Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

R. De Nicola, F.W. Vaandrager

Three logics for branching bisimulation

Computer Science/Department of Software Technology Report CS-R9012 June

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Research (N.W.O.).

Three Logics for Branching Bisimulation

Rocco De Nicola
IEI - CNR
Via S. Maria, 46 I-56126 Pisa
ITALY
DENICOLA@ICNUCEVM.CNUCE.CNR.IT

Frits Vaandrager
CWI
P.O. Box 4079, 1009 AB Amsterdam
THE NETHERLANDS
FRITSV@CWI.NL

Abstract

Three temporal logics are introduced which induce on labelled transition systems the same identifications as branching bisimulation. The first is an extension of Hennessy-Milner Logic with a kind of "until" operator. The second is another extension of Hennessy-Milner Logic which exploits the power of backward modalities. The third is CTL without the next-time operator interpreted over all paths, not just over maximal ones. 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.*

Key Words and Phrases: semantics, concurrency, reactive systems, labelled transition systems, branching bisimulation equivalence, Hennessy-Milner logic, until operators, backward modalities, Kripke structures, CTL*, stuttering equivalence, doubly labelled transition systems.

1985 Mathematics Subject Classification: 03B70, 68Q05, 68Q55.

1987 CR Categories: F.1.1, F.3.0, F.4.9.

Note: The first author has been partially supported by Esprit Basic Research Action Program, Project 3011 CEDISYS and by CNR Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, project LAMBRUSCO. The research of the second author was supported by RACE project 1040, SPECS. Part of the research was carried out during a visit of the second author to Pisa which was supported by CEDISYS and LAMBRUSCO.

An extended abstract of this paper appeared in the Proceedings of the 5th Annual Symposium on Logic in Computer Science (LICS '90), Philadelphia, USA, June 1990, IEEE Computer Society Press, Los Alamitos, CA, 1990, pp. 118-129.

Report CS-R9012
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Contents

1. Introduction

The operational semantics of concurrent systems has often been described by means of labelled transition systems. However, these descriptions are frequently too concrete and do not always give the same account of systems which exhibit identical observable behaviour. The addition of a plausible notion of behavioural equivalence permits to overcome these problems; see [DeN87] and [vGl90] for comparative presentations of many such equivalences.

Together with the definition of the equivalences, different attempts have been made towards defining new logics which permit the specification of concurrent systems. In particular, temporal logic has been seen as a promising approach (see [REX89]). To date, there is no general agreement on the type of temporal logic to be used, and, since logics also naturally give rise to equivalence classes consisting of all those systems which satisfy the same formulae, often the proposed logics have been compared with operational equivalences for a better understanding and evaluation.

A well known result relating operational and logical semantics is that reported in [HM85]. In that paper, a modal logic, now known as Hennessy-Milner Logic (HML), is defined which, when interpreted over (transition) labelled transition systems with and without silent actions, is proved to be in full agreement with two operational equivalences called *strong* and *weak observational equivalence*, respectively. Other correspondences have been established in [BCG88]; two equivalences over Kripke structures (state-labelled transition systems) are related to two variants of CTL* [EH86]. 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 which have the same labelling.

Recently, a new notion of behavioural equivalence for labelled transition systems, called *branching bisimulation* ($\approx_b$), has been proposed [GW89]. It aims at generalizing strong observational equivalence to ignore silent actions while preserving the branching structures of systems. Indeed, $\approx_b$ considers two systems to be equivalent only if every computation, i.e. every sequence of (visible and silent) actions and states, of one system has a correspondent in the other; corresponding computations have the same sequence of visible actions and are such that all their intermediate states have equivalent potentials. Branching bisimulation is more restrictive than weak observational equivalence but has a pleasant axiomatic characterization which leads to a complete canonical term rewriting system [DIN90] and does indeed preserve the branching structures of systems. In [GV90] 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.

In this paper, we study the logical characterization of branching bisimulation, and propose three different logics which serve our scope.

The first logic, L_U, is obtained from HML by replacing the indexed operator <a> with a kind of "until" operator. The new binary operator, written φ<a>φ', tests whether a system can reach, by exhibiting a visible action a, a state which satisfies φ' while moving only through states which satisfy φ. It is worth noting that the original HML can be recovered from L_U in the sense that for any HML formula there exists a formula in L_U which is logically equivalent. Clearly, if no silent action is present, L_U induces the same identifications as HML.

The second logic, L_{BF}, stems from the characterization of $\approx_b$ as *back and forth bisimulation equivalence* [DMV90]. It extends HML with reverse operators (see [Sti90]). These operators permit inquiries to be made about the past of computations. It turns out that the logic L_{BF} is at least as expressive as the logic L_U: we show that the φ<a>φ' modalities of L_U are definable in terms of the back and forth modalities of L_{BF}.

The third logic which we use to characterize $\approx_b$ is a variant of CTL*, more specifically it is CTL*- X when interpreted, as in the original proposal (see [ES89]), over all runs of Kripke structures and not just over maximal runs. Together with this correspondence, we provide a variant of branching bisimulation which is in full agreements with CTL*- X interpreted over maximal runs. The steps we perform to prove the correspondence between CTL*-X and $\approx_b$ allow us to establish some interesting connections between the state- and action-based approaches to the semantics of concurrent systems. Indeed, we establish the relationships between CTL* and $\approx_b$ by relating both to variants of the stuttering equivalence over Kripke structures ($\approx_s$) of [BCG88].

We give a logical characterization of two variants of stuttering equivalence. The first equivalence is weaker than $\approx_s$ and is insensitive to *divergence* (infinite stuttering), we call it *divergence blind stuttering equivalence* ($\approx_{dbs}$). Its definition is new and simpler than that of $\approx_s$. The second equivalence, called *divergence sensitive stuttering equivalence* ($\approx_{dss}$) is defined in terms of the first one. We prove that $\approx_{dss}$ induces the same identification as CTL*-X interpreted over maximal runs and thus, since a similar result for $\approx_s$ has been proved in [BCG88], we have that $\approx_{dss}$

coincides with $\approx_s$, the original stuttering equivalence. Our characterization of $\approx_s$ in terms of $\approx_{dbs}$ is used as a key step towards the $O(m \times n)$ algorithm for deciding stuttering equivalence of [GV90]. Finally, we define a divergence sensitive version of branching bisimulation which coincides with $\approx_s$.

To relate branching bisimulation and stuttering equivalence, we present two general constructions which, given a labelled transition system, yield an enriched system in which both states and transitions are labelled. For both constructions, the generated system has almost the same structure as that of the original one: in the first construction the unfoldings of the two systems are isomorphic, the second construction just places a new state in the middle of each visible transition. We prove that divergence blind stuttering equivalence and $\approx_b$, and divergence sensitive branching bisimulation and $\approx_s$ induce the same identifications on the class of enriched systems.

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 which hold along it. The second logic introduces a backward modality which permits to test both for properties which are verified after the execution of a particular visible action and for properties which where enjoyed before the execution of the action.

We provide now 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 of [GW89]; however, it can be easily proved that our equivalence does indeed coincides with the original one.

Definition 2.1. (*Labelled Transition Systems*)
A *labelled transition system* (or *LTS*) is a triple $\mathcal{A} = (S, A, \rightarrow)$ where:
- S is a set of *states*;
- A is a set of *actions*; the *silent action* τ is not in A;
- $\rightarrow \subseteq S \times (A \cup \tau) \times S$ is the *transition relation*; an element $(r,\alpha,s) \in \rightarrow$ is called a *transition*, and is usually written as $r -\alpha\rightarrow s$.

We let $A_\tau = A \cup \{\tau\}$; $A_\varepsilon = A \cup \{\varepsilon\}$, $\varepsilon \notin A_\tau$. Moreover, we let r, s, ... range over S; a, b, ... over A; α, β, ... over A_τ and k, ... over A_ε.

We will also make use of the mapping $(.)^\circ: A_\tau \rightarrow A_\varepsilon$ which is such that $\alpha^\circ = \alpha$ if $\alpha \in A$ and $\alpha^\circ = \varepsilon$ otherwise. ♦

Definition 2.2. (*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 sequences is denoted by juxtaposition; λ denotes the empty sequence; $|\pi|$ denotes the length of a sequence π. ♦

Definition 2.3. (*Paths and runs over LTS's*)

Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS.

- A sequence $(s_0,\alpha_0,s_1) \ldots (s_{n-1},\alpha_{n-1},s_n) \in \rightarrow^*$ is called a *path* from s_0;
- a *run* from $s \in S$ is a pair (s,π), where π is a path from s;
- we write $run_{\mathcal{A}}(s)$, or just run(s), for the set of runs from s;
- we write $run_{\mathcal{A}}$ for the set of runs in $\mathcal{A}$.

We let $\pi, \ldots$ range over paths and $\rho, \sigma, \ldots$ over runs. ♦

Definition 2.4. (*Many step transitions and bounded nondeterminism*)

Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS.

i) Let $=\varepsilon=>$ be the transitive and reflexive closure of $-\tau\rightarrow$. For $a \in A$, we define on S, $r=a=>s$ if and only if there exists r' and s' in S such that $r=\varepsilon=>r'-a\rightarrow s'=\varepsilon=>s$.

ii) $\mathcal{A}$ has *bounded nondeterminism* iff for all $s \in S$ and $k \in A_\varepsilon$ the set $\{r \mid s=k\Rightarrow r\}$ is finite. ♦

Definition 2.5. (*Branching bisimulation*)

Let $\mathcal{A} = (S, A, \rightarrow)$ be a 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 R s and $r-\alpha\rightarrow r'$, then either $\alpha=\tau$ and r' R s, or $\exists$ s', s'' such that $s=\varepsilon=>s'-\alpha\rightarrow s''$, r R s' and r' R s''.
- Two states r, s of S are *branching bisimilar,* abbreviated $\mathcal{A}$: $r \approx_b s$ or $r \approx_b s$, if there exists a branching bisimulation relating r and s. ♦

The arbitrary union of branching bisimulation relations is again a branching bisimulation; $\approx_b$ is the maximal branching bisimulation and is an equivalence relation.

We could have strengthened the above definition by requiring *all* intermediate states in $s=\varepsilon\Rightarrow s'$ to be related with r. The following lemma implies that this would have lead to the same equivalence relation.

Lemma 2.6. (*Stuttering lemma, cf. Lemma 1.3 of [GW89]*)

Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS and let for some $n > 0$, $(s_0,\tau,s_1) \ldots (s_{n-1},\tau,s_n)$ be a path in $\mathcal{A}$ with $s_0 \approx_b s_n$. Then for all $0 \leq i \leq n$: $s_0 \approx_b s_i$.

Proof: See [DMV90], Lemma 2.3.2. ♦

In the rest of the paper we will study the relationships between branching bisimulation and the equivalence induced by different logics. Given a logical language L and an associated satisfaction relation $\models$, a general definition of the equivalence $\sim_L$ on the states of a labelled transition system $\mathcal{A}$ induced by L-formulas, is given by:

$\mathcal{A}$: r $\sim_L$ s if and only if $(\forall \varphi \in L : \mathcal{A}, r \models \varphi \Leftrightarrow \mathcal{A}, s \models \varphi)$.

We will show that, for three significantly different logics, $\sim_L$ coincides with branching bisimulation equivalence.

2.1. Until operators

The first logic we will introduce is a variant of Hennessy-Milner Logic which rather than the family of diamond operator <a> has an indexed until operator. Below, we will introduce our new logic after presenting syntax and semantics of the original HML.

Definition 2.7. (*Hennessy-Milner Logic*)
Let A be a given alphabet of symbols. The syntax of HML is defined by the following grammar where we let φ, φ', ... range over HML formulas (and k over A_ε):

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

Definition 2.8. (*The satisfaction relation for HML*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS. *Satisfaction* of a HML-formula φ by a state $s \in S$, notation $\mathcal{A}, s \models \varphi$, 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 $s =k\Rightarrow s'$ and $s' \models \varphi$. ♦

For labelled transition systems with bounded nondeterminism, the above logic has been proved, in [HM85], to be in full agreement with the equivalence relation known as weak observational equivalence, which is based on a slightly less demanding bisimulation than that of Definition 2.5. In order to consider equivalent two states weak bisimulation only requires them to lead, via the same sequences of visible actions, to equivalent states; the intermediate states along the path are not investigated. In order to take also the properties of these intermediate states into account, within the new version of HML, 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 k, a state which satisfies φ' while moving only through intermediate states which satisfy φ.

Definition 2.9. (*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 we let φ, φ'... range over L_U formulas:

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

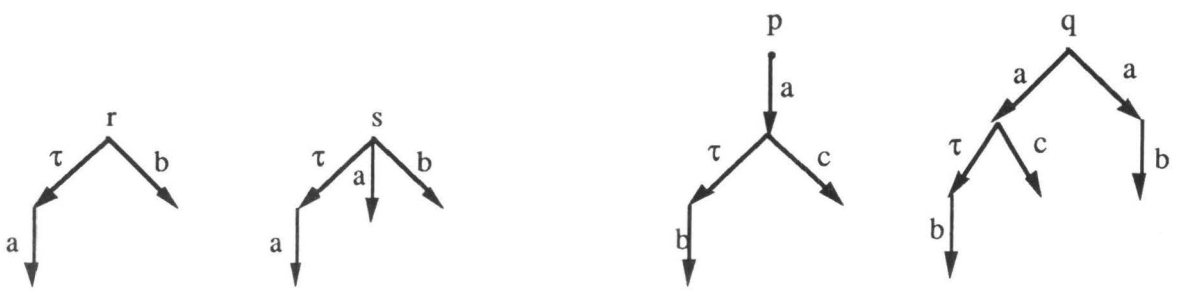
Definition 2.10. (*The satisfaction relation for L_U*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS. *Satisfaction* of an L_U-formula φ by a state $s\in S$, notation $\mathcal{A},s \models \varphi$, or just $s \models \varphi$, is defined inductively by:

- $s \models T$ 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 $k=\varepsilon$ and $s \models \varphi'$, or there is a run $(s, (s_0,\tau,s_1)\ldots(s_{n-1},\tau,s_n)\ (s_n,\alpha,s_{n+1}))$ such that $\forall i \leq n$: $s_i \models \varphi$, $k=\alpha^\circ$ and $s_{n+1} \models \varphi'$ with $n \geq 0$. ♦

It is possible to define, within L_U, other temporal operators; we will write $\langle k\rangle\varphi$ for $T\langle k\rangle\varphi$ and $[k]\varphi$ for $\neg\langle k\rangle\neg\varphi$. It is worth noting that 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 "$\langle k\rangle\langle\varepsilon\rangle\varphi$". In the latter formula, we need to have $\langle\varepsilon\rangle$ after $\langle k\rangle$ because our 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 runs are considered which may continue with sequences of invisible actions. Clearly, if no silent action is present, L_U and HML induce the same identifications on LTS's.

We give now two pairs of systems and two formulae which show the additional power of L_U when compared with the original Hennessy-Milner Logic. The two pairs $\langle r, s\rangle$ and $\langle p, q\rangle$ are just two instances of the second and third τ'laws (see e.g. [Mil89]) respectively; thus they are certainly not differentiated by HML.

Example 2.11. (*Two pairs of states which are weak observational equivalent but not branching bisimilar*).
Consider the following LTS $\mathcal{A}_{2.11}$:



If we let $\varphi = (<b>T)\ <a>\ T$ we have $s \models \varphi$ while $r \not\models \varphi$.
If we let $\varphi' = [a]\ (<c>T)$ we have $p \models \varphi'$ while $q \not\models \varphi'$. ♦

Theorem 2.12. (*L_U and branching bisimulation induce the same identifications on bounded LTS's*).
Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS with bounded nondeterminism. Then for all r, s in S:

$\mathcal{A}$: $r \approx_b s$ if and only if $\mathcal{A}$: $r \sim_{L_U} s$.

Proof: "$\Rightarrow$" Suppose $r \approx_b s$ and let $\varphi \in L$ such that $r \models \varphi$. With rather straightforward induction on the structure of φ one can prove that $s \models \varphi$.
"$\Leftarrow$" 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 $p \text{–}\alpha\rightarrow p'$. 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 collection $\mathbf{Q}$ of all paths from q of the form $(q_0,\tau,q_1)..(q_{n-1},\tau,q_n)(q_n,\alpha,q')$ 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 will prove that $\mathbf{Q}$ is finite by reductio ad absurdum. Suppose it is infinite. Consider the tree, $T_{\mathbf{Q}}$, which has as nodes the paths in $\mathbf{Q}$, and which has an edge from node π to node π' if and only if π' can be obtained from π by extending it with a single transition. Since $\mathcal{A}$ has bounded nondeterminism, $T_{\mathbf{Q}}$ will certainly be finitely branching. Therefore, by König's lemma, the tree has an infinite path. But this implies that there is an infinite, acyclic path in $\mathcal{A}$, starting in q, with τ-transitions only. This contradicts the assumption that $\mathcal{A}$ has bounded nondeterminism.

In order to prove the transfer property, it is sufficient to show that there is a path in $\mathbf{Q}$ with the property that all states on the path, 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 path. We will derive a contradiction. We can split $\mathbf{Q}$ into two subsets $\mathbf{Q}_S$ and $\mathbf{Q}_f$ such that for any path π_s in $\mathbf{Q}_S$ there is a formula φ_{π_s} which holds in p but not in all non-final states of π_s, and for any path π_f in $\mathbf{Q}_f$ there is a formula φ_{π_f} which holds in p' but not in the last state of π_f. Let φ_S be the conjunction of the formulas φ_{π_s} with π_s in $\mathbf{Q}_S$ and let φ_f be the conjunction of the formulas φ_{π_f} with π_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 $\varphi = \varphi_S<\varepsilon>(\varphi_f \wedge \varphi_0)$. We have that $p \models \varphi$ while $q \not\models \varphi$ and thus a contradiction.
2. $\alpha\neq\tau$. Now we take $\varphi = \varphi_S<\alpha>\varphi_f$ and we have a contradiction because $p \models \varphi$ but $q \not\models \varphi$. ♦

2.2. Backward Modalities

In this section, we present a new kind of bisimulation which we call back and forth bisimulation. It not only requires the futures of equivalent processes to be equivalent but constraints also their pasts. This new bisimulation has been put forward in [DMV90], where it is proved that it induces

on LTS's 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 which permits analyzing the past of computations. The spirit of the last generalization of HML is similar to that proposed by Hennessy and Stirling in [HS85], the relevant difference is that we take into account also the possibility that some of the actions might be invisible while they deal with visible actions only and thus do not admit partially controlled state changes. Indeed, the past operator is introduced in [HS85] only to capture non-continuous 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.

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 which represents its *history*. We can easily generalize to runs the transition relation over states:

- $\rho -\alpha\rightarrow \sigma$ if there exists a run $\theta = (s, (s, \alpha, s'))$ such that ρ concatenated with θ gives σ;
- $\rho =\varepsilon=> \sigma$ if there exist $\rho_0, \rho_1, \ldots \rho_n$, $(n \geq 0)$, such that $\rho = \rho_0$, $\rho_n = \sigma$ and for all $0 \leq i < n$:
  $\rho_i -\tau\rightarrow \rho_{i+1}$;
- $\rho =\alpha=> \sigma$ if there exist ρ', σ' such that $\rho =\varepsilon=>\rho' -\alpha\rightarrow \sigma' =\varepsilon=> \sigma$.

Below 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 [DMV90]. Here, we would only like to stress, once again, that we do not define bisimulations as relations between states anymore but as relations between runs. The equivalence of two given states is obtained by considering all runs from them.

Definition 2.13. (*Back and forth bisimulation*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS. Two states $r, s \in S$ are *back and forth bisimilar*, abbreviated $\mathcal{A}$: $r \approx_{bf} s$ or $r \approx_{bf} s$, if there exists a symmetric relation $R \subseteq run_{\mathcal{A}} \times run_{\mathcal{A}}$, called a *back and forth bisimulation*, satisfying:

i) $(r, \lambda)\ R\ (s, \lambda)$;

ii) if $\rho\ R\ \sigma$ and $\rho =k=>\rho'$ then there exists a σ' such that $\sigma =k=>\sigma'$ and $\rho'\ R\ \sigma'$;

iii) if $\rho\ R\ \sigma$ and $\rho' =k=>\rho$ then there exists a σ' such that $\sigma' =k=>\sigma$ and $\rho'\ R\ \sigma'$. ♦

Theorem 2.14. (*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: See [DMV90], Theorem 2.3.3. ♦

Definition 2.15. (*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 formulae of the language:

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

Definition 2.16. (*The Satisfaction Relation for L_{BF}*)
i) Let $\mathcal{A} = (S, A, \rightarrow)$ be a 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 $\rho =k=> \rho'$ and $\rho' \models \varphi$;
- $\rho \models \langle \leftarrow k\rangle\varphi$ iff there exists a run ρ' such that $\rho' =k=> \rho$ and $\rho' \models \varphi$.

ii) For $s \in S$ and $\varphi \in L_{BF}$ we define $s \models \varphi$ iff $(s, \lambda) \models \varphi$. ♦

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 [HS85] 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. Going back to systems with silent actions, below, we show that also L_{BF} is able to differentiate the systems of Example 2.11.

Example 2.17.

Let p, q, r and s be as in Example 2.11, 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 2.18. (*L_{BF} and branching bisimulation induce the same identifications on bounded LTS's*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS with bounded nondeterminism. Then for all r, s in S:

$\mathcal{A}$: $r \approx_b s$ if and only if $\mathcal{A}$: $r \sim_{L_{BF}} s$.

Proof: Given a LTS $\mathcal{A}$, we can build a new one, $BF(\mathcal{A})$, which is obtained by replacing the single step transition relation of $\mathcal{A}$ with the corresponding many step forward and backward

arrows between paths of $\mathcal{A}$. More precisely, we define $BF(\mathcal{A}) = (\text{run}_{\mathcal{A}}, A_{bf}, \rightarrow_{bf})$ where $A_{bf} = A_\varepsilon \cup \{{}^{\leftarrow}k \mid k \in A_\varepsilon\}$ and for $\rho, \rho' \in \text{run}_{\mathcal{A}}$ and $k \in A_\varepsilon$, $\rho - k \rightarrow_{bf} \rho'$ iff $\rho = k => \rho'$ and $\rho - {}^{\leftarrow}k \rightarrow_{bf} \rho'$ iff $\rho' = k => \rho$. We can now prove that $\mathcal{A}$: $r \approx_{bf} s$ if and only if $BF(\mathcal{A})$: $r \approx s$, where $\approx$ stands for Milner's strong observational equivalence. The claim then follows directly from Theorem 2.14 and from the HML characterization of $\approx$ in [HM85]. ♦

We can now prove that the logic L_{BF} is at least as expressive as the logic L_U, in the sense that we can give for each L_U-formula an L_{BF}-formula which is logically equivalent to it.

Theorem 2.19. (*L_{BF} is at least as expressive as L_U*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS and $s \in S$. There exists a mapping $f: L_U \rightarrow L_{BF}$ such that for all φ in L_U: $s \models \varphi$ iff $s \models f(\varphi)$.

Proof: f is defined inductively by
- $f(T) = T$;
- $f(\neg\varphi) = \neg f(\varphi)$;
- $f(\varphi \wedge \varphi') = f(\varphi) \wedge f(\varphi')$
- $f(\varphi \langle\varepsilon\rangle \varphi') = \langle\varepsilon\rangle(f(\varphi') \wedge [{}^{\leftarrow}\varepsilon]\,(f(\varphi') \vee f(\varphi)))$
- $f(\varphi \langle a \rangle \varphi') = \langle a \rangle([{}^{\leftarrow}\varepsilon]\, f(\varphi') \wedge [{}^{\leftarrow}a]\, f(\varphi))$.

It is straightforward to check, by induction on the structure of φ, that f has the required property. ♦

At the moment we do not know whether the reverse of the above theorem holds, i.e., whether we can find a mapping $g: L_{BF} \rightarrow L_U$ which, like f, preserves the satisfaction relation.

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 [BCG88]. First of all, we introduce the relevant notation for the class of structures which have been used to interpret CTL* and to define stuttering equivalence.

Definition 3.1. (*Kripke structures*)
Let **AP** be a fixed nonempty set of *atomic proposition names* ranged over by p, q, ... A *Kripke structure* (or *KS*) is a triple $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ where:
- S is a set of *states*;
- $\mathcal{L}: S \rightarrow 2^{\mathbf{AP}}$ is the *proposition labelling*;

- $\rightarrow \subseteq S \times S$ is the *transition relation*; an element $(r,s) \in \rightarrow$ is called a *transition* and is usually written as $r \rightarrow s$.

We let r, s, ... range over states of Kripke structures. ♦

Definition 3.2. (*Notation for Kripke structures*)

Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a *Kripke structure*.

- A (finite or infinite) sequence $(s_0, s_1)(s_1, s_2) \ldots \in \rightarrow^{\infty}$ is called a *path* from s_0; if the sequence of pairs of states is maximal the path is called *fullpath.*
- A *run* from $s \in S$ is a pair (s,π), where π is a path from s.
- We write $\text{run}_{\mathcal{K}}(s)$, or just run(s), for the set of runs from s, and $\mu\text{run}_{\mathcal{K}}(s)$, or just μrun(s), for the set of *maximal runs* (i.e., runs whose second element is a fullpath) from s.
- We let $\rho, \sigma, \theta, \eta, \ldots$ range over runs.
- If $\rho = (s, \pi)$ is a run and $\pi = (s_0,s_1)(s_1,s_2)...$, then $\text{first}(\rho) = s$, $\text{path}(\rho) = \pi$ and $\text{states}(\rho) = s_0s_1s_2\ldots$. Moreover, if π is finite then last(ρ) denotes the last state of ρ.
- With $\rho < \theta$ and $\rho \leq \theta$ we indicate that run θ is a proper suffix, respectively a suffix, of run ρ.
- Concatenation of runs is denoted by juxtaposition. ♦

Definition 3.3. (*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 φ,... range over state formulas and π,... 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 $\varphi U \varphi'$ are path formulas;
- if π is a path formula, then so is $\neg\pi$. ♦

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

Now, we present two different satisfaction relations for the logics introduced above. This will be done by relying on different structures to interpret formulae. In one case, we will use only maximal runs of Kripke structures to interpret path formulae, in the other, we will use both finite and

infinite runs. Due to its ability of describing non continuous properties like fairness, the generally accepted interpretation of CTL*, is that based on maximal runs only. The less restrictive interpretation, however, has a series of interesting properties and is the version of CTL* which was originally proposed (see [ES89]).

Definition 3.4. (*Two satisfaction relations for CTL**)

Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a Kripke structure.

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

- $s \models p$ iff $p \in \mathcal{L}(s)$
- $s \models \neg\varphi$ 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 run $\rho \in run(s)$ such that $\rho \models \pi$
- $\rho \models \varphi$ iff $first(\rho) \models \varphi$
- $\rho \models \neg\pi$ iff $\rho \not\models \pi$
- $\rho \models \pi \wedge \pi'$ iff $\rho \models \pi$ and $\rho \models \pi'$
- $\rho \models \pi U \pi'$ 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 exist η, θ such that the path of η has length 1, $\rho = \eta\theta$ and $\theta \models \pi$.

ii) *Satisfaction wrt maximal paths* of a state formula φ by a state s, notation $(\mathcal{K},)s \models_\mu \varphi$, and of a path formula π by a maximal run ρ, notation $(\mathcal{K},)\rho \models_\mu \pi$, 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 run $\rho \in \mu run(s)$ such that $\rho \models_\mu \pi$. ♦

3.1. 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 [BCG88]. Browne, Clarke and Grümberg assume to deal always with structures whose states are never deadlocked; if systems have to be modelled which contain states without any outgoing transition they assume the presence of a transition from the final state to itself, thus all maximal runs 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 which 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 3.5. *(Divergence blind stuttering equivalence)*

Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a Kripke structure.

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

- $\mathcal{L}(r)=\mathcal{L}(s)$ and
- if $r \rightarrow r'$, then there exist $s_0, s_1,.., s_n$ such that $s_0 = s$ and for all $i < n$: $s_i \rightarrow s_{i+1}$, r R s_i and r' R s_n.

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

iii) Two runs ρ,σ are *divergence blind stuttering equivalent,* notation $\mathcal{K}$: $\rho \approx_{dbs} \sigma$ or $\rho \approx_{dbs} \sigma$, if there is a partition $B_1 B_2...$ of states(ρ) and a partition $B'_1 B'_2...$ of states(σ) such that for all j, B_j and B'_j are both non-empty and every state in B_j is divergence blind stuttering equivalent to every state in B'_j. ♦

Lemma 3.6.

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

Proof: The actual proof is trivial, only notationally heavy; it is left to the reader. ♦

Theorem 3.7. Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a Kripke structure and let r, s ∈ S with $r \approx_{dbs} s$.

Then for every CTL*–X formula φ: $r \models \varphi$ iff $s \models \varphi$.

Proof: Suppose $r \approx_{dbs} s$. Let $\rho \in$ run(r) and $\sigma \in$ run(s) with $\rho \approx_{dbs} \sigma$ and let χ be either a state formula or a path formula which does not contain any X-operator. We will prove the following statements:

i) if χ is a state formula then $r \models \chi$ implies $s \models \chi$

ii) if χ is a path formula then $\rho \models \chi$ implies $\sigma \models \chi$.

by induction on the structure of χ. The reverse implications follow by symmetry

First, we consider the case of state formulae.

1. $\chi = p$: $r \models p$ iff $p \in \mathcal{L}(r)$, the latter implies $p = \mathcal{L}(s)$, by definition of $r \approx_{dbs} s$, hence $s \models \chi$.
2. $\chi = \neg\varphi$: $r \models \neg\varphi$ implies $r \not\models \varphi$, this by induction implies $s \not\models \varphi$, thus $s \models \neg\varphi$.
3. $\chi = \varphi\wedge\varphi'$: $r \models \varphi\wedge\varphi'$ implies $s \models \varphi\wedge\varphi'$ follows by induction similarly to case 2.
4. $\chi = \exists\pi$: Suppose $r \models \exists\pi$. Then there exists a run $\rho \in$ run(r) such that $\rho \models \pi$. By Lemma 3.6, we can find a run $\sigma \in$ run(s) such that $\rho \approx_{dbs} \sigma$; moreover, by induction we have that $\sigma \models \pi$. Thus $s \models \exists\pi$.

Next, we consider the four cases of path formulae.

5. $\chi = \varphi$: We have $\rho \models \varphi$ implies $first(\rho) = r \models \varphi$ which by induction implies $first(\sigma) = s \models \varphi$. From the latter we can conclude $\sigma \models \varphi$.
6. $\chi = \neg\pi$: Easy induction.
7. $\chi = \pi \wedge \pi'$: Easy induction.
8. $\chi = \pi U \pi'$: Suppose $\rho \models \pi U \pi'$. There exists then a run θ with $\rho \leq \theta$ such that $\theta \models \pi'$ and for all $\rho \leq \nu < \theta$, $\nu \models \pi$. Since $\rho \approx_{dbs} \sigma$, there is a partition $B_1 B_2 ...$ of states(ρ) and a partition $B'_1 B'_2 ...$ of states(σ) such that for all j, B_j and B'_j are both non-empty and every state in B_j is stutteringly bisimilar to every state in B'_j. Let ρ_i be a suffix of ρ that has the first element of block B_i as first state. Similarly, let σ_i be a suffix of σ that has the first element of block B'_i as first state. Now let B_k be the block in which the first state of θ occurs. One can easily check that $\theta \approx_{dbs} \sigma'_k$, then by induction we have $\sigma'_k \models \pi'$. Let η be a run such that $\sigma \leq \eta < \sigma'_k$, and let B_l' be the block in which the first state of η occurs; we have $\rho_l \approx_{dbs} \eta$. Since $l < k$ we have also $\rho \leq \rho_l < \theta$ and thus $\rho_l \models \pi$; by induction we obtain also $\eta \models \pi$. ♦

Theorem 3.8. Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ 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 an algorithm for deciding stuttering equivalence that is presented in [GV90].

• For B, B' $\subseteq$ S the set 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:

$$pos(B, B') = \{s' \in B \mid \exists\, n \geq 0,\ \exists\, s_0, s_1, \ldots, s_n = s \text{ such that } \forall i < n: s_i \in B \text{ and } s_i \rightarrow s_{i+1} \text{ and } s_n \in B'\}$$

• Call B' a *splitter* of B if and only if $\emptyset \neq pos(B, B') \neq B$.

• 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 pos(B, B') and B - 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 | $\mathcal{L}$(r) = $\mathcal{L}$(s)} | 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 [GV90] 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 which associates a CTL–X formula to each block which only holds for the states in that block. This is trivial for the initial partition. Since states in the same block have the same labelling while states in different blocks have different labelling, one can

easily give a propositional formula for each block which only holds for its states. Suppose that, at one moment, during the execution of the algorithm, B' is a splitter for B and block B is split into B_1 = pos(B, B') and B_2 = B - 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 with B_1 the formula $\varphi \wedge (\exists\, \varphi U \varphi')$ and with B_2 the formula $\varphi \wedge (\neg\exists\, \varphi U \varphi')$. For the other blocks the associated formulas remain unchanged. Now, if we associate to every state the formula that is associated to the block of the final partition in which the state occurs, then this formula will have the required property. ♦

Theorem 3.9. (*Divergence blind stuttering, CTL*-X and CTL-X agree for* $\models$)
Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ 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: (i) $\Rightarrow$ (ii) follows from Theorem 3.7; (ii) $\Rightarrow$ (iii) is immediate; and (iii) $\Rightarrow$ (i) follows from Theorem 3.8. ♦

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 [BCG88] and which does not ignore divergence. The new version is defined in terms of the divergence blind one.

Definition 3.10. (*Extending Kripke structures with livelocked state*)
Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ 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 $p_0 \notin \mathcal{L}(s)$. Define the Kripke structure $\mathcal{K}_1$ by $\mathcal{K}_1 = (S', \mathcal{L}', \rightarrow')$ where

- $S' = S \cup \{s_0\}$,
- $\mathcal{L}' = \mathcal{L} \cup \{\langle s_0, \{p_0\}\rangle\}$ and
- $\rightarrow' = \rightarrow \cup \{\langle s, s_0\rangle \mid s$ is on a cycle of states with the same label or it has no outgoing edges$\}$. ♦

Definition 3.11. (*Divergence sensitive stuttering equivalence*)
Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a finite state Kripke structure.

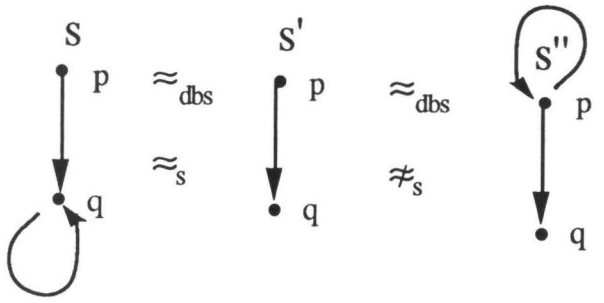
i) Two states r, s $\in$ S are *stuttering equivalent*, abbreviated ($\mathcal{K}$:) $r \approx_s s$, iff $\mathcal{K}_1$: $r \approx_{dbs} s$.

ii) Two runs ρ, σ of $\mathcal{K}$ are *stuttering equivalent*, abbreviated ($\mathcal{K}$:) $\rho \approx_s \sigma$, iff $\mathcal{K}_1$: $\rho \approx_{dbs} \sigma$. ♦

The next example evidences the different stress the two equivalences put on divergence (infinite stuttering). Please, notice that also divergence sensitive stuttering does not distinguishes between

deadlock and divergence, the equivalence is sensitive to any divergence except for that occurring in otherwise deadlocked states.

Example 3.12. (*Differences between $\approx_s$ and $\approx_{dbs}$*)



Let $\varphi = \forall$ F q. Then $s \models_\mu \varphi$ and $s' \models_\mu \varphi$, whereas $s'' \not\models_\mu \varphi$, $s \not\models \varphi$, $s' \not\models \varphi$ and $s'' \not\models \varphi$. ♦

Lemma 3.13.

Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a finite state Kripke structure, let r, s ∈ S with $r \approx_s s$ and let $\rho \in \mu\text{run}(r)$. Then there exists a $\sigma \in \mu\text{run}(s)$ such that $\rho \approx_s \sigma$.

Proof: Given any run ρ from r in $\mathcal{K}$ then it is also a run in $\mathcal{K}_1$. Since $\mathcal{K}_1$: $r \approx_{dbs} s$, we can use Lemma 3.6 to find a run σ from s in $\mathcal{K}_1$ which is equivalent to ρ. It must be that σ is also a run of $\mathcal{K}$ because if s_0 was in σ then the latter could never be related to ρ. If σ is maximal in $\mathcal{K}$ 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' = last(ρ) is a deadlocked state. Thus in $\mathcal{K}_1$ there is the transition (r', s_0). Let s' = last(σ), since $\rho \approx_{dbs} \sigma$ we have $\mathcal{K}_1$: $r' \approx_{dbs} s'$. We can now rely on the fact that $\approx_{dbs}$ is a divergence blind stuttering bisimulation to find u_0, u_1, ...u_n such that s' = u_0 and for all i<n, $u_i \rightarrow u_{i+1}$, with $r' \approx_{dbs} u_i$ and $s_0 \approx_{dbs} u_n$ but this means that $u_n = s_0$ and that, in $\mathcal{K}$, u_{i-1} is either deadlocked or occurs in a cycle of states with the same label.

In the first case, consider the run $\sigma' = \sigma\,(s', (u_0, u_1), \ldots(u_{n-2}, u_{n-1}))$. One can easily check that σ' is maximal and that $\rho \approx_s \sigma'$.

In the second case, let the path $\pi = (v_0, v_1), \ldots(v_{m-1}, v_m)$ be a cycle of states with the same label starting in u_{n-1} (so $v_0 = v_m = u_{n-1}$). One can easily show that all states in a cycles of states with the same label are divergence blind stuttering equivalent. Now consider the run $\sigma''' = \sigma'(\sigma'')^\omega$ with $\sigma'' = (u_{n-1}, \pi)$; run σ''' is maximal and $\rho \approx_s \sigma'''$.

The case of ρ infinite is dealt similarly and is left to the reader; it relies on the fact that K has only a finite number of states. ♦

Theorem 3.14. Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a finite state Kripke structure and let r, s ∈ 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 3.7 for divergence blind stuttering equivalence and replace Lemma 3.6 by Lemma 3.13. ♦

Theorem 3.15. Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a finite state Kripke structure and let $s \in S$.
Then there exists a CTL–X formula φ such that for all $r \in S$: $r \models_{\mu} \varphi$ iff $r \approx_S s$.

Proof: Similar to the proof of Theorem 3.8. Now, we apply the partition refinement algorithm on the structure $\mathcal{K}_1$. We associate a formula to each block different from $\{s_0\}$, which when interpreted over $\mathcal{K}$, only holds for the states in that block. The interesting case is the one where a block B with associated formula φ, is split in a block B_1 of states, from which after some stuttering there is a transition to s_0, and in a block $B_2 = B - B_1$. Now associate the formula $\varphi \wedge (\exists\, G\, \varphi)$ to B_1 and the formula $\varphi \wedge (\forall\, F \neg \varphi)$ to B_2. ♦

By combining Theorems 3.14 and 3.15 we obtain the following results.

Theorem 3.16. (*Stuttering, CTL*-X and CTL-X agree for* $\models_{\mu}$)
Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ 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 $s \models_{\mu} \varphi$. ♦

Since a similar result was proved in [BCG88], we have, as a corollary of the above theorem, that our version of stuttering equivalence coincides with that of [BCG88] for finite state Kripke structures without deadlocked states, i.e. for the class of KS's they consider.

3.2. 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 will get the new logical characterization of branching bisimulation by relating it to the divergence blind stuttering equivalence studied above. We will need some preliminary work which allows us to relate the different structures on which branching and stuttering equivalence are defined, namely Kripke structures and Labelled Transition Systems.

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

Definition 3.17. (*Doubly Labelled Transition Systems*)
A L^2TS is a structure $\mathcal{AK} = (S, A, \rightarrow, \mathcal{L})$ where $(S, A, \rightarrow)$ is a LTS and $\mathcal{L}: S \rightarrow 2^{\mathbf{AP}}$ is a labelling function which associates a set of atomic propositions to each state.
With $LTS(\mathcal{AK})$ we denote the substructure $(S, A, \rightarrow)$ of $\mathcal{AK}$ and with $KS(\mathcal{AK})$ we denote the substructure $(S, \mathcal{L}, \rightarrow')$ of $\mathcal{AK}$ where $r \rightarrow' s$ if and only if $\exists\, \alpha : r -\alpha\rightarrow s$.
Equivalences defined on the states of a LTS or KS can be lifted to L^2TS in the obvious way by ignoring either the labels of the states or the labels of the transitions:

- $\mathcal{AK}: r \sim s \Leftrightarrow_{def} LTS(\mathcal{AK}): r \sim s$ and
- $\mathcal{AK}: r \sim s \Leftrightarrow_{def} KS(\mathcal{AK}): r \sim s$. ♦

The actual definition of L^2TS is far 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 the states. Because of this, we introduce the class of consistent L^2TS's.

Definition 3.18. (*Consistent L^2TS's*)
A L^2TS $(S, A, \rightarrow, \mathcal{L})$ is *consistent* if there exist two functions
• *effect*: $2^{\mathbf{AP}} \times A_\tau \rightarrow 2^{\mathbf{AP}}$ and
• *action*: $2^{\mathbf{AP}} \times 2^{\mathbf{AP}} \rightarrow A_\tau$
such that
i) $effect(l, \tau) = l$;
ii) $action(l, l) = \tau$;
iii) $r -\alpha\rightarrow s$ implies ($\mathcal{L}(s) = effect(\mathcal{L}(r), \alpha)$ and $\alpha = action(\mathcal{L}(r), \mathcal{L}(s))$). ♦

What this definition amounts to saying is that the states which are connected by an invisible action have the same labels and the labels of adjacent states differ only for the repercussion of the label of the transition connecting them. The above restriction on L^2TS's, permits performing the first step toward relating branching bisimulation and CTL^*-X, because stuttering equivalence and branching bisimulation agree when they are defined on consistent L^2TS's.

Theorem 3.19. (*Divergence blind stuttering and branching bisimulation equivalence agree on consistent L^2TS's*)
If $\mathcal{AK} = (S, A, \rightarrow, \mathcal{L})$ is a consistent L^2TS then for all r, s in S:

$\mathcal{AK}: r \approx_{dbs} s$ if and only if $\mathcal{L}(r) = \mathcal{L}(s)$ and $\mathcal{AK}: r \approx_b s$.

Proof: Immediate from the definitions. ♦

We can now start studying the relationships between stuttering equivalence as defined on Kripke structures and branching bisimulation as defined on labelled transition systems. First of all, we will present a straightforward way of labelling 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. It is worth remarking that one of the main sources of problems in these transformations is the presence of invisible actions.

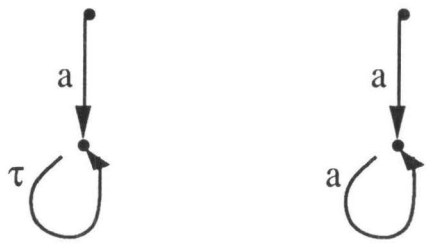
Definition 3.20. *(From KS's to L^2TS's)*
Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a Kripke structure. The L^2TS $\mathbf{tr}(\mathcal{K})$ is defined as $(S, 2^{AP}, \rightarrow', \mathcal{L})$ where
• $r -\tau\rightarrow' s$ if and only if $r \rightarrow s$ and $\mathcal{L}(r) = \mathcal{L}(s)$;
• $r -l\rightarrow' s$ if and only if $r \rightarrow s$ and $\mathcal{L}(r) \neq \mathcal{L}(s)$ and $\mathcal{L}(s) = l$. ♦

The construction of Definition 3.20 is reported also in [Kou90]. One can easily verify that $\mathbf{tr}(\mathcal{K})$ is consistent and moreover that $KS(\mathbf{tr}(\mathcal{K})) = \mathcal{K}$. The theorem below is an immediate consequence of these properties.

Theorem 3.21. Let $\mathcal{K} = (S, \mathcal{L}, \rightarrow)$ be a Kripke structure. Then for all r, s in S:
$\mathcal{K}$: $r \approx_{dbs} s$ if and only if $\mathcal{L}(r) = \mathcal{L}(s)$ and $\mathbf{tr}(\mathcal{K})$): $r \approx_b s$.

Proof: By the above property, we have that $\mathcal{K}$: $r \approx_{dbs} s$ if and only if $KS(\mathbf{tr}(\mathcal{K}))$: $r \approx_{dbs} s$. By definition $KS(\mathbf{tr}(\mathcal{K}))$: $r \approx_{dbs} s$ if and only if $\mathbf{tr}(\mathcal{K}))$: $r \approx_{dbs} s$. Now, since $\mathbf{tr}(\mathcal{K})$ is consistent, the latter holds if and only if $\mathcal{L}(r) = \mathcal{L}(s)$ and $\mathbf{tr}(\mathcal{K}))$: $r \approx_b s$. ♦

The construction of a L^2TS from a LTS is less straightforward than the above construction starting from a KS. The first thing which 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. However, this does not deal with situations where transitions with different labels enter a single state. Moreover, problems arise with structures like those reported below which capture a very different intuition but would be identified by the outlined naive transformation with the result of confusing repeated occurrences of the same visible action with τ-cycles.



One possible solution is proposed in [CLM89]. There, a given LTS is extended by labelling each state with the set of the labels of the runs which lead to it; runs are labelled by the set of those actions which are performed an odd number of times. Unfortunately, this construction does not always lead to consistent L^2TS's 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 LTS's which lead to unique labelling and this restricted class of LTS's gives rise to consistent L^2TS's only.

We now propose two new transformation functions which permit building a consistent L^2TS from any LTS, while preserving the structure of the LTS. The LTS of Example 3.23 below shows that in general it is not possible to give a labelling of the states in an LTS such that the resulting L^2TS is consistent. Thus there exists no transformation function which preserves the structure of the LTS up to isomorphism.

The first transformation function exploits the naive tranformation function sketched above: a state in the L^2TS is labelled 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. To deal with states reacheable by more than one labelled transition, the L^2TS contains a new state, s_a, for each state s of the LTS and each label a of the transitions after which s can be reached via a path consisting only of τ-transitions. To deal with the problem of two consecutive a-transitions, an additional state, $s_{\underline{a}}$, is introduced for each state that can be reached via two a-transitions (possibly with τ-steps in between). Below, we present the just schetched construction in full detail:

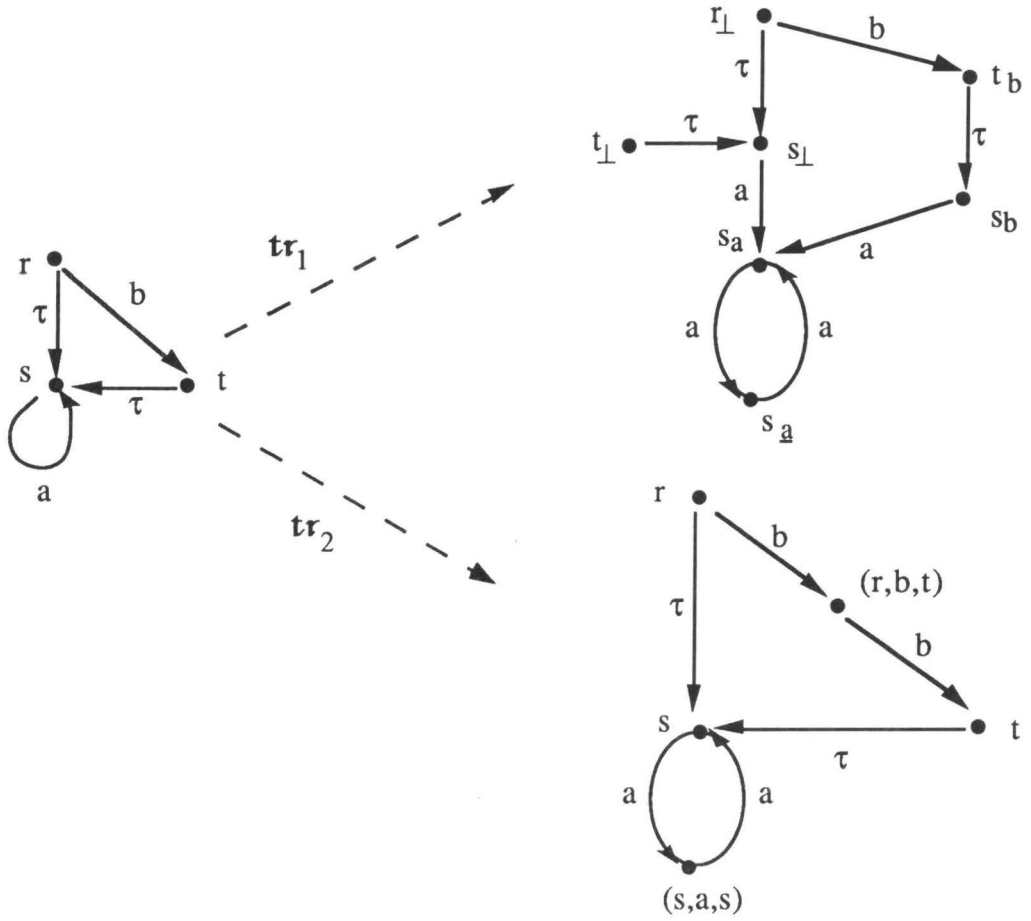
Definition 3.22. (*From LTS's to consistent L^2TS's: first transformation*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS.
Let $A^+ = A \cup \{\underline{a} \mid a \in A\} \cup \{\bot\}$ and define $\vdash \subseteq S \times A^+$ inductively by:

1. $r \vdash \bot$;
2. $r \vdash l, l \neq a$ and $r -a\rightarrow s$ implies $s \vdash a$;
3. $r \vdash a$ and $r -a\rightarrow s$ implies $s \vdash \underline{a}$;
4. $r \vdash l$ and $r -\tau\rightarrow s$ implies $s \vdash l$.

Now the L^2TS $\mathbf{tr}_1(\mathcal{A})$ can be defined as $(S', A, \rightarrow', \mathcal{L})$ where

- $S' = \{s_l \mid s \vdash l\}$;
- $\rightarrow' = \{(r_l, a, s_a) \mid r \vdash l,\ l \neq a \in A \text{ and } r -a\rightarrow s\} \cup$
  $\{(r_a, a, s_{\underline{a}}) \mid r \vdash a \text{ and } r -a\rightarrow s\} \cup$
  $\{(r_l, \tau, s_l) \mid r \vdash l \text{ and } r -\tau\rightarrow s\}$.
- $\mathcal{L}(s_l) = \{l\}$.

Example 3.23. (*Translating LTS's into L^2TS's*)



One can easily check that for a given LTS $\mathcal{A}$, $\mathbf{tr}_1(\mathcal{A})$ is a consistent L^2TS. The next proposition vindicates our construction in that it shows that the structure $\mathbf{tr}_1(\mathcal{A})$ and that of $\mathcal{A}$ are very similar. Note that, as a direct corollary of the next proposition, two states r, s are branching bisimilar if and only if states r_1 and s_1 are such for some given l.

Proposition 3.24. (*The structure of $\mathcal{A}$ and $\mathbf{tr}_1(\mathcal{A})$ is very similar*)
If s is a state of a LTS $\mathcal{A}$ and $s_\perp$ a corresponding state of LTS($\mathbf{tr}_1(\mathcal{A})$) then s and $s_\perp$ give rise to isomorphic unfoldings. ♦

Now, Proposition 3.24 and Theorem 3.19, together with Theorem 3.9, allow us to prove the following important theorem which says that, via transformation $\mathbf{tr}_1$, CTL*-X can be viewed as a logic for branching bisimulation equivalence.

Theorem 3.25.
Let $\mathcal{A} = (S, A, \rightarrow)$ be a finite LTS. Then for all r, s in S:

$\mathcal{A}$: $r \approx_b s$ if and only if $\forall \varphi \in$ CTL*-X, $\mathbf{tr}_1(\mathcal{A}), r_\perp \models \varphi \Leftrightarrow \mathbf{tr}_1(\mathcal{A}), s_\perp \models \varphi$. ♦

Clearly, due to Theorem 3.9, we can also replace CTL*-X with CTL-X in the above theorem.

Example 3.26. Consider the LTS $\mathcal{A}_{2.11}$ of Example 2.11.
If we define $\varphi = \exists$ ($\exists$ Fb) U a, then $\mathbf{tr}_1(\mathcal{A}_{2.11})$, $s_\perp \models \varphi$ but $\mathbf{tr}_1(\mathcal{A}_{2.11})$, $r_\perp \not\models \varphi$.
If we define $\varphi' = \exists$ ((a $\Rightarrow \forall G\neg$ c) U b), then $\mathbf{tr}_1(\mathcal{A}_{2.11})$, $q_\perp \models \varphi'$ but $\mathbf{tr}_1(\mathcal{A}_{2.11})$, $p_\perp \not\models \varphi'$. ♦

A disadvantage of the above transformation from LTS's to L^{2}TS's is that it may lead to a quadratic blowup in the size of the system (unless one assumes that the alphabet A is finite and fixed). Therefore, we will now describe a second, simpler transformation which is linear in the number of states and transitions. It was suggested in [EL85], where it is described in a setting without silent actions. The price to be paid is that corresponding states in the LTS and the L^{2}TS do no longer give rise to isomorphic unfoldings (Proposition 3.24). 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. In the definition below we give the details of the construction.

Definition 3.27. (*From LTS's to consistent L^{2}TS's: second transformation*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS. The L^{2}TS $\mathbf{tr}_2(\mathcal{A})$ is defined as $(S', A, \rightarrow', \mathcal{L})$ where
• $S' = S \cup \{(r,a,s) \mid a \in A \text{ and } r -a\rightarrow s\}$;
• $\rightarrow' = \{(r,\tau,s) \mid r -\tau\rightarrow s\} \cup \{(r,a,(r,a,s)) \mid r -a\rightarrow s\} \cup \{((r,a,s),a,s) \mid r -a\rightarrow s\}$;
• For $r, s \in S$ and $a \in A$: $\mathcal{L}(s) = \{\perp\}$ and $\mathcal{L}((r,a,s)) = \{a\}$. ♦

It is immediate from the definitions that $\mathbf{tr}_2(\mathcal{A})$ is a consistent L^{2}TS. The lower part of Example 3.23 shows how we will render with $\mathbf{tr}_2$ the same system which we used to exemplify $\mathbf{tr}_1$.

Proposition 3.28.
Two states r, s in an LTS $\mathcal{A}$ are branching bisimilar if and only if they are so in $\mathbf{tr}_2(\mathcal{A})$.

Proof: Straightforward from the definition. ♦

The next theorem is the analogue of Theorem 3.25 in a setting where $\mathbf{tr}_2$ is used instead of $\mathbf{tr}_1$.

Theorem 3.29. Let $\mathcal{A} = (S, A, \rightarrow)$ be a finite LTS. Then for all r, s in S:
$\mathcal{A}$: $r \approx_b s$ if and only if $\forall \varphi \in$ CTL*-X, $\mathbf{tr}_2(\mathcal{A})$, $r \models \varphi \Leftrightarrow \mathbf{tr}_2(\mathcal{A})$, $s \models \varphi$. ♦

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

If we define $\varphi = \exists\ (\exists\ Fb)\ U\ a$, then $\mathbf{tr}_2(\mathcal{A}_{2.11}), s \models \varphi$ but $\mathbf{tr}_2(\mathcal{A}_{2.11}), r \not\models \varphi$.
If we define $\varphi' = \exists\ ((a \Rightarrow \forall G\neg\ c)\ U\ b)$, then $\mathbf{tr}_2(\mathcal{A}_{2.11}), q \models \varphi'$ but $\mathbf{tr}_2(\mathcal{A}_{2.11}), p \not\models \varphi'$.
Notice that, although in general the transformations $\mathbf{tr}_1$ and $\mathbf{tr}_2$ lead to different models, the above distinguishing formulas are the same as the ones in Example 3.26. ♦

Remark 3.31. The logic CTL-X is more expressive than the Hennessy-Milner logic with until operator L_U, irrespective of whether transformation $\mathbf{tr}_1$ or $\mathbf{tr}_2$ is used; we leave it as an exercise to the reader to define the modalities of L_U in terms of CTL-X.

We conclude this section by introducing a new version of branching bisimulation which for finite systems is in full agreement with the stuttering equivalence of [BCG88] and thus with the equivalence induced by the standard interpretation of CTL* and CTL without the next-time operator. 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 3.32. (*Extending LTS's with livelocked state*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a finite LTS, let s_0 be a state not in S and let d be a distinct action not in A. Define the LTS $\mathcal{A}_d = (S', A', \rightarrow')$ where $S' = S \cup \{s_0\}$, $A' = A \cup \{d\}$ and
$\rightarrow' = \rightarrow \cup \{\langle s, d, s_0\rangle \mid s$ occurs in a τ-cycle or has no outgoing transitions$\}$. ♦

Definition 3.33. (*Divergence sensitive branching bisimulation*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a finite LTS. Two states r, s in S are *divergence sensitive branching bisimilar,* abbreviated ($\mathcal{A}$:) $r \approx_{dsb} s$, if and only if $\mathcal{A}_d$: $r \approx_b s$. ♦

Theorem 3.34. (*Stuttering equivalence and divergence sensitive branching bisimulation agree on consistent L^2TS's*)
If $\mathcal{AK} = (S, A, \rightarrow, \mathcal{L})$ is a finite and consistent L^2TS then for all r, s in S:
$\mathcal{AK}$: $r \approx_s s$ if and only if $\mathcal{AK}$: $r \approx_{dsb} s$ and $\mathcal{L}(r) = \mathcal{L}(s)$. ♦

The final theorem of this paper states that for both our transformations from LTS's to L^2TS's, divergence sensitive branching bisimulation equivalence coincides with the equivalence induced by CTL*-X under the standard interpretation.

Theorem 3.35.
Let $\mathcal{A} = (S, A, \rightarrow)$ be a finite LTS. Then for all r, s in S:
$\mathcal{A}$: $r \approx_{dsb} s$ if and only if $\forall \varphi \in$ CTL*-X, $\mathbf{tr}_1(\mathcal{A}), r_\perp \models_\mu \varphi \Leftrightarrow \mathbf{tr}_1(\mathcal{A}), s_\perp \models_\mu \varphi$

Theorem 3.35.

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

$\mathcal{A}$: $r \approx_{dsb} s$ if and only if $\forall \varphi \in$ CTL*-X, $\mathbf{tr}_1(\mathcal{A}), r_\perp \models_\mu \varphi \Leftrightarrow \mathbf{tr}_1(\mathcal{A}), s_\perp \models_\mu \varphi$

if and only if $\forall \varphi \in$ CTL*-X, $\mathbf{tr}_2(\mathcal{A}), r \models_\mu \varphi \Leftrightarrow \mathbf{tr}_2(\mathcal{A}), s \models_\mu \varphi$. ♦

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 which are in full agreement with branching bisimulation equivalence ($\approx_b$). The first logic, L_U, is an extension of Hennessy-Milner Logic with a kind of "until" operator. It is suggested by the definition of branching bisimulations in [GW89]. The second logic, L_{BF}, is another extension of Hennessy-Milner Logic which exploits the power of backward modalities; it stems directly from the alternative characterization of branching bisimulation presented in [DMV90]. The third logic is CTL* (see e.g. [ES89]) without the next-time operator (CTL*-X). The latter characterization exploits the relationships between variants of stuttering equivalence [BCG88] and CTL*.

The philosophy behind the backward generalization of HML is very similar to that of the logic called J_T, introduced by Hennessy and Stirling to deal with non-continuous properties of generalized transition systems with infinite computations [HS85]. 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) labelled transition systems, J_T has no more discriminating power than strong observational equivalence (see also [DMV90]). The characterization of $\approx_b$ in terms of a more abstract version of J_T gives strength to the claim that branching bisimulation is indeed a natural generalization of strong bisimulation and that it can be easily extended to cope with infinitary properties of systems. In the paper, we have shown that the modalities in L_U are definable in terms of those in L_{BF}. An interesting open question is whether each L_{BF} formula is logically equivalent to a L_U formula.

We have proved 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 (KS's) and Labelled Transition Systems (LTS's). Thus, transformation functions from one structure to the other were on demand. In the literature, various translations have been proposed. In [JKP90], 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 labelling, and by introducing a transition in correspondence of each pair of consecutive transitions in the LTS.

The construction of [JKP90], does not relate stuttering equivalence and branching bisimulation. This is partly accomplished by a (linear) translation from KS's to LTS's that was first

presented in [Kou90] (see our Definition 3.20) and by a translation from LTS's to KS's that is described in [CLM89]. However, the latter construction only works for a special class of LTS's which does not allow systems with states reachable via two paths which contain the same action resp. an even and an odd number of times.

One of the contributions of this paper is the definition of two transformation functions from general LTS's with invisible labels to KS's. The transformations permit naturally relating branching bisimulation to divergence blind stuttering equivalence and divergence sensitive branching bisimulation to stuttering equivalence. The first construction leads to a quadratic blowup of the states of the resulting KS but preserves the structure of the source LTS up to isomorphism of unfoldings. The second construction, which is inspired by a construction of [EL85] and generalizes it to deal with invisible labels, is linear and preserves structure too, although not in the sense of strong bisimulation equivalence. The second transformation permits moving freely between a state-based and an action-based representation of a concurrent system and thus indifferently using automatic tools which have designed for reasoning about either model.

To facilitate the discussion, we have also introduced a new kind of structures, Doubly Labelled Transition Systems (L^2TS), which are used as target of the translation functions from Labelled Transition Systems and Kripke structures. We have 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 and it is also worthwhile studying the equivalences which are obtained once the agreement is lost.

Finally, we would like to mention the work of Stirling [Sti89], who provides a different interpretation of CTL*-X by using LTS extended with the double arrow relation $=\alpha=>$ and 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.

5. Acknowledgements

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.

6. References

[BCG88] M.C. Browne, E.M. Clarke & O. Grümberg: Characterizing Finite Kripke Structures in Propositional Temporal Logic. *Theoret. Comp. Sci.*, **59** (1,2), 1988, pp. 115-131.

[CLM89] E.M. Clarke, D.E. Long & K.L. Macmillan: Compositional Model Checking. In *Proceedings 4th Annual Symposium on Logic in Computer Science (LICS)*, Asilomar, California, IEEE Computer Society Press, Washington, 1989, pp. 353-362.

[DeN87] R. De Nicola: Extensional Equivalences for Transition Systems, *Acta Informatica*, **24**, 1987, pp. 211-237.

[DIN90] R. De Nicola, P. Inverardi & M. Nesi: Using Axiomatic Presentation of Behavioural Equivalences for Manipulating CCS Specifications. In *Automatic Verification Methods for Finite State Systems* (J. Sifakis, ed.) Lecture Notes in Computer Science **407**, Springer-Verlag, 1990, pp. 54-67.

[DMV90] R. De Nicola, U. Montanari & F.W. Vaandrager: Back and Forth Bisimulations, 1990; to appear in *Proceedings CONCUR '90*, Amsterdam, Lecture Notes in Computer Science, Springer-Verlag, 1990.

[EH86] E.A.Emerson & J.Y. Halpern: "Sometimes" and "Not Never" Revisited: on Branching Time versus Linear Time Temporal Logic. *Journal of ACM*, **33**, 1, 1986, pp. 151-178.

[EL85] E.A. Emerson & C.L. Lei: Temporal model checking under generalized fairness constraints. In *18th Annual Hawaii International Conference on System Sciences.*

[ES89] E. A. Emerson & J. Srinivasan: Branching Time Temporal Logic. In [REX89], pp. 123-172.

[GV90] J.F. Groote & F.W. Vaandrager: An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence. CWI Report CS-R9001, Amsterdam, 1990; to appear in *Proceedings ICALP '90*, Warwick, Lecture Notes in Computer Science, Springer-Verlag, 1990.

[vGl90] R.J. van Glabbeek: Comparative Concurrency Semantics and Refinement of Actions, Ph.D. Thesis, Free University, Amsterdam, 1990.

[GW89] R.J. van Glabbeek & W.P. Weijland: Branching Time and Abstraction in Bisimulation Semantics (extended abstract). In *Information Processing '89* (G.X. Ritter, ed.), Elsevier Science Publishers B.V. (North Holland), 1989, pp. 613-618.

[HM85] M. Hennessy & R. Milner: Algebraic Laws for Nondeterminism and Concurrency. *Journal of ACM*, **32**, 1985, pp. 137-161.

[HS85] M. Hennessy & C. Stirling : The Power of the Future Perfect in Program Logics. *Information and Control*, **67**, 1985, pp. 23-52.

[JKP90] B. Jonsson, A.H. Khan & J. Parrow: 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 **407**, Springer-Verlag, 1990, pp. 179-188.

[Kou90] M. Koutny: Axiom system induced by CTL* logic, 1990. To appear in *Fundamenta Informaticae.*

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

[REX89] *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency,* (de Bakker, de Roever and Rozenberg, eds.) Lecture Notes in Computer Science **354**, Springer-Verlag, 1989.

[Sti89] C. Stirling: Temporal Logics for CCS, in [REX89], pp. 660-672.

[Sti90] C. Stirling: Modal and Temporal Logics, In *Handbook of Logic in Computer Science,* Vol I, (Abramsky, ed.), to appear, 1990.