been made towards defining (modal and temporal) logics that permit specifying specific properties of concurrent systems. The logics having the advantage over behavioral equivalences of not always requiring to specify the full behavior of a system; they permit one to concentrate on specifying particular properties of a system, like safety, fairness, etc., that are of interest.

Indeed, modal and temporal logics have been proved useful formalisms for specifying and verifying properties of concurrent systems (see, e.g., de Bakker et al. [1989] and Manna and Pnueli [1992]), and different tools have been developed to support such activities [Clarke et al. 1986; Cleaveland et al. 1990]. However, to date, there is no general agreement on the type of logic to be used. Since a logic naturally gives rise to equivalences (two systems are equivalent if they satisfy the same formulas) often the proposed logics have been contrasted with behavioral equivalences for a better understanding and evaluation. In general, establishing a direct correspondence between a logic and a behavioral equivalence provides additional confidence in both approaches.

A well-known result relating operational and logical semantics is that reported in Hennessy and Milner [1985]. In that paper, a modal logic, now known as Hennessy–Milner Logic (HML), is defined which, when interpreted over (arc-) labeled transition systems with and without silent actions, is proved to be in full agreement with the two operational equivalences called *strong* and *weak observational equivalence*. Other correspondences have been established in Browne [1988], where two equivalences over Kripke structures (node-labeled transition systems) are related to two variants of CTL* [Emerson and Halpern 1986]. It is first shown that a variant of strong observational equivalence coincides with the equivalence induced by CTL*; and then that CTL* without the next operator (CTL*-X) is in full agreement with *stuttering equivalence*, an equivalence based on the idea of merging adjacent states that have the same labelling.

Recently, a new notion of behavioral equivalence for labeled transition systems, called *branching bisimulation* ($\approx_b$), has been proposed [van Glabbeek and Weijland 1989]. It aims at generalizing strong observational equivalence to ignore silent actions while preserving the branching structures of systems. Branching bisimulation considers two systems as equivalent only if every computation, that is, every alternating sequence of (visible and silent) actions and states, of one system has a correspondent in the other. By correspondent computations it is meant computations with the same sequence of visible actions and such that all their intermediate states have equivalent potentials.

Branching bisimulation is more restrictive than weak observational equivalence but has a pleasant axiomatic characterization that leads to a complete canonical-term rewriting system [Akkerman and Baeten 1990; De Nicola et al. 1990] and does indeed preserve the branching structures of systems. In Groote et al. [1990] an $O(m \times n)$ algorithm—m is the number of transitions and n is the number of states in the transition system—for branching bisimulation is presented; a trial implementation of this algorithm runs faster than existing tools for deciding weak observational equivalence. An additional pleasant properties of branching bisimulation is that it is resistant to refinement of actions while weak bisimulation is not [Darondeau and Degano 1990; van Glabbeek and Weijland 1989/1991].

In this paper, we propose three logical characterizations of branching bisimulation, that on one hand permit a deeper understanding of the equivalence itself and on the other hand permit using existing tools to tackle the problem of mechanical support to the verification of properties of concurrent systems. The three logics we will present are (natural extensions of) well-known and thoroughly studied logics. The first logic we will consider, L_U, is obtained from HML by replacing the indexed operator $\langle a \rangle$ with a kind of "until" operator. The new binary operator, written $\varphi\langle a \rangle\varphi'$, tests whether a system can reach, by exhibiting a visible action "a", a state that satisfies φ' while moving only through states that satisfy φ. The second logic, L_{BF}, stems from the characterization of $\approx_b$ as a *back and forth bisimulation equivalence* [De Nicola et al., 1990]. It extends HML with reverse modalities that permit inquiries to be made about the past of computations (see, e.g., Lichtenstein et al. [1985], Stirling [1992], Street [1982]). The third logic that we use to characterize $\approx_b$ is a variant of CTL*. More specifically it is CTL*-X interpreted, as in the original proposal (see Emerson and Srinivasan [1989]), over all paths of Kripke structures and not just over maximal ones.

The actual proof of the correspondence between CTL*-X and $\approx_b$ has interesting side-effects. It requires establishing precise connections between branching bisimulation and the *stuttering equivalence* over Kripke structures ($\approx_s$) defined by Browne et al. [1988]; these connections permit clarifying the relationships between the state- and action-based approaches to the semantics of concurrent systems also in presence of invisible events. We set up a general construction that, given a labeled transition system, yields an enriched system that has essentially the same structure of the original one, but carries labels on both states and transitions. We prove that a *divergence blind* version of stuttering equivalence and $\approx_b$, and a *divergence sensitive* version of branching bisimulation and $\approx_s$ induce the same identifications on the enriched transition systems.

The different logics characterizing the same equivalence over a given class of systems can serve different (complementary) purposes. Indeed, as they are at the moment, L_U and L_{BF} cannot really be used to specify systems properties (they would need at least adding a recursion operator). However, due to their closeness to the operational description, they are optimal for explaining the differences between inequivalent systems. CTL, on the other hand, has been successfully used to specify systems properties.

Since their publication in the conference version of this paper [De Nicola and Vaandrager 1990b], its results have inspired subsequent work.

—Korver [1992] has defined an algorithm that, given two states of a finite automaton that are not branching bisimulation equivalent, produces an L_U formula that distinguishes between them. Such an algorithm provides a useful extension of the algorithm of Groote and Vaandrager [1990] since it helps a user in understanding why certain finite state systems are inequivalent. Polak [1992] describes an implementation of the algorithm of Korver [1992] on the top of that of Groote and Vaandrager [1990].

—The intermediate structures we had to introduce to prove that CTL* is an adequate logic for branching bisimulation led us to define also an action-based version of CTL, [De Nicola and Vaandrager 1990a] that we called ACTL. This logic can naturally be used to describe safety and liveness properties of systems and permits reasoning in terms of the actions they can perform, rather than in terms of the properties of their states.

—Minor modifications of the translation functions between Kripke Structures and Labeled Transition Systems have also allowed us to build a model checker for ACTL that completely relies on the existing model checker for CTL and guarantees linear model checking for action-based formulas [De Nicola et al. 1993]. This has permitted the implementation of a verification environment where both logical and behavioral properties can be proved by relying on a single underlying model, namely that of Labeled Transition Systems (see, e.g., LITE [Bolognesi et al. 1995]).

—The alternative characterization of $\approx_s$ in terms of divergence blind stuttering equivalence has been used as a key step towards the $O(m \times n)$ algorithm for deciding stuttering equivalence of Groote and Vaandrager [1990], which is a definite improvement of the $O(n^5)$ algorithm of Browne et al. [1988].

—In Laroussinie et al. [1995], it has been shown that, for each L_U formula, there is a logically equivalent L_{BF} formula, and vice versa. The (rather complex) proof uses an auxiliary logic L_{BU} that combines the modalities of L_U and L_{BF}, and gives effective procedures by which L_U formulas can be rewritten into L_{BF} formulas, and vice versa, via rewrite steps that preserve logical equivalence.

The rest of the paper is organized as follows: In the next section, we present branching bisimulation and two of the logics that will be used to characterize it; namely, HML with the until operator and HML with backward modalities. In the third section, we consider CTL and CTL* and show that minor variants of them are in full agreement with branching bisimulation; to do this we define transformations that permit us to move freely between state labeled systems and transition labeled ones. The final section contains concluding remarks, comparisons with related works and pointers to further research.

2. *Branching Bisimulation and Hennessy–Milner Logics*

In this section, we introduce two logical characterizations of branching bisimulation equivalence based on Hennessy–Milner Logic, HML for short. The first logic relies on a kind of until operator which, given a sequence of transitions (run), permits testing not only what is true after that run but also what are the properties that hold along it. The second logic introduces a backward modality and permits to test both for properties that hold after the execution of a particular visible action and for properties that were enjoyed before the execution of the action.

2.1. Labeled Transition Systems and Branching Bisimulation.

We will now provide the necessary background definitions about transition systems and their runs and introduce branching bisimulation. The actual definition of the latter is slightly simpler and apparently less restrictive than the original one proposed in van Glabbeek and Weijland [1989/1991]; however, it can be easily proved that our equivalence does indeed coincide with the original one.

Definition 2.1.1 (*Notation for strings*). Let K be any set. K^* stands for the set of finite sequences of elements of K; K^ω denotes the set of infinite sequences of elements of K; K^∞ stands for $K^\omega \cup K^*$. Concatenation of a sequence in K^* with a sequence in K^∞ is denoted by juxtaposition; ϵ denotes the empty sequence; $|\sigma|$ denotes the length of a sequence σ.

Definition 2.1.2 (*Labeled Transition Systems*). A *labeled transition system* (or *LTS*) is a triple $\mathscr{A} = (S, A, \rightarrow)$ where:

—S is a set of *states*;
—A is a set of *actions*; also a *silent action* τ is assumed that is not in A;
—$\rightarrow \subseteq S \times (A \cup \tau) \times S$ is the *transition relation*. An element (r, α, s) of $\rightarrow$, usually written as $r \xrightarrow{\alpha} s$, is called a *transition*.

We assume $\epsilon \notin A$ and use A_τ to denote $A \cup \{\tau\}$ and A_ϵ to denote $A \cup \{\epsilon\}$. Moreover, we let $r, s, \ldots$ range over S; $a, b, \ldots$ range over A; $\alpha, \beta, \ldots$ range over A_τ; and $h, k, \ldots$ range over A_ϵ. We will also make use of the mapping $(\cdot)^\circ : A_\tau \rightarrow A_\epsilon$ defined by $\alpha^\circ = \alpha$ if $\alpha \in A$ and $\alpha^\circ = \epsilon$ otherwise.

Definition 2.1.3 (*Runs over LTSs*). Let $\mathscr{A} = (S, A, \rightarrow)$ be an LTS.

—a *run* of $\mathscr{A}$ is a finite, nonempty alternating sequence $\rho = s_0 \alpha_0 s_1 \alpha_1 \cdots s_{n-1} \alpha_{n-1} s_n$ of states and actions, beginning and ending with a state, such that, for $0 \leq i < n$, $s_i \xrightarrow{\alpha_i} s_{i+1}$. We say also that ρ is a *run from* s_0;
—If $\rho = s_0 \alpha_0 s_1 \alpha_1 \cdots s_{n-1} \alpha_{n-1} s_n$ is a run then *first*$(\rho) = s_0$ and *last*$(\rho) = s_n$;
—We write $\text{run}_{\mathscr{A}(s)}$, or just run(s), for the set of runs from s;
—We write $\text{run}_{\mathscr{A}}$ for the set of runs in $\mathscr{A}$.

We let $\rho, \sigma, \ldots$ range over runs. With abuse of notation, we will sometimes write s for the run consisting just of state s.

Definition 2.1.4 (*Many step transitions and bounded nondeterminism*). Let $\mathscr{A} = (S, A, \rightarrow)$ be an LTS.

(i) With $\overset{\epsilon}{\Rightarrow}$ we denote the transitive and reflexive closure of $\xrightarrow{\tau}$. For $a \in A$, we define the relation $\overset{a}{\Rightarrow}$ on S, by $r \overset{\alpha}{\Rightarrow} s$ iff there exists r' and s' in S such that $r \overset{\epsilon}{\Rightarrow} r' \xrightarrow{a} s' \overset{\epsilon}{\Rightarrow} s$.
(ii) $\mathscr{A}$ has *bounded nondeterminism* iff for all $s \in S$ and $k \in A_\epsilon$ the set $\{r \overset{k}{\Rightarrow} r\}$ is finite.

Definition 2.1.5 (*Branching bisimulation*). Let $\mathscr{A} = (S, A, \rightarrow)$ be an LTS.

—A relation $R \subseteq S \times S$ is called a *branching bisimulation* if it is symmetric and satisfies the following *transfer property*: if $r \mathrel{R} s$ and $r \xrightarrow{\alpha} r'$, then either $\alpha = \tau$ and $r' \mathrel{R} s$, or $\exists s', s''$ such that $s \overset{\epsilon}{\Rightarrow} s' \xrightarrow{\alpha} s''$, $r \mathrel{R} s'$ and $r' \mathrel{R} s''$.
—Two states r, s of S are *branching bisimilar*, abbreviated $\mathscr{A}$: $r \approx_b s$, or $r \approx_b s$, if there exists a branching bisimulation relating r and s.

The diagrams shown in Figure 1 summarize the main transfer properties of branching bisimulation. We have used the dotted lines to represent the relations that have to be established in order to conclude that the two states connected by the plain line are equivalent.

It can be easily proved [van Glabbeek and Weijland 1989] that the arbitrary union of branching bisimulation relations is again a branching bisimulation, and that $\approx_b$ is the maximal branching bisimulation and an equivalence relation. We could have strengthened the above definition of branching bisimulation by requiring *all* intermediate states in $s \overset{\epsilon}{\Rightarrow} s'$ to be related with r. The following lemma implies that this would have led to the same equivalence relation. The same would have happened if we had allowed for extra τ-moves after reaching s'' and required that all reached states be related to r'.

LEMMA 2.1.6 (*Stuttering lemma*). *Let $\mathscr{A} = (S, A,)$ be an LTS and let $s_0 \tau s_1 \tau \cdots s_{n-1} \tau s_n$, $n > 0$, be a run in $\mathscr{A}$ with $s_0 \approx_b s_n$. Then, for all $0 \leq i \leq n$, $s_0 \approx_b s_i$.*

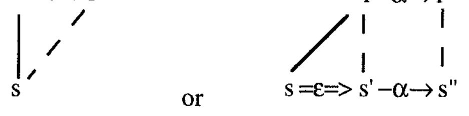


FIG. 1. Transfer diagrams for branching bisimulation.

PROOF. This lemma is due originally to van Glabbeek and Weijland [1989]. In its present form, it has been proved in De Nicola et al. [1990]. It is reported in the appendix for the sake of completeness. □

We would like to note that the definition of *weak bisimulation* [Milner 1989] is similar to that for the branching one; it only relies on a slightly less demanding transfer property, namely:

—if $r \mathrel{R} s$ and $r \overset{\alpha}{\Rightarrow} r'$, then $\exists s'$ such that $s \overset{\alpha}{\Rightarrow} s'$ and $r' \mathrel{R} s'$.

This means that two states are considered equivalent if they lead, via the same sequences of visible actions, to equivalent states; the intermediate states are not questioned. Formally, two states r, s of an LTS $\mathscr{A}$ are *weakly bisimilar*, notation $\mathscr{A}$: $r \approx_{\mathrm{w}} s$, or $r \approx_{\mathrm{w}} s$, if there exists a weak bisimulation relating r and s.

The diagram shown in Figure 2 summarizes the transfer property for the weak bisimulation. We have used the same notational conventions of Figure 1.

2.2. HENNESSY–MILNER LOGIC. In the rest of the paper, we will study the equivalences induced by different logics. For this, the following general definition will be useful.

Given a logical language L and an associated satisfaction relation $\models$ interpreted over states of a labelled transition system $\mathscr{A}$, the equivalence $\sim_{\mathrm{L}}$ on the states of $\mathscr{A}$, induced by L-formulas, is given by:

$$\mathscr{A}: r \sim_{\mathrm{L}} s \text{ if and only if } (\forall \varphi \in L: \mathscr{A}, r \models \varphi \Leftrightarrow \mathscr{A}, s \models \varphi).$$

The main aim of this paper is to show that, for three significantly different logics L, the equivalence $\sim_{\mathrm{L}}$ coincides with branching bisimulation equivalence.

In the following definitions, we will present syntax and semantics of the original Hennessy–Milner Logic (HML) and state the main characterization theorem, which establishes the strict correspondence between HML and weak bisimulation. In the definitions, and in the rest of the paper, we will use T to denote the Boolean value true.

Definition 2.2.1 (*Hennessy–Milner Logic*). Let A be a given alphabet of symbols. The syntax of HML is defined by the following grammar, where $\varphi, \varphi', \ldots$ range over HML-formulas and k ranges over A_ϵ:

$$\varphi ::= T \mid \neg \varphi \mid \varphi \wedge \varphi' \mid \langle k \rangle \varphi.$$

Definition 2.2.2 (*The satisfaction relation for HML*). Let $\mathscr{A} = (S, A, \rightarrow)$ be a LTS. *Satisfaction* of a HML-formula φ by a state $s \in S$, notation $\mathscr{A}, s \models \varphi$, or just $s \models \varphi$, is defined inductively by:

—$s \models T$	always
—$s \models \neg \varphi$	iff $s \not\models \varphi$
—$s \models \varphi \wedge \varphi'$	iff $s \models \varphi$ and $s \models \varphi'$
—$s \models \langle k \rangle \varphi$	iff there is an s' such that $s \overset{k}{\Rightarrow} s'$ and $s' \models \varphi$.

$$\begin{tikzcd} r \arrow[r, Rightarrow, "k"] \arrow[d, no head] & r' \arrow[d, no head, dashed] \\ s \arrow[r, Rightarrow, "k"] & s' \end{tikzcd}$$

FIG. 2. Transfer diagram for weak bisimulation.

For labeled transition system with bounded nondeterminism, the above logic has been proved to be in full agreement with weak observational equivalence [Hennessy and Milner 1985].

THEOREM 2.2.3 (*HML and $\approx_w$ induce the same identifications on bounded LTSs*). *Let $\mathscr{A} = (S, A, \rightarrow)$ be a LTS with bounded nondeterminism. Then, for all r, s in S:*

$$\mathscr{A}\colon r \approx_w s \textit{ if and only if } \mathscr{A}\colon r \sim_{HML} s.$$

2.3. UNTIL OPERATORS. We are now set to introduce the variant of Hennessy–Milner Logic, which, rather than a family of diamond operators, has indexed until operators. Within the new version of HML, in order to take also the properties of the intermediate states of a run into account, we replace the diamond operator $\langle k \rangle \varphi$ with a binary operator, written $\varphi \langle k \rangle \varphi'$, which is used to test, whether a system can reach via action k, a state which satisfies φ' while moving only through intermediate states that satisfy φ.

Definition 2.3.1 (*Hennessy–Milner Logic with Until*: L_U). Let A be a given alphabet of symbols. The syntax of the language L_U is defined by the following grammar where $\varphi, \varphi' \cdots$ range over L_U-formulas and $\langle k \rangle$ ranges over A_ϵ:

$$\varphi ::= T \mid \neg\, \varphi \mid \varphi \wedge \varphi' \mid \varphi \langle k \rangle \varphi'.$$

Definition 2.3.2 (*The satisfaction relation for* L_U). Let $\mathscr{A} = (S, A, \rightarrow)$ be an LTS. *Satisfaction* of an L_U-formula φ by a state $s \in S$, notation $\mathscr{A}, s \models \varphi$, or just $s \models \varphi$, is defined inductively by:

—$s \models t$	always
—$s \models \neg\, \varphi$	iff $s \not\models \varphi$
—$s \models \varphi \wedge \varphi'$	iff $s \models \varphi$ and $s \models \varphi'$
—$s \models \varphi \langle k \rangle \varphi'$	iff either $k = \epsilon$ and $s \models \varphi'$, or there is a run $s_0 \tau s_1 \tau \cdots s_{n-1} \tau s_n \alpha s_{n+1}$ such that $s_0 = s$ and $\forall i \leq n$: $s_i \models \varphi$, $k = \alpha^\circ$ and $s_{n+1} \models \varphi'$ with $n \geq 0$.

It is possible to define, within L_U, other temporal operators. For example, we will write $\langle k \rangle \varphi$ for $T \langle k \rangle \varphi$, $\varphi [k] \varphi'$ for $\neg(\neg\, \varphi \langle k \rangle \neg\, \varphi')$ and $[k]\varphi$ for $\neg\, T[k]\varphi$. The original HML can be recovered from L_U in the sense that the diamond operator "$\langle k \rangle \varphi$" of HML is rendered by our $\langle k \rangle \langle \epsilon \rangle \varphi$ or, more directly, by $T\langle k \rangle (T \langle \epsilon \rangle \varphi)$. In the latter formula, we need to have $\langle \epsilon \rangle$ after $\langle k \rangle$ because the relativized until operators are interpreted only over runs which always end with the action which indexes them. In HML, this restriction is not present and, when defining satisfaction of $\langle k \rangle \varphi$, runs are considered which may continue with sequences of invisible actions. Clearly, if no silent action is present, the logics L_U and HML are equivalent.

We exhibit now two pairs of systems and two formulas that show the additional power of L_U when compared with the original Hennessy–Milner Logic. The two pairs $\langle r, s \rangle$ and $\langle p, q \rangle$ of Example 2.3.3 are just two instances of the second and third τ-law (see, e.g., Milner [1989]), respectively. Thus, since $r \approx_w s$ and $p \approx_w q$, these states are certainly not differentiated by HML. However, we will see that there exist L_U formulas that can tell them apart.

Example 2.3.3 (*L_U can distinguish weakly equivalent states*). Consider LTS $\mathscr{A}_{2.3.3}$, as shown in Figure 3.

If we let $\varphi = (\langle b\rangle T)\langle a\rangle\, T$, then $s \models \varphi$ while $r \not\models \varphi$.

If we let $\varphi' = [a](\langle c\rangle T)$, then $p \models \varphi'$ while $q \not\models \varphi'$.

For the first pair of systems, we have that on the one hand from state s it is possible to perform an a-step such that, at any point before the a actually takes place, it is still possible to perform a b-step. On the other hand, from state r, there is only one execution possible that contains an a, and in this execution, the option of performing a b-step is lost after the initial τ-step.

For the second pair of systems, we have that if an a-step is performed from state p, then always immediately after this, the option is left of performing a c-step; this is not the case for state q.

We are now ready to establish the relationships between branching bisimulation equivalence and the equivalence induced by L_U. In the theorem below, we will restrict attention to bounded LTSs simply for a matter of separation of concerns. We do not foresee many problems in generalizing our results by resorting to infinitary logics in the same vein of Milner [1989]. However, the addition of such infinitary connectives would have complicated definitions and proofs without adding much insight.

THEOREM 2.3.4 (*L_U and $\approx_b$ induce the same identifications on bounded LTSs*). *Let $\mathscr{A} = (S, A, \rightarrow)$ be an LTS with bounded nondeterminism. Then, for all r, s in S:*

$$\mathscr{A}\colon r \approx_b s \text{ if and only if } \mathscr{A}\colon r \sim L_U s.$$

PROOF. "$\Rightarrow$" Suppose $r \approx_b s$ and let $\varphi \in L$. With rather straightforward induction on the structure of φ we prove that $r \models \varphi$ iff $s \models \varphi$.

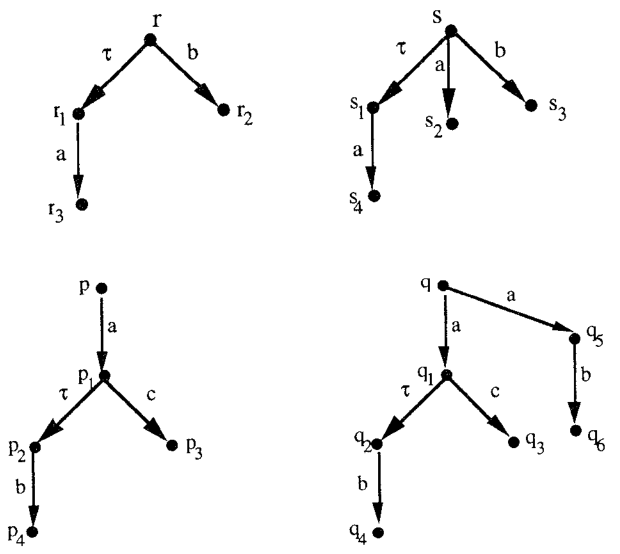


FIG. 3. Two pairs of threes that are not branching bisimilar.

(1) If $\varphi = T$, then obviously $s \models \varphi$ and $r \models \varphi$.
(2) If $\varphi = \neg \varphi'$, then, by definition $r \models \varphi$ iff $r \not\models \varphi'$. By induction $r \not\models \varphi'$ iff $s \not\models \varphi'$. Again by definition $s \not\models \varphi'$ iff $s \models \varphi$.
(3) If $\varphi = \varphi_1 \wedge \varphi_2$, then by definition we have $s \models \varphi_1$ and $s \models \varphi_2$ and the claim follows by straightforward induction.
(4) If $\varphi = \varphi_1 \langle k \rangle \varphi_2$, suppose that $r \models \varphi$. We will prove that $s \models \varphi$. The reverse implication then follows by symmetry. We have to distinguish two cases:

(i) $k = \epsilon$ and $r \models \varphi_2$;
(ii) there exists a run $r_0 \tau r_1 \tau \cdots r_{n-1} \tau r_n \alpha r_{n-1}$ such that $r = r_0$, $k = \alpha^\circ$ and $\forall i \leq n$: $r_i \models \varphi_1$, $k = \alpha^\circ$ and $r_{n+1} \models \varphi_2$.

In case (i), by the inductive hypothesis, we have $s \models \varphi_2$; hence, $s \models \varphi$ follows.

In case (ii), by repeatedly applying the transfer property of branching bisimulation equivalence, we can construct a matching execution from s. The simplest case is when $k = \epsilon$ and $r_{n+1} \approx_b s$. In this case, the matching run consists just of s and $s \models \varphi$ follows by induction. Otherwise, there exists a run $s_0 \tau s_1 \tau \cdots s_{m-1} \tau s_m \alpha s_{m+1}$ with $s = s_0$ and by the stuttering lemma (Lemma 2.1.6) $r \approx_b s_i$ for all $i \leq m$ and $r_{n+1} \approx_b s_{m+1}$. From the inductive hypothesis, we have that $s_i \models \varphi_1$ for all $i \leq m$, and that $s_{m+1} \models \varphi_2$. From this, $s \models \varphi$ follows.

"$\Leftarrow$" Suppose $r \sim_{L_U} s$. We prove that $\sim_{L_U}$ is a branching bisimulation. Clearly the relation is symmetric. Suppose $p \sim_{L_U} q$ and $p - \alpha \rightarrow p'$. A first possibility is that both $\alpha = \tau$ and $p' \sim_{L_U} q$. In this case the transfer property holds trivially. So suppose that either $\alpha \neq \tau$ or not $p' \sim_{L_U} q$.

Consider the set **Q** of all runs from q of the form $q_0 \tau q_1 \cdots q_{n-1} \tau q_n \alpha q'$ with $q_0 = q$ such that there are no cycles in the τ-part (i.e., $\forall i, j$: $q_i = q_j$ implies $i = j$). We claim that **Q** is finite. To see this, consider the set S_0 of states that occur in a nonfinal position of a run of **Q**, and the set S_1 of final states of runs of **Q**. Since $\mathscr{A}$ has bounded nondeterminism and since $q \overset{\epsilon}{\Rightarrow} q'$, for all states $q' \in S_0$, we have that S_0 is finite. Similarly, we can deduce that also S_1 is finite, because $q \overset{k}{\Rightarrow} q'$ for all states $q'' \in S_1$. Finiteness of S_1 and S_0 together with the fact that there are no repetitions of states of S_0 in the τ-part of the runs in **Q** implies that **Q** is finite.

In order to prove the transfer property, it is sufficient to show that there is a run in **Q** with the property that all states on the run, except for the last one, are related via $\sim_{L_U}$ to p, and the last state is related via $\sim_{L_U}$ to p'. Suppose that there is no such run. We will derive a contradiction. We can split **Q** into two subsets $\mathbf{Q}_s$ and $\mathbf{Q}_f$ such that for any run σ_s in $\mathbf{Q}_s$ there is a formula φ_{σ_s} that holds in p but not in all nonfinal states of σ_s, and for any run σ_f in $\mathbf{Q}_f$ there is a formula φ_{σ_f} that holds in p' but not in the last state of σ_f. Let φ_s be the conjunction of the formulas φ_{σ_s} with σ_s in $\mathbf{Q}_s$ and let φ_f be the conjunction of the formulas φ_{σ_f} with σ_f in $\mathbf{Q}_f$. Now we can distinguish between two cases.

(1) $\alpha = \tau$. In this case, since not $(p' \sim_{L_U} q)$, there exists a φ_0 such that $p' \models \varphi_0$ but $q \not\models \varphi_0$. Consider the formula $\varphi = \varphi_s \langle \epsilon \rangle (\varphi_f \wedge \varphi_0)$. We have that $p \models \varphi$ while $q \not\models \varphi$ and thus a contradiction.

(2) $\alpha \neq \tau$. Now we take $\varphi = \varphi_s\langle \alpha \rangle \varphi_f$ and we have a contradiction because $p \models \varphi$ but $q \not\models \varphi$. $\square$

2.4. BACKWARD MODALITIES. In this section, we present a new kind of bisimulation that we call *back-and-forth bisimulation.* It not only requires the futures of equivalent processes to be equivalent but constrains also their pasts. This new bisimulation has been put forward in De Nicola et al. [1990], where it is proved that it induces on LTSs the same identifications as branching bisimulation. Here, we take advantage of this result and introduce a variant of Hennessy–Milner Logic with a backward modality that permits analyzing the past of computations. The spirit of this generalization of HML is similar to that proposed by Hennessy and Stirling [1985]; the relevant difference is that, here, the possibility that some of the actions might be invisible is also taken into account. In Hennessy and Stirling [1985], only visible actions are considered, and thus partially controlled state changes are not permitted. Indeed, the past operator is introduced in Hennessy and Stirling [1985] only to capture noncontinuous properties (e.g., fairness) of generalized transition systems. There it is also proved that, in the case of classical (limit-closed) transition systems without silent moves, the equivalence induced by the logic with the past operator coincides with strong bisimulation equivalence.

Before actually introducing the new logic, we need additional notation. Since we want to talk about the past of systems, we need to define transition relations on runs rather than on single states; this enables us to go back from a state along the run that represents its *history.* We can easily generalize the definition of the transition relation from states to runs:

—$\rho \xrightarrow{\alpha} \sigma$, if there exists a state s such that $\sigma = \rho\alpha s$;
—$\rho \overset{\epsilon}{\Rightarrow} \sigma$, if there exists $\rho_0, \rho_1, \dots, \rho_n$, $n \geq 0$, with $\rho = \rho_0$, $\rho_n = \sigma$ and $\rho_i \xrightarrow{\tau} \rho_{i+1}$ for all $0 \leq i < n$;
—$\rho \overset{a}{\Rightarrow} \sigma$, if there exist φ', σ' such that $\varphi \overset{\epsilon}{\Rightarrow} \rho' \xrightarrow{a} \sigma' \overset{\epsilon}{\Rightarrow} \sigma$.

In Definition 2.4.1, we present the definition of back-and-forth bisimulation; more detailed discussions and motivations of the new bisimulation and its consequences can be found in De Nicola et al. [1990]. Here, we would only like to stress, once again, that we do not define this new bisimulation as a relation between states but as a relation between runs.

Definition 2.4.1 (*Back-and-forth bisimulation*). Let $\mathscr{A} = (S, A, \to)$ be an LTS. Two states $r, s \in S$ are *back-and-forth bisimilar*, abbreviated $\mathscr{A}$: $r \approx_{\mathrm{bf}} s$ or $r \approx_{\mathrm{bf}} s$, if there exists a symmetric relation $R \subseteq \mathrm{run}_{\mathscr{A}} \times \mathrm{run}_{\mathscr{A}}$, called a *back-and-forth bisimulation*, satisfying:

(i) $r \,\mathrm{R}\, s$;
(ii) if $\rho \,\mathrm{R}\, \sigma$ and $\rho \overset{k}{\Rightarrow} \rho'$, then $\exists \sigma'$ such that $\sigma \overset{k}{\Rightarrow} \sigma'$ and $\rho' \,\mathrm{R}\, \sigma'$;
(iii) if $\rho \,\mathrm{R}\, \sigma$ and $\rho' \overset{k}{\Rightarrow} \rho$, then $\exists \sigma'$ such that $\sigma' \overset{k}{\Rightarrow} \sigma$ and $\rho' \,\mathrm{R}\, \sigma'$.

The diagram of Figure 4 illustrates that, in order to prove that two states are back and forth bisimulation equivalent, we need to prove that both their past and their future are in the same relation. As in the diagrams for weak and branching bisimulation, we have used the dotted lines to represent the relations that have to be established in order to conclude that the two states connected by the plain line are equivalent.

$$\begin{tikzcd}
r'' \arrow[r, Rightarrow, "h"] \arrow[d, dash] & r \arrow[r, Rightarrow, "k"] \arrow[d, dash] & r' \arrow[d, dash] \\
t'' \arrow[r, Rightarrow, "h"] & s \arrow[r, Rightarrow, "k"] & s'
\end{tikzcd}$$

FIG. 4. Transfer diagram for back-and-forth bisimulation.

THEOREM 2.4.2 (*Back-and-forth and branching bisimulation induce the same identifications*). *Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS. Then for r, s in S: $\mathcal{A}$: $r \approx_b s$ if and only if $\mathcal{A}$: $r \approx_{bf} s$.*

PROOF. This lemma has been proved in De Nicola et al. [1990]; it is reported in the appendix for the sake of completeness. □

Definition 2.4.3 (*Hennessy–Milner Logic with backward modalities*: L_{BF}). Let A be a given alphabet of symbols. The syntax of Back-and-Forth Logic L_{BF} is defined by the following grammar where φ and φ' denote generic formulas and k ranges over A_ϵ:

$$\varphi ::= T \mid \neg\, \varphi \mid \varphi \wedge \varphi' \mid \langle k \rangle \varphi \mid \langle ^{\leftarrow} k \rangle \varphi.$$

Definition 2.4.4 (*The satisfaction relation for* L_{BF}). Let $\mathcal{A} = (S, A, \rightarrow)$ be an LTS. *Satisfaction* of an L_{BF}-formula φ by a run ρ of $\mathcal{A}$, notation $\mathcal{A}$, $\rho \models \varphi$, or just $\rho \models \varphi$, is defined inductively by:

—$\rho \models T$	always;
—$\rho \models \neg\, \varphi$	iff $\rho \not\models \varphi$;
—$\rho \models \varphi \wedge \varphi'$	iff $\rho \models \varphi$ and $\rho \models \varphi'$;
—$\rho \models \langle k \rangle \varphi$	iff there exists a run ρ' such that $\rho \overset{k}{\Rightarrow} \rho'$ and $\rho' \models \varphi$;
—$\rho \models \langle ^{\leftarrow} k \rangle \varphi$	iff there exists a run ρ' such that $\rho' \overset{k}{\Rightarrow} \rho$ and $\rho' \models \varphi$.

It is worth pointing out that, when interpreted over transition systems without silent actions, the above logic does not provide us with any additional discriminating power with respect to HML. This consideration agrees with Hennessy and Stirling [1985] where it is shown that for the class of transition systems we are considering here, when no silent action is present, HML and L_{BF} do coincide. Thus, we have that HML, L_{BF} and L_{U} induce the same identifications on systems without silent actions. However, the example below shows that also L_{BF} is able to differentiate the systems of Example 2.3.3 and thus that, when dealing with systems with silent action, L_{BF} is more expressive than HML.

Example 2.4.5 (L_{BF} *can distinguish weakly equivalent states*). Let p, q, r, and s be as in Example 2.3.3, and let $[k] = \neg\langle k \rangle \neg$ and $[^{\leftarrow} k] = \neg\langle ^{\leftarrow} k \rangle \neg$.

If $\varphi = \langle a \rangle [^{\leftarrow} a] \langle b \rangle T$, then $s \models \varphi$ while $r \not\models \varphi$.

If $\varphi' = [a][b]\langle ^{\leftarrow} b \rangle \langle c \rangle T$, then $p \models \varphi'$ while $q \not\models \varphi'$.

THEOREM 2.4.6 (L_{BF} *and* $\approx_b$ *induce the same identifications on bounded LTSs*). *Let $\mathcal{A} = (S, A, \rightarrow)$ be an LTS with bounded nondeterminism. Then for all r, s in S:*

$$\mathcal{A}\!: r \approx_b s \text{ if and only if } \mathcal{A}\!: r \sim_{\mathrm{L}_{BF}} s.$$

PROOF. Given an LTS $\mathscr{A}$, we can build a new one, $BF(\mathscr{A})$, which is obtained by replacing the single step transition relation of $\mathscr{A}$ with the corresponding many step forward and backward arrows between runs of $\mathscr{A}$. More precisely, we define $BF(\mathscr{A}) = (\text{run}_{\mathscr{A}}, A_{\text{bf}}, \rightarrow_{\text{bf}})$ where $A_{\text{bf}} = A_\epsilon \cup \{{}^{\leftarrow}k | k \in A_\epsilon\}$ and for $\rho, \rho' \in \text{run}_{\mathscr{A}}$ and $k \in A_\epsilon$, $\rho \overset{k}{\rightarrow}_{\text{bf}} \rho'$ iff $\rho \overset{k}{\Rightarrow} \rho'$ and $\rho \overset{k}{\rightarrow}_{\text{bf}} \rho'$ iff $\rho' \overset{k}{\Rightarrow} \rho$. We can now prove that $\mathscr{A}$: $r \approx_{\text{bf}} s$ if and only if $BF(\mathscr{A})$: $r \approx s$, where $\approx$ stands for Milner's strong observational equivalence. The claim then follows directly from Theorem 2.4.2 and from the HML characterization of $\approx$ in Hennessy and Stirling [1985]. □

3. *Branching Bisimulation and CTL**

In this section, we shall study the relationship of branching bisimulation with a different type of logic, the temporal logic known as CTL*. This will be achieved by relating branching bisimulation to a variant of the stuttering equivalence defined and related to CTL* in Browne et al. [1988].

3.1 CTL* AND ITS MODELS. First of all, we introduce the relevant notation for the class of structures that have been used to interpret CTL* and to define stuttering equivalence.

Definition 3.1.1 (*Kripke structures*). Let **AP** be a fixed nonempty set of *atomic proposition names* ranged over by $p, q, \ldots$. A *Kripke structure* (or *KS*) is a triple $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ where:

—S is a set of *states*;
—$\mathscr{L}$: $S \rightarrow 2^{\mathbf{AP}}$ is the *proposition labeling*;
—$\rightarrow \subseteq S \times S$ is the *transition relation*; an element $(r, s) \in \rightarrow$, usually written as $r \rightarrow s$, is called a *transition*.

We let $r, s, \ldots$ range over states of Kripke structures.

Definition 3.1.2 (*Notation for Kripke structures*). Let $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ be a *Kripke structure*.

—A nonempty (finite or infinite) sequence $s_0 s_1 s_2 \cdots \in S^\infty$ such that $s_i \rightarrow s_{i+1}$, with $i \geq 0$, is called a *path* from s_0; if the sequence of pairs of states is maximal the path is called a *fullpath*.
—We write $\text{path}_{\mathscr{K}}(s)$, or just path(s), for the set of paths from s, and $\mu\text{path}_{\mathscr{K}}(s)$, or just $\mu\text{path}(s)$, for the set of *maximal paths* (fullpaths) from s.
—We let $\rho, \sigma, \theta, \eta, \ldots$ range over paths.
—If $\rho = s_0 s_1 s_2 \ldots$ is a path then *first*$(\rho) = s_0$; if ρ is finite then *last*(ρ) denotes the last state of ρ.
—With $\rho < \theta$ and $\rho \leq \theta$ we indicate that path θ is a proper suffix, respectively a suffix, of path ρ.

Definition 3.1.3 (*CTL* and CTL*). The set of formulas CTL* is defined as the smallest set of state formulas such that:

—if $p \in \mathbf{AP}$, then p is a state formula;
—if φ and φ' are state formulas, then $\neg\varphi$ and $\varphi \wedge \varphi'$ are state formulas;
—if π is a path formula, then $\exists\pi$ is a state formula;
—if φ is a state formula, then φ is a path formula;
—if π and π' are path formulas, then $\neg\pi$, $\pi \wedge \pi'$, $X\pi$ and $\pi U \pi'$ are path formulas.

We let $\varphi, \ldots$ range over state formulas and $\pi, \ldots$ over path formulas.

CTL is defined as the subset of CTL* in which we restrict path formulas to be:

—if φ and φ' are state formulas, then $X\varphi$ and $\varphi \mathrm{U} \varphi'$ are path formulas;
—if π is a path formula, then so is $\neg \pi$.

Below, when we write CTL*-X and CTL-X, we refer to the subsets of CTL* and CTL, consisting of formulas without the next (X) operator. Moreover, we write T for $\neg(p_0 \wedge \neg p_0)$, where p_0 is some arbitrarily chosen atomic proposition name, $\pi \vee \pi'$ for $\neg(\neg \pi \wedge \neg \pi')$, $\pi \Rightarrow \pi'$ for $\neg \pi \wedge \pi'$, $\forall \pi$ for $\neg \exists \neg \pi$, $F\pi$ for $T \mathrm{U} \pi$, and $G\pi$ for $\neg F \neg \pi$.

Now, we present two different satisfaction relations for CTL*. This will be done by relying on different structures to interpret formulas. In one case, we will use only maximal paths of Kripke structures to interpret path formulas; in the other, we will use both finite and infinite paths. Due to its ability of describing noncontinuous properties like fairness, the generally accepted interpretation of CTL*, is that based on maximal paths only. The less restrictive interpretation, however, has a series of interesting properties and is the version of CTL* that was originally proposed (see Emerson and Srinivasan [1989].

Definition 3.1.4 (*Two satisfaction relations for CTL**). Let $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ be a Kripke structure.

(i) Satisfaction of a state formula φ by a state s, notation $\mathscr{K}, s \models \varphi$ or just $s \models \varphi$, and of a path formula π by a path ρ, notation $\mathscr{K}, \rho \models \pi$ or just $\rho \models \pi$, is defined inductively by:

—$s \models p$	iff	$p \in \mathscr{L}(s)$
—$s \models \neg \varphi$	iff	$s \not\models \varphi$
—$s \models \varphi \wedge \varphi'$	iff	$s \models \varphi$ and $s \models \varphi'$
—$s \models \exists \pi$	iff	there exists a path $\rho \in \mathrm{path}(s)$ such that $\rho \models \pi$
—$\rho \models \varphi$	iff	$\mathrm{first}(\rho) \models \varphi$
—$\rho \models \neg \pi$	iff	$\rho \not\models \pi$
—$\rho \models \pi \wedge \pi'$	iff	$\rho \models \pi$ and $\rho \models \pi'$
—$\rho \models \pi \mathrm{U} \pi'$	iff	there exists a θ with $\rho \leq \theta$ such that $\theta \models \pi'$ and for all $\rho \leq \eta < \theta$: $\eta \models \pi$
—$\rho \models X\pi$	iff	there exists a state s and a path θ such that $\rho = s\theta$ and $\theta \models \pi$.

(ii) Satisfaction with respect to maximal paths of a state formula φ by a state s, notation $\mathscr{K}, s \models_\mu \varphi$ (or briefly $s \models_\mu \varphi$), and of a path formula π by a maximal path ρ, notation $\mathscr{K}, \rho \models_\mu \pi$ (or briefly $\rho \models_\mu \pi$) is defined by replacing in the above definition $\models$ by $\models_\mu$ and the definition of $s \models \exists \pi$ by:

—$s \models_\mu \exists \pi$ iff there exists a path $\rho \in \mu\mathrm{path}(s)$ such that $\rho \models_\mu \pi$.

3.2. CTL* AND STUTTERING EQUIVALENCES. We will now introduce stuttering equivalence. Actually, our definition of stuttering equivalence, although similar in spirit, is slightly different from that of Browne et al. [1988] they consider only structures whose states are never deadlocked; if systems that contain states without outgoing transition have to be modeled, they assume the presence of a transition from the final state to itself, thus all maximal paths of a system are infinite. We will take a somewhat complementary approach and rather than avoiding deadlocked states, we do emphasize their presence.

We will present two variants of stuttering equivalence that differ in the way they deal with divergent processes. These two variants will be proved to be in direct correspondence with the two interpretations of CTL* described above.

Definition 3.2.1 (*Divergence blind stuttering equivalence*). Let $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ be a Kripke structure.

(i) A relation $R \subseteq S \times S$ is called a *divergence blind stuttering bisimulation* (DBSB) if it is symmetric and whenever $r \mathrel{\mathrm{R}} s$ then:

—$\mathscr{L}(r) = \mathscr{L}(s)$ and
—if $r \rightarrow r'$, then there exist, with $n \geq 0, s_0, s_1, \dots, s_n$ such that $s_0 = s$ and for all $i < n$: $s_i \rightarrow s_{i+1}, r \mathrel{\mathrm{R}} s_i$ and $r' \mathrel{\mathrm{R}} s_n$.

(ii) Two states r, s are *divergence blind stuttering equivalent*, abbreviated $\mathscr{K}$: $r \approx_{\mathrm{dbs}} s$ or $r \approx_{\mathrm{dbs}} s$, if there exists a divergence blind stuttering bisimulation relating r and s.

(iii) Two paths ρ, σ and *divergence blind stuttering equivalent*, notation $\mathscr{K}$: $\rho \approx_{\mathrm{dbs}} \sigma$ or $\rho \approx_{\mathrm{dbs}} \sigma$, if ρ can be partitioned as $\rho_1 \rho_2 \cdots$ and σ can be partitioned as $\sigma_1 \sigma_2 \cdots$ in such a way that, for all j, sequences ρ_j and σ_j are both nonempty and every state in ρ_j is divergence blind stuttering equivalent to every state in σ_j.

As in the case of branching bisimulation, we have that the arbitrary union of DBSBs in again a DBSB, and that $\approx_{\mathrm{dbs}}$ is the maximal DBSB and an equivalence relation.

LEMMA 3.2.2. *Let $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ be a Kripke structure, let $r, s \in S$ with $r \approx_{dbs} s$, and let $\rho \in path(r)$. Then there exists a $\sigma \in path(s)$ such that $\rho \approx_{dbs} \sigma$.*

PROOF. The actual proof is easy, only notationally somewhat cumbersome; it is left to the reader. □

THEOREM 3.2.3. *Let $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ be a Kripke structure and let $r, s \in S$ with $r \approx_{dbs} s$. Then for every CTL*-X formula φ: $r \models \varphi$ iff $s \models \varphi$.*

PROOF. Suppose $r \approx_{\mathrm{dbs}} s$. Let $\rho \in \mathrm{path}(r)$ and $\sigma \in \mathrm{path}(s)$ with $\rho \approx_{\mathrm{dbs}} \sigma$ and let χ be either a state formula or a path formula that does not contain any X-operator. We will prove the following statements by induction on the structure of χ.

(i) If χ is a state formula, then $r \models \chi$ if and only if $s \models \chi$
(ii) If χ is a path formula, then $\rho \models \chi$ if and only if $\sigma \models \chi$.

First, we consider the case of state formulas.

(1) $\chi = p$: $r \models p$ iff $p \in \mathscr{L}(r)$, the latter is equivalent to $p \in \mathscr{L}(s)$ by definition of $r \approx_{\mathrm{dbs}} s$, and $p \in \mathscr{L}(s)$ iff $s \models \chi$.
(2) $\chi = \neg \varphi$: $r \models \neg \varphi$ iff $r \not\models \varphi$, this by induction is equivalent to $s \not\models \varphi$, which in turn is equivalent to $s \models \neg \varphi$.
(3) $\chi = \varphi \wedge \varphi'$: the fact that $r \models \varphi \wedge \varphi'$ iff $s \models \varphi \wedge \varphi'$ follows since, by induction, $r \models \varphi$ and $r \models \varphi'$ iff $s \models \varphi$ and $s \models \varphi'$.

(4) $\chi = \exists\pi$: Suppose $r \models \exists\pi$. Then there exists a path $\rho' \in \text{path}(r)$ such that $\rho' \models \pi$. By Lemma 3.2.2, we can find a path $\sigma' \in \text{path}(s)$ such that $\rho' \approx_{\text{dbs}} \sigma'$; moreover, by induction, we have that $\sigma \models \pi$. Thus, $s \models \exists\pi$. The other direction is symmetric.

Next, we consider the four cases of path formulas.

(5) $\chi = \varphi$: We have $\rho \models \varphi$ iff $\text{first}(\rho) = r \models \varphi$, which by induction is equivalent to $\text{first}(\sigma) = s \models \varphi$. By definition, the last statement is equivalent to $\sigma \models \varphi$.
(6) $\chi = \neg\pi$: Easy induction.
(7) $\chi = \pi \wedge \pi'$: Easy induction.
(8) $\chi = \pi \,\text{U}\, \pi'$: Suppose $\rho \models \pi \,\text{U}\, \pi'$. Then, there exists a path θ with $\rho \leq \theta$ such that $\theta \models \pi'$ and for all $\rho \leq \nu < \theta, \nu \models \pi$. Since $\rho \approx_{\text{dbs}} \sigma$, there exists a partition $\rho_1\rho_2 \cdots$ of ρ and a partition $\sigma_1\sigma_2 \cdots$ of σ such that for all j, ρ_j and σ_j are both nonempty and every state in ρ_j is stutteringly bisimilar to every state in σ_j. Now, let ρ_k be a suffix of ρ in which the first state of θ occurs. One can easily check that $\theta \approx_{\text{dbs}} \sigma_k\sigma_{k+1} \cdots$. Thus, by induction we have $\sigma_k\sigma_{k+1} \cdots \models \pi'$. Let η be a path such that $\sigma \leq \eta < \sigma_k\sigma_{k+1} \cdots$, and let σ_1 be the block in which the first state of η occurs; we have $\rho_1\rho_{1+1} \cdots \approx_{\text{dbs}} \eta$. Since $1 < k$, we have also $\rho < \rho_1\rho_{1+1} \cdots < \theta$ and thus $\rho_1\rho_{1+1} \models \pi$. By induction, we obtain also $\eta \models \pi$. The other direction is symmetric. □

THEOREM 3.2.4. *Let $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ be a finite state Kripke structure and let $s \in S$. Then there exists a CTL-X formula φ such that for all $r \in S$: $r \models \varphi$ iff $r \approx_{dbs} s$.*

PROOF. The actual proof is based on the algorithm for deciding divergence blind stuttering equivalence that is presented in Groote and Vaandrager [1990].

—For $B, B' \subseteq S$ the set $\text{pos}(B, B')$ is defined as the set of states in B from which, after some initial stuttering, a state in B' can be reached:

$$\text{pos}(B, B') = \{s \in B \mid \exists\ n \geq 0, \exists\ s_0, \ldots, s_n \in B, \exists\ s' \in B' \text{ such that } s = s_0 \text{ and } (\forall\ 0 < i \leq n\colon s_i \in B \text{ and } s_{i-1} \rightarrow s_i) \text{ and } s_n \rightarrow s'\}.$$

—Call B' a *splitter* of B if and only if $\varnothing \neq \text{pos}(B, B') \neq B$.
—If P is a partition of S with $B, B' \in P$ and B' is a splitter of B, define $Ref_p(B, B')$ as the partition obtained from P by replacing B by $\text{pos}(B, B')$ and $B - \text{pos}(B, B')$.
—A partition is *stable* if for no $B, B' \in P$, B' is a splitter of B.

Consider the following algorithm:

```
$P := \{\{r \in S \mid \mathscr{L}(r) = \mathscr{L}(s)\} \mid s \in S\}$;
**while** $P$ is not stable **do**
  **choose** $B, B' \in P$ such that $B'$ is a splitter of $B$;
  $P := Ref_p(B, B')$
  **od**
```

In Groote and Vaandrager [1990], it is shown that two states are in the same block of the final partition exactly when they are divergence blind stuttering equivalent. The idea of our proof is that, while executing the algorithm, we maintain a mapping that associates a CTL-X formula to each block that only holds for the states in that block. This is trivial for the initial partition. Since states in the same block have the same labeling while states in different blocks have different labeling, one can easily give a propositional formula for each block that only holds for its states. Suppose that, at one moment, during the execution of the algorithm, B' is a splitter for B and this is split into $B_1 = \text{pos}(B, B')$ and $B_2 = B - \text{pos}(B, B')$. Let φ' be the formula associated to B' and let φ be the formula associated to B. In the new partition we associate to B_1 the formula $\varphi \wedge (\exists\, \varphi \,\mathrm{U}\, \varphi')$ and to B_2 the formula $\varphi \wedge (\neg \exists\, \varphi \,\mathrm{U}\, \varphi')$. For the other blocks, the associated formulas remain unchained. Now, if we associate to every state the formula ψ that is associated to the block of the final partition in which the state occurs, then ψ will have the required property. □

THEOREM 3.2.5 (*Divergence blind stuttering, CTL*-X and CTL-X agree for* $\models$). *Let* $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ *be a finite state Kripke structure and let* $r, s \in S$. *The following statements are equivalent*:

(i) $r \approx_{dbs} s$,
(ii) *For every CTL*-X formula* φ: $r \models \varphi$ *iff* $s \models \varphi$, *and*
(iii) *for every CTL-X formula* φ: $r \models \varphi$ *iff* $s \models \varphi$.

PROOF. We have that (i) $\Rightarrow$ (ii) follows from Theorem 3.2.3; (ii) $\Rightarrow$ (iii) is immediate; while (iii) $\Rightarrow$ (i) follows from Theorem 3.2.4. □

Now, we introduce the new version of stuttering equivalence which, for finite state Kripke structures, can be proved to coincide with the original stuttering equivalence of Browne et al. [1988] and does not ignore divergence. The new version is defined in terms of the divergence blind one, and relies on adding to Kripke Structures a fresh state that is used as sink-state for deadlocked or divergent states.

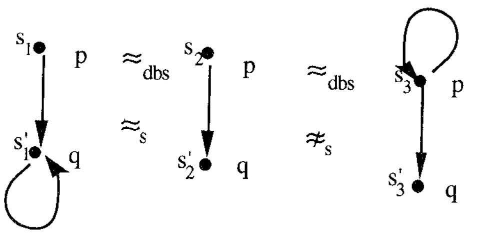
Definition 3.2.6 (*Extending Kripke structures with livelocked state*). Let $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ be a finite state Kripke structure, let s_0 be a state not in S and let p_0 be an atomic proposition such that for all $s \in S$ we have $p_0 \notin \mathscr{L}(s)$. Define the Kripke structure $L(\mathscr{K})$ by $L(\mathscr{K}) = (S', \mathscr{L}', \rightarrow)$ where

—$S' = S \cup \{s_0\}$,
—$\mathscr{L}' = \mathscr{L} \cup \{\langle s_0, \{p_0\}\rangle\}$ and
—$\rightarrow' = \rightarrow \cup \{(s, s_0) | s$ is on a cycle of states with the same label or has no outgoing edges$\}$.

Definition 3.2.7 (*Divergence sensitive stuttering equivalence*). Let $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ be a finite state Kripke structure.

(i) Two states $r, s \in S$ are *stuttering equivalent*, abbreviated $\mathscr{K}$: $r \approx_s s$, iff $L(\mathscr{K})$: $r \approx_{dbs} s$.
(ii) Two paths ρ, σ of $\mathscr{K}$ are *stuttering equivalent*, abbreviated $\mathscr{K}$: $\rho \approx_s \sigma$, iff $L(\mathscr{K})$: $\rho \approx_{dbs} \sigma$.

The next example illustrates the different stress the two equivalence put on divergence (infinite stuttering). Note that also divergence sensitive stuttering equivalence does not distinguish between deadlock and divergence; the equivalence is sensitive to any divergence except for that occurring in otherwise deadlocked states.



Example 3.2.8 (*Differences between* $\approx_s$ *and* $\approx_{dbs}$)

Let $\varphi = \forall \mathrm{F} q$. Then $s_1 \models_\mu \varphi$ and $s_2 \models_\mu \varphi$, whereas $s_3 \not\models_\mu \varphi, s_1 \not\models \varphi, s_2 \not\models \varphi$ and $s_3 \not\models \varphi$.

LEMMA 3.2.9. *Let $\mathscr{K} = (S, \mathscr{L}, \to)$ be a finite state Kripke structure, let $r, s \in S$ with $r \approx_s s$ and let $\rho \in \mu run(r)$. Then there exists a $\sigma \in \mu run(s)$ such that $\rho \approx_s \sigma$.*

PROOF. Given any maximal path ρ from r in $\mathscr{K}$ then it is also a path in $L(\mathscr{K})$. Since $L(\mathscr{K})$: $r \approx_{\mathrm{dbs}} s$, we can use Lemma 3.2.2 to find a path σ from s in $\mathscr{L}(\mathscr{K})$ which is equivalent to ρ. It must be that σ is also a path of $\mathscr{K}$ because if s_0 was in σ then the latter could never be related to ρ. If σ is maximal in $\mathscr{K}$, then we are done. Now, suppose that it is not; we have to distinguish whether ρ is finite or not.

In case ρ is finite, since it is also maximal it must be that $r' = \text{last}(\rho)$ is a deadlocked state. Thus, in $L(\mathscr{K})$, there is the transition (r', s_0). Let $s' = \text{last}(\sigma)$, since $\rho \approx_{\mathrm{dbs}} \sigma$ we have $L(\mathscr{K})$: $r' \approx_{\mathrm{dbs}} s'$. We can now rely on the fact that $\approx_{\mathrm{dbs}}$ is a divergence blind stuttering bisimulation to find, for some $n \geq 0$, a sequence $u_0 u_1 \cdots u_n$ such that $s' = u_0$ and for all $i < n$, $u_i \to u_{i+1}$ and $r' \approx_{\mathrm{dbs}} u_i$ and $s_0 \approx_{\mathrm{dbs}} u_n$. But this means that $u_n = s_0$ so that, in $\mathscr{K}, u_{n-1}$ is either deadlocked or occurs in a cycle of states with the same label. In case u_{n-1} is deadlocked, consider path $\sigma' = \sigma u_1 \cdots u_{n-2} u_{n-1}$. One can easily check that σ' is maximal in $\mathscr{K}$ and $\rho \approx_s \sigma'$. In case u_{n-1} occurs in a cycle of states with the same label, let the path $\pi = v_0 v_1 \cdots v_m$ be a cycle of states with the same label starting in u_{n-1} (i.e., $u_{n-1} = v_0, \forall i < m$: $v_i \to v_{i+1}$ and $v_m \to v_0$). One can easily show that all states in a cycle of states with the same label are divergence blind stuttering equivalent. Now consider the path σ'' obtained by concatenating $\sigma u_1 \cdots u_{n-2}$ with $(\pi)^\omega$; it is maximal and we have $\rho \approx_s \sigma''$.

The case of ρ infinite is dealt similarly and is left to the reader; it relies on the fact that $\mathscr{K}$ has only a finite number of states. □

THEOREM 3.2.10. *Let $\mathscr{K} = (S, \mathscr{L}, \to)$ be a finite state Kripke structure and let $r, s \in S$ with $r \approx_s s$. Then for every CTL*-X formula φ: $r \models_\mu \varphi$ iff $s \models_\mu \varphi$.*

PROOF. Copy the proof of the corresponding Theorem 3.2.3 for divergence blind stuttering equivalence and replace Lemma 3.2.2 by Lemma 3.2.9. □

THEOREM 3.2.11. *Let $\mathscr{K} = (S, \mathscr{L}, \to)$ be a finite state Kripke structure and let $s \in S$. Then there exists a CTL-X formula φ such that for all $r \in S$: $r \models_\mu \varphi$ iff $r \approx_s s$.*

PROOF. Similar to the proof of Theorem 3.2.4. Now, we apply the partition refinement algorithm on the structure $L(\mathscr{K})$. We associate a formula to each block different from $\{s_0\}$, which when interpreted over $\mathscr{K}$, only holds for the states in that block. The interesting case is the one where a block B with associated formula φ, is split into a block B_1 of states, from which after some stuttering there is a transition to s_0, and into a block $B_2 = B - B_1$. Now associate the formula $\varphi \wedge (\exists\, \mathrm{G}\, \varphi)$ to B_1 and the formula $\varphi \wedge (\forall\, \mathrm{F} \neg \varphi)$ to B_2. $\square$

By combining Theorem 3.2.10 and 3.2.11, we obtain the following results:

THEOREM 3.2.12 (*Stuttering, CTL*-X and CTL-X agree for* $\models_\mu$). *Let* $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ *be a finite state Kripke structure and let* $r, s \in S$. *Then the following statements are equivalent*:

(i) $r \approx_s s$,
(ii) *for every CTL*-X formula* φ: $r \models_\mu \varphi$ *iff* $s \models \mu\varphi$, *and*
(iii) *for every CTL-X formula* φ: $r \models_\mu \varphi$ *iff* $s \models_\mu \varphi$. $\square$

Since a similar result was proved in Browne et al. [1988], we have, as a corollary of the above theorem, that our version of stuttering equivalence coincides with that of Browne et al. [1988] for finite state Kripke structures without deadlocked states, that is, for the class of KSs they consider.

3.3. DIVERGENCE BLIND STUTTERING EQUIVALENCES AND BRANCHING BISIMULATIONS. In this section, we want to study the relationships between branching bisimulation and CTL*-X. We will do it, by exploiting the relationships between stuttering equivalence and this logic. Indeed, we get the new logical characterization of branching bisimulation by relating CTL*-X to the divergence blind stuttering equivalence studied above. We need some preliminary work that will enable us to relate the different structures on which branching and stuttering equivalence are defined; namely, Kripke Structures and Labeled Transition Systems.

We introduce a new kind of structure that can be projected naturally on both Labeled Transition Systems and Kripke structures. The new structures will be called Doubly Labeled Transition Systems (L^{2}TS).

Definition 3.3.1 (*Doubly Labeled Transition Systems*). An L^{2}TS is a structure $\mathscr{D} = (S, A, \rightarrow, \mathscr{L})$ where $(S, A, \rightarrow)$ is an LTS and $\mathscr{L}$: $S \rightarrow 2^{\mathrm{AP}}$ is a labeling function that associates a set of atomic propositions to each state. With LTS($\mathscr{D}$) we denote the substructure $(S, A, \rightarrow)$ of $\mathscr{D}$ and with KS($\mathscr{D}$) we denote the substructure $(S, \mathscr{L}, \rightarrow')$ of $\mathscr{D}$ where $r \rightarrow' s$ if and only if $\exists \alpha$: $r \xrightarrow{\alpha} s$. Equivalences defined on the states of an LTS or of a KS can be naturally lifted to L^{2}TS by ignoring either the labels of the states or the labels of the transitions:

—$\mathscr{D}$: $r \sim s \Leftrightarrow_{\mathrm{def}} \mathrm{LTS}(\mathscr{D})$: $r \sim s$
—$\mathscr{D}$: $r \sim' s \Leftrightarrow_{\mathrm{def}} \mathrm{KS}(\mathscr{D})$: $r \sim' s$.

The actual definition of L^{2}TS is too general for our interests. Indeed, the generalized transition systems which we need have also to guarantee a certain degree of consistency between the labels of two adjacent states and the labels of the transitions connecting these states. Because of this, we introduce the restricted class of consistent L^{2}TSs. Essentially, the restrictions amount to requiring that the states that are connected by invisible actions have the same labels and that the only difference between the labels o^c adjacent states connected by a visible transition be the information carried by the label of the transition connecting them.

Definition 3.3.2 (*Consistent L^2TSs*). A L^2TS $(S, A, \rightarrow, \mathscr{L})$ is *consistent* if there exists a function

—*action*: $2^{\mathbf{AP}} \times 2^{\mathbf{AP}} \rightarrow A_\tau$

such that, for any subsets P, Q, Q' of **AP**, we have:

(i) $action(P, P) = \tau$;
(ii) $action(P, Q) = action(P, Q')$ implies $Q = Q'$;
(iii) $r \xrightarrow{\alpha} s$ implies $\alpha = action(\mathscr{L}(r), \mathscr{L}(s))$.

The above restriction on L^2TSs, permits performing the first step toward relating branching bisimulation and CTL*-X; indeed, stuttering equivalence and branching bisimulation agree when they are defined on consistent L^2TSs.

THEOREM 3.3.3 (*Divergence blind stuttering and $\approx_b$ agree on consistent L^2TSs*). *If $\mathscr{D} = (S, A, \rightarrow, \mathscr{L})$ is a consistent L^2TS, then for all r, s in S*:

$$\mathscr{D}: r \approx_{dbs} s \text{ if and only if } \mathscr{L}(r) = \mathscr{L}(s) \text{ and } \mathscr{D}: r \approx_b s.$$

PROOF. Immediate from the definitions of the equivalences and from the consistency requirements on $\mathscr{D}$. □

We can now start studying the relationships between stuttering equivalence as defined on Kripke structures and branching bisimulation as defined on labeled transition systems. We set up general construction that given a labeled transition system or a Kripke structure yields an enriched system, that has a structure similar to the original one, but carries labels on both states and transitions. It is worth remarking that one of the main sources of problems in these transformations is the presence of invisible actions.

First of all, we present a straightforward way of labeling the transitions in a Kripke structure in such a way that divergence blind stuttering equivalence in the original structure coincides with branching bisimulation equivalence in the enriched structure.

Definition 3.3.4 (*From KSs to L^2TSs*). Let $\mathscr{K} = (S, \mathscr{L}, \rightarrow)$ be a Kripke structure. The L^2TS $tr(\mathscr{K})$ is defined as $(S, 2^{\mathbf{AP}}, \rightarrow', \mathscr{L})$ where

—$r \xrightarrow{\tau}{}' s$ if and only if $r \rightarrow s$ and $\mathscr{L}(r) = \mathscr{L}(s)$;
—$r \xrightarrow{1}{}' s$ if and only if $r \rightarrow s$ and $\mathscr{L}(r) \neq \mathscr{L}(s)$ and $\mathscr{L}(s) = 1$.

In Figure 5, we present an example of the above defined construction.

One can easily verify that $tr(\mathscr{K})$ is consistent and moreover that KS$(tr(\mathscr{K})) = \mathscr{K}$. Theorem 3.3.5 is an immediate consequence of these properties.

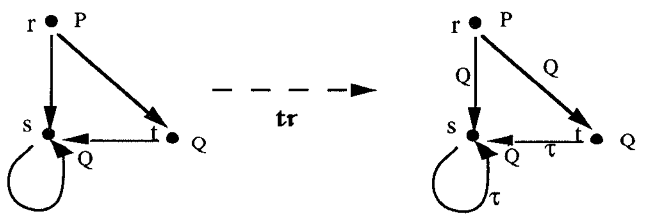


FIG. 5. An example translation from KS to L^2TS.

THEOREM 3.3.5. *Let $\mathscr{K} = (S, \mathscr{L}, \to)$ be a Kripke structure. Then for all r, s in S*:

$$\mathscr{K}\colon r \approx_{dbs} s \text{ if and only if } \mathscr{L}(r) = \mathscr{L}(s) \text{ and } \mathscr{lr}(\mathscr{K})\colon r \approx_b s.$$

PROOF. By the above property, we have that $\mathscr{K}$: $r \approx_{\mathrm{dbs}} s$ if and only if KS($\mathscr{lr}(\mathscr{K})$): $r \approx_{\mathrm{dbs}} s$. By definition KS($\mathscr{lr}(\mathscr{K})$): $r \approx_{\mathrm{dbs}} s$ if and only if $\mathscr{lr}(\mathscr{K})$: $r \approx_{\mathrm{dbs}} s$. Now, since $\mathscr{lr}(\mathscr{K})$ is consistent, it follows from Theorem 3.3.3 that the latter holds if and only if $\mathscr{L}(r) = \mathscr{L}(s)$ and $\mathscr{lr}(\mathscr{K})$: $r \approx_{\mathrm{b}} s$. □

The construction of a L^{2}TS from a LTS is less straightforward than the construction starting from a KS. The first idea that comes to mind is to label a state with the label of a transition leading to it, if the label is visible, and with the label of the source state of the transition otherwise. This would not deal properly with situations where a state is accessed by transitions with different labels. Moreover, problems arise with structures like those reported in Figure 6 that capture a very different intuition but would be identified by the outlined naive transformation.

One possible solution is proposed in Clarke et al. [1989]. There, a given LTS is extended by labeling each state with the set of the labels of the paths which lead to it; paths are labeled by the set of those actions which are performed an odd number of times. Unfortunately, this construction does not always lead to consistent L^{2}TSs and is not able to cope with systems whose states can be reached via two paths which contain the same action an even resp. an odd number of times. Indeed, the authors restrict attention to those LTSs that lead to unique labeling and this restricted class of LTSs gives rise to consistent L^{2}TSs only.

The LTS in the left hand side of Figure 7 shows that in general it is not possible to give a labeling of the states in an LTS such that the resulting L^{2}TS is consistent. Thus, there exists no transformation function which preserves the structure of the LTS up to isomorphism.

In De Nicola and Vaandrager [1990b], we presented a transformation that gives rise to L^{2}TS that have isomorphic unfolding with the original transition systems; that transformation has the disadvantage that it might lead to a quadratic blowup in the size of the system (unless one assumes that the alphabet A is finite and fixed).

Below, we describe an alternative transformation from LTS to L^{2}TS that is linear in the number of states and transitions. It was suggested in Emerson and Lei [1984] where it is presented in a setting without silent actions. The price to be paid for choosing this transformation is that corresponding states in the LTS and the L^{2}TS do no longer give rise to isomorphic unfoldings. However, the structure of the LTS and the L^{2}TS are still very similar: the L^{2}TS is obtained by placing a new state in the middle of each visible transition of the LTS.

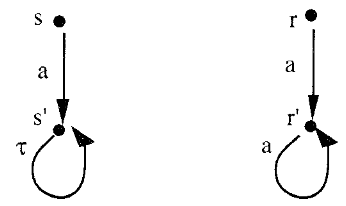


FIG. 6. A pair of transition systems critical for transformations.

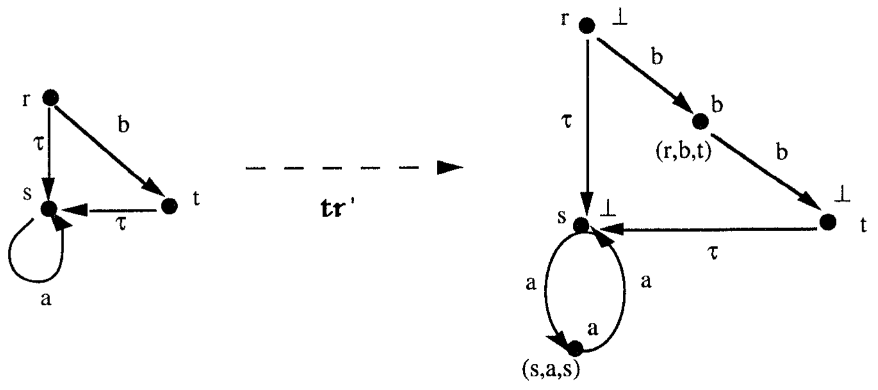


FIG. 7. An example translation from LTS to L^2TS.

Definition 3.3.6 (*From LTSs to consistent L^2TSs*). Let $\mathscr{A} = (S, A, \to)$ be an LTS. The L^2TS $tr'(\mathscr{A})$ is defined as $(S', A, \to', \mathscr{L})$ where

—$S' = S \cup \{(r, a, s) | a \in A \text{ and } r \xrightarrow{a} s\}$;
—$\to' = \{(r, \tau, s) | r \xrightarrow{\tau} s\} \cup \{(r, a, (r, a, s)) | r \xrightarrow{a} s\} \cup \{((r, a, s), a, s) | r \xrightarrow{a} s\}$;
—For $r, s \in S$ and $a \in A$: $\mathscr{L}(s) = \{\perp\}$ and $\mathscr{L}((r, a, s)) = \{a\}$.

It is immediate from the definitions that $tr'(\mathscr{A})$ is a consistent L^2TS. We report in Figure 7 an example of the above-defined construction.

PROPOSITION 3.3.7. *Two states r, s in an LTS $\mathscr{A}$ are branching bisimilar if and only if they are so in $tr'(\mathscr{A})$.*

PROOF. Let $\mathscr{A} = (S, A, \to)$ be a LTS and $tr'(\mathscr{A})$ be its translation.
First, suppose $\mathscr{A}$: $r \approx_b s$. Then there exists a branching bisimulation R on $\mathscr{A}$ with $\langle r, s\rangle \in R$. Define relation R' on S' by

$$R' = R \cup \{\langle (t, a, t'), (u, a, u')\rangle \in \to^2 | a \in A, t \mathrm{R} u \textit{ and } t' R u'\}.$$

It is routine to check that R' is a branching bisimulation on $tr'(\mathscr{A})$; from this $tr'(\mathscr{A})$: $r \approx_b s$ follows. For the other direction, suppose $tr'(\mathscr{A})$: $r \approx_b s$. Then there is a branching bisimulation on $tr'(\mathscr{A})$ with $\langle r, s\rangle \in R$. Define

$$R' = R \cap (S \times S).$$

Again, it is routine to show that R' is a branching bisimulation on $\mathscr{A}$, and that $\mathscr{A}$: $r \approx_b s$. □

Now, Proposition 3.3.7 and Theorem 3.3.3, together with Theorem 3.2.5, allow us to prove the following important theorem which says that, via transformation tr', CTL*-X can be viewed as a logic for branching bisimulation equivalence.

THEOREM 3.3.8. *Let $\mathscr{A} = (S, A, \to)$ be a finite LTS. Then for all r, s in S:*

$$\mathscr{A}: r \approx_b s \textit{ if and only if } \forall \varphi \in CTL^*\text{-}X,\ tr'(\mathscr{A}), r \models \varphi \Leftrightarrow tr'(\mathscr{A}), s \models \varphi.$$

Clearly, due to Theorem 3.2.5, we can also replace CTL*-X with CTL-X in the above theorem.

Example 3.3.9 (*CTL*-X can distinguish weakly equivalent states*). Consider the LTS $\mathscr{A}_{2.3.3}$ of Example 2.3.3.

If we define $\varphi = \exists(\exists Fb) \,\mathrm{U}\, a$, then $\mathscr{KS}'(\mathscr{A}_{2.3.3}), s \models \varphi$ but $\mathscr{KS}'(\mathscr{A}_{2.3.3}), r \not\models \varphi$.

If we define $\varphi' = \exists((a \Rightarrow \forall G \neg c) \,\mathrm{U}\, b)$, then $\mathscr{KS}'(\mathscr{A}_{2.3.3}), q \models \varphi$ but $\mathscr{KS}'(\mathscr{A}_{2.3.3})$, $p \not\models \varphi'$.

3.4. Stuttering Equivalences and Divergence Sensitive Branching Bisimulation.

We conclude this chapter by introducing a new version of branching bisimulation that for finite systems is in full agreement with the stuttering equivalence of Browne et al. [1988] and thus with the equivalence induced by the standard interpretation of CTL* and CTL without the next-time operator, when interpreted over maximal paths. What we need is nothing more than a divergence sensitive version of the original definition of Section 2. We pedantically follow the approach we took to define stuttering equivalence from its divergence blind version.

Definition 3.4.1 (*Extending L^2TSs and LTSs with livelocked state*).

(i) Let $\mathscr{D} = (S, A, \rightarrow, \mathscr{L})$ be a finite $\mathrm{L^2TS}$, let s_0 be a fresh state, a_0 be a fresh action and let p_0 be a fresh atomic proposition. The $\mathrm{L^2TS}$ with livelocked states $L(\mathscr{D}) = (S', A', \rightarrow', \mathscr{L}')$ is given by:

—$S' = S \cup \{s_0\}$,
—$A' = A \cup \{a_0\}$,
—$\rightarrow' = \rightarrow \cup \{\langle s, a_0, s_0\rangle \mid s$ occurs in a τ-cycle or has no outgoing transitions$\}$ and
—$\mathscr{L}' = \mathscr{L} \cup \langle\{s_0, \{p_0\}\rangle\}$.

(ii) Let $\mathscr{A} = (S, A, \rightarrow)$ be a finite LTS. The LTS with livelocked states $L(\mathscr{A})$ is given by $\mathrm{LTS}(L(\mathscr{D}'))$ where $\mathscr{D}'$ is the $\mathrm{L^2TS}(S, A, \rightarrow, \emptyset)$.

The following facts are immediate from the above definitions and from Definition 3.2.6.

LEMMA 3.4.2. *Let $\mathscr{D} = (S, A, \rightarrow, \mathscr{L})$ be a finite and consistent L^2TS. Then*

(i) *$L(\mathscr{D})$ is a finite and consistent L^2TS;*
(ii) *$LTS(L(\mathscr{D})) = L(LTS(\mathscr{D}))$;*
(iii) *$KS(L(\mathscr{D})) = L(KS(\mathscr{D}))$.*

Definition 3.4.3 (*Divergence sensitive branching bisimulation*). Let $\mathscr{A} = (S, A, \rightarrow)$ be a finite LTS. Two states r, s in S are *divergence sensitive branching bisimilar*, abbreviated $(\mathscr{A}\colon)\ r \approx_{\mathrm{dsb}} s$, if and only if $L(\mathscr{A})\colon r \approx_{\mathrm{b}} s$.

THEOREM 3.4.4 (STUTTERING EQUIVALENCE AND $\approx_{\mathrm{DSB}}$ AGREE ON CONSISTENT $\mathrm{L^2}$TSs). *Let $\mathscr{D} = (S, A, \rightarrow, \mathscr{L})$ be a finite and consistent L^2TS then for all r, s in S:*

$$\mathscr{D}\colon r \approx_{\mathrm{s}} s \textit{ if and only if } \mathscr{D}\colon r \approx_{\mathrm{dsb}} s \textit{ and } \mathscr{L}(r) = \mathscr{L}(s).$$

PROOF. The theorem follows from the following chain of double implications:

$\mathscr{D}$: $r \approx_s s$ $\Leftrightarrow$ (Definition 3.3.1)
KS($\mathscr{D}$): $r \approx_s s$ $\Leftrightarrow$ (Definition 3.2.7)
L(KS($\mathscr{D}$)): $r \approx_{\text{dbs}} s$ $\Leftrightarrow$ (Lemma 3.4.2 (iii))
KS(L($\mathscr{D}$)): $r \approx_{\text{dbs}} s$ $\Leftrightarrow$ (Definition 3.3.1)
L($\mathscr{D}$): $r \approx_{\text{dbs}} s$ $\Leftrightarrow$ (Lemma 3.4.2 (i) and Theorem 3.3.3)
L($\mathscr{D}$): $r \approx_{\text{b}} s$ and $\mathscr{L}(r) = \mathscr{L}(s)$ $\Leftrightarrow$ (Definition 3.3.1)
LTS(L($\mathscr{D}$)): $r \approx_{\text{b}} s$ and $\mathscr{L}(r) = \mathscr{L}(s)$ $\Leftrightarrow$ (Lemma 3.4.2 (ii))
L(LTS($\mathscr{D}$)): $r \approx_{\text{b}} s$ and $\mathscr{L}(r) = \mathscr{L}(s)$ $\Leftrightarrow$ (Definition 3.4.3)
LTS($\mathscr{D}$): $r \approx_{\text{dsb}} s$ and $\mathscr{L}(r) = \mathscr{L}(s)$ $\Leftrightarrow$ (Definition 3.3.1)
$\mathscr{D}$: $r \approx_{\text{dsb}} s$ and $\mathscr{L}(r) = \mathscr{L}(s)$. □

The final theorem states that for our transformation $\mathscr{t\ell}'$ from LTSs to L^{2}TSs, divergence sensitive branching bisimulation equivalence coincides with the equivalence induced by CTL*-X under the standard interpretation over maximal paths.

LEMMA 3.4.5. *Let $\mathscr{A} = (S, A, \rightarrow)$ be a finite LTS. Then for all r, s in S*:

$$\mathscr{A}: r \approx_{dsb} s \text{ if and only if } \mathscr{t\ell}'(\mathscr{A}): r \approx_{dsb} s.$$

PROOF. Similar to the proof of Proposition 3.3.7. □

THEOREM 3.4.6. *Let $\mathscr{A} = (S, A, \rightarrow)$ be a finite LTS. Then for all r, s in S*:

$$\mathscr{A}: r \approx_{dsb} s \text{ if and only if } \forall \varphi \in CTL^*\text{-}X,\ \mathscr{t\ell}'(\mathscr{A}), r \models_\mu \varphi \Leftrightarrow \mathscr{t\ell}'(\mathscr{A}), s \models_\mu \varphi.$$

PROOF

$\mathscr{A}$: $r \approx_{\text{dsb}} s$ $\Leftrightarrow$ (Lemma 3.4.5)
$\mathscr{t\ell}'(\mathscr{A})$: $r \approx_{\text{dsb}} s$ $\Leftrightarrow$ (States in S have label $\{\perp\}$)
$\mathscr{t\ell}'(\mathscr{A})$: $r \approx_{\text{dsb}} s$ and $\mathscr{L}(r) = \mathscr{L}(s)$ $\Leftrightarrow$ (Theorem 3.4.4)
$\mathscr{t\ell}'(\mathscr{A})$: $r \approx_s s$ $\Leftrightarrow$ (Theorem 3.2.12)
$\forall \varphi \in$ CTL*-X, $\mathscr{t\ell}'(\mathscr{A}), r \models_\mu \varphi \Leftrightarrow \mathscr{t\ell}'(\mathscr{A}), s \models_\mu \varphi$. □

As always, we can replace CTL*-X with CTL-X in the above theorem.

4. *Conclusions and Related Work*

In this paper, we have introduced three significantly different logics that are in full agreement with branching bisimulation equivalence ($\approx_{\text{b}}$). The first logic, L_{U}, is an extension of Hennessy–Milner Logic [Hennessy and Milner 1985] with a kind of "until" operator; it is close in spirit with the actual definition of branching bisimulation in van Glabbeek and Weijland [1989/1991]. The second logic, L_{BF}, is another extension of Hennessy–Milner Logic that exploits the power of backward modalities; it stems directly from the alternative characterization of branching bisimulation presented in De Nicola et al. [1990b]. The third logic is CTL* (see, e.g., Emerson and Srinivasan [1989]) without the next-time operator (CTL*-X).

The latter characterization exploits the relationships between variants of stuttering equivalence [Browne et al. 1988] and CTL*. We have established that branching bisimulation equivalence is in full agreement with CTL*-X by proving that it is in full agreement with a divergence blind variant of stuttering equivalence. The actual proof had to face the problem that the two equivalences are defined on different structures; namely, Kripke structures (KSs) and Labeled Transition Systems (LTSs). Thus, transformation functions from one structure to the other were needed.

We defined a transformation function from general LTSs with invisible labels to KSs. This transformation permits naturally relating branching bisimulation to divergence blind stuttering equivalence and divergence sensitive branching bisimulation to stuttering equivalence. The transformation enjoys two important properties; it is linear in the size of the systems and preserves a close correspondence between the source and the target system. We defined also a translation from KSs to LTSs, which enjoys the same properties of that from LTSs to KSs. These two transformations and their properties permit one to move freely between an action-based and a state-based view of concurrent system and to use automatic tools that have been designed for reasoning within either model.

To facilitate the discussion, we also introduce a new kind of structures, Doubly Labeled Transition Systems (L^2TS), which were used as target of the translation functions. We proved that branching bisimulation and stuttering equivalence are in full agreement on a subclass of L^2TS in which a strong consistency relation between the labels of the nodes and those of the incoming and outgoing arcs holds and have proved that our translations always yield consistent L^2TS. Here, we want to remark that the new structures are interesting in their own in that they permit richer descriptions of systems. It is certainly worthwhile exploring how much the consistency constraint on L^2TS can be relaxed while keeping full agreement between the state-based and the action-based equivalence. It would also be interesting studying the equivalences that are obtained once consistency is lost.

The philosophy behind the backward generalization of HML is very similar to that of the logic called J_T, introduced by Hennessy and Stirling [1985] to deal with noncontinuous properties of generalized transition systems with infinite computations. The relevant difference is that L_{BF} permits abstracting from silent actions, while J_T does not. Indeed, in the context of traditional (limit closed) labeled transition systems, J_T has no more discriminating power than strong observational equivalence (see also De Nicola et al. [1990b]). The characterization of $\approx_b$ in terms of a more abstract version of J_T gives strength to the claims that branching bisimulation is indeed a natural generalization of strong bisimulation and that it can be easily extended to cope with infinitary properties.

Stirling [1989] provides a different interpretation of CTL*-X based on LTS extended with the double arrow relation $\overset{\alpha}{\Rightarrow}$; he shows that weak bisimulation and the newly interpreted CTL* (with the next-time operator) are in full agreement. This result is weaker than ours and is a direct consequence of the fact that strong bisimulation and CTL* are in full agreement.

In the literature, various translations functions between Kripke Structures and Labeled Transition Systems have been proposed. Our transformation function from general LTSs to KSs was inspired by Emerson and Lei [1985]; we generalized their proposal to deal with invisible labels. The transformation function from KSs to LTSs was independently proposed by Koutny [1991]. In Jonsson et al. [1990], a LTS is translated into a KS by introducing, in correspondence of each transition in the LTS, a state in the associated KS with the obvious labeling, and by introducing a transition in correspondence of each pair of consecutive transitions in the LTS. This construction does not permit relating stuttering equivalence and branching bisimulation. Another translation from LTSs to KSs is described in Clarke et al. [1989] but their construction only works for a special class of LTSs which does not allow systems with states reachable via two paths which contain the same action respectively, an even and an odd number of times. In the paper by De Nicola and Vaandrager [1990b], we have proposed another transformation that has the advantage over all the others that it preserves the structure of the source LTS up to isomorphism of unfoldings; it however might lead to a quadratic blowup in the size of the system (unless one assumes that the alphabet A is finite and fixed).

The problem of the complete axiomatization of L_U and L_{BF} is still open, and so are the satisfiability and model checking problems for these logics. We think that, by expressing L_U within the modal mu-calculus of Kozen [1983] and Pratt [1981] we can answer the latter questions for the logic L_U; the complexity of the translation algorithm is, however, not clear to us. Since the translations of Laroussinie et al. [1993] are effective, the connection with the modal mu-calculus would also give satisfiability and model checking algorithms for L_{BF}. However, since again the complexity of the translation algorithm is unclear, and possibly quite high, this might not be the most efficient route.

Appendix A

LEMMA A.1 (Lemma 2.1.6). *Let $\mathscr{A} = (S, A, \rightarrow)$ be an LTS and let $s_0 \tau s_1 \tau \cdots s_{n-1} \tau s_n$, $n > 0$, be a run in $\mathscr{A}$ with $s_0 \approx_b s_n$. Then for all $0 \le i \le n$: $s_0 \approx_b s_i$.*

PROOF. Define for $i > 0$.

$$\mathscr{R}_0 = \approx_b$$

$$\mathscr{R}_i = \mathscr{R}_{i-1} \cup \left\{(r, r'), (r', r) | \exists r'': r \overset{\epsilon}{\Rightarrow} r' \overset{\tau}{\rightarrow} r'' \,\&\, r \mathscr{R}_{i-1} r''\right\}$$

$$\mathscr{R}_\omega = \cup_{i < \omega} \mathscr{R}_i$$

First, we show that $\mathscr{R}_\omega$ enjoys the property that we want to prove for $\approx_b$. Let for some $n > 0$, $r_0 \tau r_1 \cdots r_{n-1} \tau r_n$ be a run with $r_0 \mathscr{R}_\omega r_n$. Indeed, we can prove by induction on n that for all $0 \le i \le n$: $r_0 \mathscr{R}_\omega r_i$. If $n = 1$, the statement is trivially correct. Now consider the case $n > 1$. Since $r_0 \mathscr{R}_\omega r_n$, there exists an $m < \omega$ with $r_0 \mathscr{R}_m r_n$. By definition of $\mathscr{R}_{m+1}$: $r_0 \mathscr{R}_{m+1} r_{n-1}$. Thus, $r_0 \mathscr{R}_\omega r_{n-1}$ and, by induction hypothesis, $r_0 \mathscr{R}_\omega r_i$ for all $0 \le i \le n - 1$; this together with the hypothesis proves the claim.

Next, we will prove with induction that, for every $n < \omega$, $\mathscr{R}_n$ is a branching bisimulation. This would suffice to conclude that $\mathscr{R}_\omega$ is a branching bisimulation and that $\mathscr{R}_\omega \subseteq \approx_b$. But, by construction, we have $\approx_b \subseteq \mathscr{R}_\omega$. Hence, $\approx_b = \mathscr{R}_\omega$, and we have proved the lemma. $\mathscr{R}_0$ is a branching bisimulation because $\approx_b$ is such. Now, suppose that, for certain $n > 0$, $\mathscr{R}_{n-1}$ is a branching bisimulation. We prove that $\mathscr{R}_n$ is a branching bisimulation too. By construction $\mathscr{R}_n$ is symmetric. Suppose $r \mathscr{R}_n r'$ and $r \overset{\alpha}{\rightarrow} s$. If $r \mathscr{R}_{n-1} r'$, then the transfer property is trivially fulfilled. In the other case, we have that there exist $r''\mathscr{R}_{n-1}r$ which is reachable via a single τ transition either from r or r'. More specifically, we have two possibilities:

(1) For some r'': $r \overset{\epsilon}{\Rightarrow} r' \xrightarrow{\tau} r''$ and $r\,\mathscr{R}_{n-1}\,r''$. Using $r\,\mathscr{R}_{n-1}\,r''$, a first possibility is that $\alpha = \tau$ and $s\,\mathscr{R}_{n-1}\,r''$. But this means that $r' \overset{\epsilon}{\Rightarrow} r' \xrightarrow{\tau} r''$ with $r\,\mathscr{R}_n\,r'$ and $s\,\mathscr{R}_n\,r''$. Otherwise, there are r_1, r_2 such that $r'' \overset{\epsilon}{\Rightarrow} r_1 \xrightarrow{\alpha} r_2, r\,\mathscr{R}_{n-1}\,r_1$ and $s\,\mathscr{R}_{n-1}\,r_2$. But then $r' \overset{\epsilon}{\Rightarrow} r_1 \xrightarrow{\alpha} r_2, r\,\mathscr{R}_n\,r_1$ and $s\,\mathscr{R}_n\,r_2$.

(2) For some r'': $r' \overset{\epsilon}{\Rightarrow} r \xrightarrow{\tau} r''$ and $r'\,r_{n-1}\,r''$. Then $r' \overset{\epsilon}{\Rightarrow} r \xrightarrow{\alpha} s, r\,\mathscr{R}_n\,r$ and $s\,\mathscr{R}_n\,s$. □

LEMMA A.2 (*X-property*). *Let $(S, A, \rightarrow)$ be an LTS and let $r, s \in S$ with $r \approx_{bf} s$. Let $\mathscr{R} \subseteq run_{\mathscr{A}} \times run_{\mathscr{A}}$ be the maximal weak back-and-forth bisimulation between r and s. Then $\mathscr{R}$ has the following X-property*:

$$\forall \rho, \rho \in run(r), \forall \sigma, \sigma \in run(s):$$
$$\left[\rho \overset{\epsilon}{\Rightarrow} \rho', \sigma \overset{\epsilon}{\Rightarrow} \sigma', \rho\,\mathscr{R}\,\sigma' \,\&\, \rho'\,\mathscr{R}\,\sigma\right] \Rightarrow \rho'\,\mathscr{R}\,\sigma'.$$

PROOF. Define relation

$$\mathscr{R}' = \mathscr{R} \cup \Big\{(\rho', \sigma'), (\sigma', \rho') \mid \rho \in run(r), \sigma \in run(s):$$
$$\rho \overset{\epsilon}{\Rightarrow} \rho', \sigma \overset{\epsilon}{\Rightarrow} \sigma', \rho\,\mathscr{R}\,\sigma' \,\&\, \rho'\,\mathscr{R}\,\sigma\Big\}$$

We prove that $\mathscr{R}'$ is a weak back-and-forth bisimulation. Since $\mathscr{R}$ is the maximal back-and-forth bisimulation and $\mathscr{R} \leq \mathscr{R}'$ by construction, the fact that $\mathscr{R}'$ is a weak back-and-forth bisimulation would imply that $\mathscr{R} = \mathscr{R}'$. Thus, $\mathscr{R}$ has the X-property.

Clearly, $\mathscr{R}'$ is symmetric. Moreover, $r\mathscr{R}$'s because $r\,\mathscr{R}\,s$. Suppose $\rho'\,\mathscr{R}'\,\sigma'$ with $\rho'' \in$ run(r) and $\sigma' \in$ run(s). If $\rho'\mathscr{R}\sigma'$, then the back-and-forth conditions 2 and 3 are trivially fulfilled. Otherwise, there must be a ρ and a σ such that: $\rho \overset{\epsilon}{\Rightarrow} \rho'$, $\sigma \overset{\epsilon}{\Rightarrow} \sigma'$, $\rho\,\mathscr{R}\,\sigma'$, and $\rho'\,\mathscr{R}\,\sigma$. We check transfer property 2. Suppose $\rho' \overset{k}{\Rightarrow} \rho''$. Then $\rho \overset{k}{\Rightarrow} \rho''$. Since $\rho\,\mathscr{R}\,\sigma'$, there exists an σ'' such that $\sigma' \overset{k}{\Rightarrow} \sigma''$ and $\rho''\,\mathscr{R}'\,\sigma''$. Next we check transfer property 3. If $\rho'' \overset{k}{\Rightarrow} \rho'$, then, since $\rho'\,\mathscr{R}\,\sigma$, there exists an σ'' such that $\sigma'' \overset{k}{\Rightarrow} \sigma$ and $\rho''\,\mathscr{R}'\,\sigma''$. Now observe that $\sigma'' \overset{k}{\Rightarrow} \sigma'$.

The remaining case that $\sigma'\,\mathscr{R}'\,\rho'$ with $\sigma' \in$ run(s) and $\rho' \in$ run(r) is symmetric. □

THEOREM A.3 (*Theorem* 2.4.2). *Let $\mathscr{A} = (S, A, \rightarrow)$ be an LTS. Then for r, s in S: $\mathscr{A}$: $r \approx_b s$ if and only if $\mathscr{A}$: $r \approx_{bf} s$.*

PROOF

"$\Leftarrow$" Suppose $r \approx_b s$. Let cct be the mapping that associates to each run ρ in $\mathscr{A}$ its *concrete colored trace*, that is, the sequence which is obtained from ρ by replacing each state by its branching bisimulation equivalence class. So

$$\mathrm{cct}(s_0\alpha_1 s_1 \cdots s_{n-1}\alpha_n s_n) = (s_0/\approx_b\ \alpha_1 s_1/\approx_b \cdots s_{n-1}/\approx_b\ \alpha_n s_n/\approx_b).$$

Let ct be the mapping that associates to each run ρ in $\mathscr{A}$ its (*abstract*) *colored trace*, that is, the sequence which is obtained from cct(ρ) by removing all elements (C, τ, C) from the sequence. Define relation $\mathscr{R}$ by

$$\mathscr{R} = \{(\rho, \sigma), (\sigma, \rho) | \rho \in \text{run}(r),\ \sigma \in \text{run}(s)\ \&\ \text{ct}(\rho) = \text{ct}(\sigma)\}.$$

Using the stuttering lemma (Lemma 2.1.6), it is straightforward to check that $\mathscr{R}$ is a back and forth bisimulation between r and s.

"$\Leftarrow$" Suppose $r \approx_{\text{bf}} s$. Let $\mathscr{R} \subseteq \text{run}_{\mathscr{A}} \times \text{run}_{\mathscr{A}}$ be the maximal back-and-forth bisimulation between r and s. Define

$$\mathscr{R}' = \{(\text{last}(\rho), \text{last}(\sigma)) | \rho\, \mathscr{R}\, \sigma\}.$$

Clearly $r\, \mathscr{R}'\, s$. We show that $\mathscr{R}'$ is a branching bisimulation. $\mathscr{R}'$ is symmetric because $\mathscr{R}$ is. Suppose $r_0\, \mathscr{R}'\, s_0$. Then there are ρ, σ with $\rho\, \mathscr{R}\, \sigma$, last$(\rho) = r_0$ and last$(\sigma) = s_0$. Suppose that $r_0 \xrightarrow{\alpha} r'$ and let $\rho' = \rho\, \alpha\, r'$. In the proof of the transfer property, we distinguish between two cases.

(1) $\alpha \neq \tau$. Since $\rho \xrightarrow{\alpha} \rho'$ and $\rho\, \mathscr{R}\, \sigma$, there exist $\sigma_1, \sigma_2, \sigma'$ such that $\sigma \overset{\epsilon}{\Rightarrow} \sigma_1 \xrightarrow{\alpha} \sigma_2 \overset{\epsilon}{\Rightarrow} \sigma'$ and $\rho'\, \mathscr{R}\, \sigma'$. Since $\sigma_2 \overset{\epsilon}{\Rightarrow} \sigma'$, there exists a ρ_1 such that $\rho_1 \overset{\epsilon}{\Rightarrow} \rho_1\, \mathscr{R}\, \sigma_2$. But since the last transition of ρ' has label α, $\rho_1 = \rho'$ so that $\rho'\, \mathscr{R}\, \sigma_2$. Because $\sigma_1 \xrightarrow{\alpha} \sigma_2$, there exists a ρ_2 such that $\rho_2 \overset{\epsilon}{\Rightarrow} \rho \xrightarrow{\alpha} \rho'$ and $\rho_2\, \mathscr{R}\, \sigma_1$. How use that $\mathscr{R}$ has the X-property (Lemma A.2) to obtain $\rho\, \mathscr{R}\, \sigma_1$. But this gives us the transfer property; namely we have

$$s_0 \overset{\epsilon}{\Rightarrow} \text{last}(\sigma_1) \xrightarrow{\alpha} \text{last}(\sigma_2),\ r_0\, \mathscr{R}'\, \text{last}(\sigma_1) \text{ and } r'\, \mathscr{R}'\, \text{last}(\sigma_2).$$

(2) $\alpha = \tau$. Since $\rho \overset{\epsilon}{\Rightarrow} \rho'$ and $\rho\, \mathscr{R}\, \sigma$, there is an $n \geq 0$ and there are σ_i for $0 \leq i \leq n$ such that $\sigma_0 = \sigma$, for $0 < i \leq n$: $\sigma_{i-1} \xrightarrow{\tau} \sigma_i$, and $\rho'\, \mathscr{R}\, \sigma_n$. If $n = 0$, then $r'\, \mathscr{R}'\, s_0$ and we have proved the transfer property. If $n > 0$, then we can go back with an ϵ-move from σ_n to σ_{n-1}. A first possibility is that ρ' can simulate this step by doing nothing: $\rho'\, \mathscr{R}\, \sigma_{n-1}$. If this is the case, then either $n = 1$ and we are ready, or we can go back one more ϵ-step from σ_{n-1} to σ_{n-2}. Repeating this, we either find $\rho'\, \mathscr{R}\, \sigma_0$, in which case we have proved the transfer property for branching bisimulation since $r'\, \mathscr{R}'\, s_0$, or, for some $m > 0$ with $\rho'\, \mathscr{R}\, \sigma_m$, we have that a backward step to σ_{m-1} is simulated by a backward step $\rho_1 \overset{\epsilon}{\Rightarrow} \rho \xrightarrow{\tau} \rho'$ with $\rho_1\, \mathscr{R}\, \sigma_{m-1}$. In this case we use the X-property (Lemma A.2) to obtain $\rho\, \mathscr{R}\, \sigma_{m-1}$. This gives us the transfer property for branching bisimulation since:

$$s_0 \overset{\epsilon}{\Rightarrow} \text{last}(\sigma_{m-1}) \xrightarrow{\tau} \text{last}(\sigma_m),$$
$$r_0\, \mathscr{R}'\, \text{last}(\sigma_{m-1}) \text{ and } r'\, \mathscr{R}'\, \text{last}(\sigma_m). \qquad \square$$

ACKNOWLEDGMENTS. The original idea of using a kind of until operator for characterizing branching bisimulation is due to Rob van Glabbeek. Electronic correspondence with Orna Grümberg helped in understanding the relationships between stuttering and weak bisimulation. The paper has also benefitted from discussions with Peter van Emde Boas, Allen Emerson, Ugo Montanari, Amir Pnueli, and Colin Stirling.

REFERENCES

ABRAMSKY, S. 1989. Observation equivalence as a testing equivalence. *Theoret. Comput. Sci. 53*, 225–241.

AKKERMAN, G. J., AND BAETEN, J. C. M. 1990. Term rewriting analysis in process algebra. Report P9006. Programming Research Group, Univ. Amsterdam, Amsterdam, the Netherlands.

BAETEN, J. C. M. 1990. Applications of process algebra. Cambridge Tract in Theoretical Computer Science, Vol. 17. Cambridge University Press, Cambridge, England.

BAETEN, J. C. M., AND WEIJLAND, W. P. 1990. Process Algebra. Cambridge Tract in Theoretical Computer Science, Vol. 18. Cambridge University Press, Cambridge England.

BLOOM, B., ISTRAIL, S., AND MEYER, A. R. 1988/1990. Bisimulation can't be traced: Preliminary report. In *Conference Record of the 15th ACM Symposium on Principles of Programming Languages (POPL)* (San Diego, Calif.), ACM, New York, pp. 229–239. Full version appeared as Tech. Rep. TR 90-1150, Cornell Univ., Ithaca, N.Y., 1990.

BOLOGNESI, T., VAN DE LANGMAAT, J. AND VISSERS, C. 1995. Lotosphere: software development with LOTUS, Kluver Dordrect.

BROWNE, M. C., CLARKE, E. M., AND GRÜMBERG, O. 1988. Characterizing finite Kripke structures in propositional temporal logic. *Theoret. Comput. Sci. 59*, 1, 2, 115–131.

CLARKE, E. M., EMERSON, E. A., AND SISTLA, A. P. 1986. Automatic verification of finite state concurrent systems using temporal logic specifications. *ACM Trans. Prog. Lang. Syst. 8*, 2 (Apr.), 244–263.

CLARKE, E. M., LONG, D. E., AND MACMILLAN, K. L. 1989. Compositional model checking. In *Proceedings of the 4th Annual Symposium on Logic in Computer Science (LICS)* (Asilomar, Calif.). IEEE Computer Society Press, Washington, D.C., pp. 353–362.

CLEAVELAND, R., PARROW, J., AND STEFFEN, B. 1990. The concurrency workbench. In *Automatic Verification Methods for Finite State Systems* J. Sifakis, ed. Lecture Notes in Computer Science, vol. 407. Springer-Verlag, New York, pp. 24–37.

DARONDEAU, PH., AND DEGANO, P. 1991. About semantic action refinement. *Fund. Inf. XIV*, 2, 221–234.

DE BAKKER, J., DE ROEVER, W. P., AND ROZENBERG, G. EDS. 1989. Linear time, branching time and partial order in logics and models for concurrency. In *Lecture Notes in Computer Science*, vol. 354, Springer-Verlag, New York.

DE NICOLA, R. 1987. Extensional equivalences for transition systems. *Acta Inf. 24*, 211–237.

DE NICOLA, R., FANTECHI, A., GNESI, S., AND RISTORI, G. 1993. An action-based framework for verifying logical and behavioral properties of concurrent systems. *Comput. Netw. and ISDN Syst. 25*, 761–778.

DE NICOLA, R., INVERARDI, P., AND NESI, M. 1990a. Using axiomatic presentation of behavioural equivalences for manipulating CCS specifications. In *Automatic Verification Methods for Finite State Systems* (Sifakis, ed.). Lecture Notes in Computer Science, Vol. 407. Springer-Verlag, New York, pp. 54–67.

DE NICOLA, R., MONTANARI, U., AND VAANDRAGER, F. W. 1990b. Back and Forth Bisimulations. In *CONCUR '90*, J. Baeten, and J. W. Klop, eds. Lecture Notes in Computer Science, vol. 458. Springer-Verlag, New York, pp. 152–165.

DE NICOLA, R., AND VAANDRAGER, F. W. 1990a. Action versus state based logics for transition systems. In *Semantics of Concurrency*, I. Guessarian, ed. Lecture Notes in Computer Science, vol. 469. Springer-Verlag, New York, pp. 407–419.

DE NICOLA, R., AND VAANDRAGER, F. W. 1990b. Three logics for branching bisimulation (Extended Abstract). In *Proceedings of the 5th Annual Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society Press, New York, pp. 118–129.

EMERSON, E. A., AND HALPERN, J. Y. 1986. "Sometimes" and "Not Never" Revisited: On branching versus linear time temporal logic. *J. ACM 33*, 1 (Jan.) 151–178.

EMERSON, E. A., AND LEI, C. L. 1984. Model checking under generalized fairness constraints. Technical Report TR-84-20, Dept. of Computer Sciences, University of Texas at Austin, Austin, Texas, June.

EMERSON, E. A., AND SRINIVASAN, J. 1989. Branching time temporal logic. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, J. de Bakker, W. P. de Roever, and G. Rozenberg, Eds Lecture Notes in Computer Science, vol. 354. Springer-Verlag, New York, pp. 123–172.

GROOTE, J. F., AND VAANDRAGER, F. W. 1990. An efficient algorithm for branching bisimulation and stuttering equivalence. In *Proceedings of ICALP '90*, M. S. Paterson, Ed. Lecture Notes in Computer Science, vol. 443. Springer-Verlag, New York, pp. 626–638.

van Glabbeek, R. J. 1990. Comparative concurrency semantics and refinement of actions. Ph.D. dissertation, Free University, Amsterdam, The Netherlands.

van Glabbeek, R. J., and Weijland, W. P. 1989/1991. Branching time and abstraction in bisimulation semantics (extended abstract). In *Information Processing '89*, G. X. Ritter, ed. North Holland, Amsterdam, The Netherlands, pp. 613–618. Full paper appeared as CWI Report CS-R9120, Amsterdam 1991.

van Glabbeek, R. J., and Weijland, W. P. 1989. Refinement in branching time semantics. In *Proceedings of the AMAST Conference*. IOWA pp. 197–201.

Hennessy, M., and Milner, R. 1985. Algebraic laws for nondeterminism and concurrency. *J. ACM 32*, 1 (Jan.). 137–161.

Hennessy, M., and Stirling, C. 1985. The power of the future perfect in program logics. *Inf. Control 67*, 23–52.

Jonsson, B., Khan, A. H., and Parrow, J. 1990. Implementing a model checking algorithm by adapting existing automated tools. In *Automatic Verification Methods for Finite State Systems* (J. Sifakis, ed.). Lecture Notes in Computer Science, vol. 407, Springer-Verlag, New York, pp. 179–188.

Korver, H. 1992. Computing distinguishing formulae for branching bisimulation. In *Proceedings of Computer Aided Verification*, K. Larsen, and A. Skou, eds. Lecture Notes in Computer Science, vol. 575. Springer-Verlag, New York, pp. 13–23.

Koutny, M. 1991. Axiom system induced by CTL* logic, 1990. *Fund. Inf. XIV* 2, 235–252.

Kozen, D. 1983. Results on the propositional mu-calculus. *Theoret. Comput. Sci., 27*, 333–354.

Laroussine, F., Pinchinat, S., and Schnoebelen, Ph. 1995. Translation results for modal logics of reactive systems. *Theoret. Comput. Sci. 140*, 1, 53–71.

Lichtenstein, O., Pnueli, A., and Zuck, L. 1985. The glory of the past. In *Proceedings of the Conference in Logics of Programs*. Lecture Notes in Computer Science, vol. 193. Springer-Verlag, New York, pp. 196–218.

Milner, R. 1989. *Communication and Concurrency*. Prentice-Hall, Englewood Cliffs, N.J.

Manna, Z., and Pnueli, A. 1992. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, New York.

Park, D. M. R. 1989. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI Conference* (P. Deussen, ed.). Lecture Notes in Computer Science, vol. 104. Springer-Verlag, New York, pp. 167–183.

Polak, E. E. 1992. An efficient implementation of branching bisimulation and distinguishing formulae. Master Thesis. Programming Research Group. Univ. Amsterdam, Amsterdam, The Netherlands.

Pratt, V. 1981. A decidable mu-calculus. In *Proceedings of the 22nd Symposium on Foundations of Computer Science*. IEEE Computer Society Press, New York, pp. 421–427.

Stirling, C. 1989. Temporal logics for CCS. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, J. de Bakker, W. P. de Roever, and G. Rozenberg, eds. Lecture Notes in Computer Science, vol. 354. Springer-Verlag, New York, pp. 660–672.

Stirling, C. 1992. Modal and temporal logics. In *Handbook of Logic in Computer Science*, vol. I (S. Abramsky, D. M. Gabbay and T. S. E. Maibaum, eds.) O. U. P., Oxford, pp. 477–563.

Street, R. S. 1982. Propositional dynamic logic of looping and converse is elementarily decidable. *Inf. Cont. 54*, 121–141.

RECEIVED OCTOBER 1990; REVISED OCTOBER 1992; ACCEPTED FEBRUARY 1994