Three Logics for Branching Bisimulation*

EXTENDED ABSTRACT

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 event-based approaches to the semantics of concurrent systems.

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 the same observable behaviour. The addition of a plausible notion of behavioural equivalence permits to overcome these problems; see [DeN87] and [vGl90] for comparative presentations.

Together with the definition of these 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 logics proposed 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 *strong* and *weak observational equivalence*, respectively. Other correspondences have been established in [BCG88]; two equivalences over Kripke frames (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 state with 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 correspo-

* 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 reasearch was carried out during a visit of the second author to Pisa which was supported by CEDISYS and LAMBRUSCO.

ndent 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 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 by limiting the formulae with the until operator to those in which φ is the constant true. 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 a back-and-forth bisimulation [DMV90]. It extends HML with a reverse operator (see [Sti89]). This operators permits inquiries to be made about the past of computations. The philosophy behind this generalization of HML is very similar to that of the logic called J_T in [HS85]; the relevant difference is that L_{BF} permits abstracting from silent actions, while J_T does not. Indeed, in the context of classic labelled transition systems, J_T has no more discriminating power than strong observational equivalence; it was introduced by Hennessy and Stirling to deal with non-continuous properties of generalized transition systems with fully visible infinite computations not obtainable as limits of finite ones. 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.

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 frames 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 a connection between the state- and event-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 ($\approx_s$) 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, we will call it divergence blind stuttering equivalence ($\approx_{dbs}$). Its definition is new; it is simpler than that of $\approx_s$, and naturally leads to a more efficient decision algorithm [GV90]. The definition of second equivalence, called divergence sensitive stuttering equivalence ($\approx_{dss}$) relies on the first and inherits its simplicity and the essence of its decision procedure. 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. Finally, we define a divergence sensitive version of branching bisimulation which coincides with $\approx_s$.

To relate branching bisimulation and stuttering equivalence, we introduce a general transformation function which, given a labelled transition system, yields an enriched system in which both states and transitions are labelled; the generated systems has the same structure as that of the original one: the unfolding of the two systems are isomorphic. 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.

Due to lack of space, all proofs will be omitted; they will be reported in the full version of the paper.

2. Branching Bisimulation and Hennessy-Milner Logics

In this section, we introduce two logical characterizations of branching bisimulation 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 $r\text{–}\alpha\!\rightarrow s$.

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 π, ... range over paths and ρ, σ, ... over runs. ♦

Definition 2.4. (*Many step transitions and bounded nondeterminism*)
i) Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS. For $a \in A$, we define on S, $r{=}a{=}{>}s$ if and only if there exists r' and s' in S such that $r{=}\varepsilon{=}{>}r'\text{–}a\!\rightarrow s'{=}\varepsilon{=}{>}s$; here $={\varepsilon}=>$ is the transitive and reflexive closure of $\text{–}\tau\!\rightarrow$.
ii) $\mathcal{A}$ has *bounded nondeterminism* iff for all $s \in S$ and for all $k \in A_\varepsilon$ 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 rRs and $r\text{–}\alpha\!\rightarrow r'$, then either $\alpha=\tau$ and r'Rs, or $\exists$ s_1, s' such that $s{=}\varepsilon{=}{>}s_1\text{–}\alpha\!\rightarrow s'$, r R s_1 and r' R s'.

- Two states r, 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 $s{=}\varepsilon{\Rightarrow}s_1$ to be related with r. The following lemma implies that this would have lead to the same equivalence relation.

Lemma 2.6. (cf. Lemma 1.3 of [GW89])
Let $\mathcal{A}$ = (S, A, →) be a LTS. Let for some $n > 0$, $(s_0,\tau,s_1) \ldots (s_{n-1},\tau,s_n)$ be a path with $s_0 \approx_b s_n$, then for all $0 \leq i \leq n$: $s_0 \approx_b s_i$. ♦

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

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

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 (HML) 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:

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

Definition 2.8. (*The satisfaction relation for HML*)
i) Let $\mathcal{A}$ = (S, A, →) be a LTS. The *satisfaction relation* $\models \subseteq S \times L_U$ 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 $\exists$ s' such that $s{=}k{=}{>}s'$ 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 the sense that it in order to consider equivalent two states it only requires them to lead via the same sequences of visible actions to equivalent states, without considering the intermediate states along the path. In order to take also the properties of these 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 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:

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

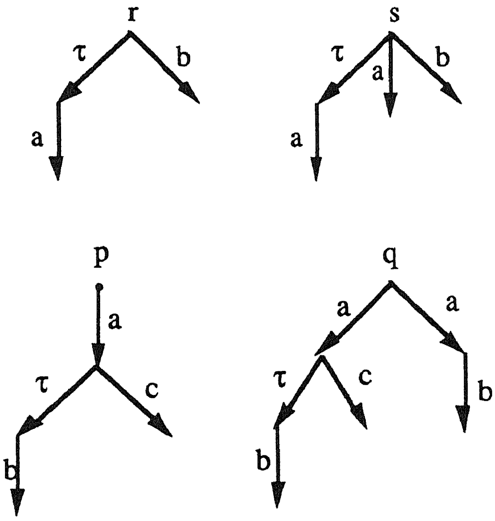
Definition 2.10. (*The satisfaction relation and the equivalence induced by L_U*)
Let $\mathcal{A}$ = (S, A, →) be a LTS. The *satisfaction relation* $\models \subseteq S \times L_U$ is defined inductively by:
- $s \models T$ always
- $s \models \neg\varphi$ iff $s \not\models \varphi$
- $s \models \varphi\wedge\varphi'$ iff $s \models \varphi$ and $s \models \varphi'$

- $s \models \varphi <k> \varphi'$ 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 $<k>\varphi$ for $T<k>\varphi$ and $[k]\varphi$ for $\neg<k>\neg\varphi$. It is worth noting that the original HML can be recovered from L_U in the sense that the diamond operator "$<k>\varphi$" of HML is rendered by "$T<k><\varepsilon>\varphi$". In the latter formula, we need to have $<\varepsilon>$ after $<k>$ because our 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 <r, s> and <p, q> are just two instances of the second and third τ'laws (see e.g. [HM85]), respectively, thus they certainly not differentiated by HML.

Example 2.11. *(Two pairs of processes which are weak observational equivalent but not branching bisimilar)*



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 $\approx_b$ induce the same identifications on bounded Labelled Transition Systems)* Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS with bounded nondeterminism. Then: $r \approx_b s$ iff $r \sim_{L_U} s$. ♦

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 action might be invisible while they deal with visible action 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 and it is proved that in the case of classical transition system without silent moves the equivalence induced by the logic with the past operator coincides with strong bisimulation.

Since we want to talk about the past of systems, we need to define our relations on runs rather than on single states; this enables us to go back from a state along the run which represents its *history*. Because of this, we introduce the notion of transition between runs:

- $\rho -\alpha\rightarrow \sigma$ if there exists a run $\theta = (s, (s, \alpha, s'))$ such that ρ concatenated with θ gives σ;
- $\rho =\varepsilon=> \sigma$ if there exist $\rho_0, \rho_1, \rho_2 \ldots \rho_n$, $(n \geq 0)$, 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

$$\rho =\varepsilon=> \rho' -\alpha\rightarrow \sigma' =\varepsilon=> \sigma.$$

More detailed discussions and motivations on the actual definitions 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 relation $R \subseteq \mathrm{run}_{\mathcal{A}}(r) \times \mathrm{run}_{\mathcal{A}}(s)$, 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 $\rho'\ R\ \sigma'$;

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

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

Theorem 2.14. (*Back and forth and branching bisimulation induce the same identifications on LTS' s*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS. $\mathcal{A}$: $r \approx_b s$ iff $\mathcal{A}$: $r \approx_{bf} s$. ♦

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:

$$\varphi ::= T \mid \neg\varphi \mid \varphi \wedge \varphi' \mid \langle k \rangle \varphi \mid \langle \overleftarrow{k} \rangle \varphi \qquad \blacklozenge$$

Definition 2.16. (*The Satisfaction Relation for L_{BF}*)
i) Let $\mathcal{A} = (S, A, \rightarrow)$ be a LTS. The *satisfaction relation* $\models \subseteq \mathrm{run}_{\mathcal{A}} \times L_{BF}$ 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 $\rho' \models \varphi$;
- $\rho \models \langle \overleftarrow{k} \rangle \varphi$ 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 action, 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 $[\overleftarrow{k}] = \neg\langle \overleftarrow{k} \rangle \neg$.
If $\varphi = \langle a \rangle [\overleftarrow{a}] \langle b \rangle T$ then $s \models \varphi$ while $r \not\models \varphi$.
If $\varphi' = [a][b]\langle \overleftarrow{b} \rangle \langle c \rangle T$ 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: $r \approx_b s$ iff $r \sim_{L_{BF}} s$. ♦

3. Branching Bisimulation and CTL*

In this section, we shall study the relationship of branching bisimulation with a different type of logic, the branching time 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 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 $\rightarrow 2^{AP}$ 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 infinite 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 $\mathrm{run}_{\mathcal{K}}(s)$, or just run(s), for the set of runs from s, and $\mu\mathrm{run}_{\mathcal{K}}(s)$, 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 $\pi = (s_0,s_1)(s_1,s_2)\ldots$, then first($\rho$)=s, path($\rho$)=$\pi$ and states(ρ)=$s_0 s_1 s_2$..
- 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 $\in$ **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 will write $\forall\pi$ 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 s

$\models \varphi$, and of a path formula π by a run ρ, notation $\rho \models \pi$, 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 $s \models_\mu \varphi$, and of a path formula π by a maximal run ρ, notation $\rho \models_\mu \pi$, is defined by replacing in the above definition $\models$ by $\models_\mu$ and the clause for $\exists\pi$ 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. Actually, we will give 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 equival.)*
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_1B_2... of states(ρ) and a partition $B'_1B'_2$... 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 $r \approx_{dbs} s$ and let $\rho \in run(r)$. Then there exists a $\sigma \in run(s)$ such that $\rho \approx_{dbs} \sigma$. ♦

Theorem 3.7.
If $r \approx_{dbs} s$, then for every CTL*–X formula φ:

$r \models \varphi$ iff $s \models \varphi$. ♦

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 $r \approx_{dbs} s$. ♦

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 $s \models \varphi$. ♦

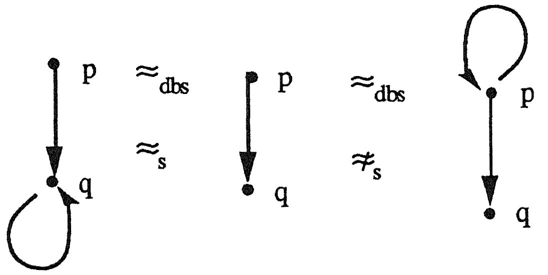
Now, we introduce the new version of stuttering equivalence which, for finite stare Kripke structures, can be proved to coincide with the original stuttering equivalence of [BCG88] and which does not ignore divergence. The new version is defined in terms of the previous one.

Definition 3.10. (*Extending Kripke frames 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 $\mathcal{K}_l$ by
$\mathcal{K}_l = (S', \mathcal{L}', \rightarrow')$ where $S' = S \cup s_0$, $\mathcal{L}' = \mathcal{L} \cup <s_0, p_0>$ and $\rightarrow' = \rightarrow \cup <s, s_0> \mid$ s has no outgoing transition or occurs in a cycle of states with the same label. ♦

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 $\mathcal{K}$: $r \approx_S s$ or $r \approx_S s$, if and only if $\mathcal{K}_l$: $r \approx_{dbs} s$.
ii) Two runs ρ, σ are *stuttering equivalent*, abbreviated $\mathcal{K}$: $\rho \approx_S \sigma$ or $\rho \approx_S \sigma$, if and only if $\mathcal{K}_l$: $\rho \approx_{dbs} \sigma$. ♦

The next example shows the different stress the two equivalences put on divergence (infinite repetition of the same state).

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



♦

Lemma 3.13.
Let $r \approx_S s$ and let $\rho \in \mu run(r)$. Then there exists a $\sigma \in \mu run(s)$ such that $\rho \approx_S \sigma$. ♦

Theorem 3.14. If $r \approx_S s$, then for every CTL*–X formula φ: $r \models_\mu \varphi$ iff $s \models_\mu \varphi$. ♦

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 $r \approx_S s$. ♦

By combining Theorems 3.14 and 3.15 we obtain the following:

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 $r, s \in S$. The following 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 $s \models_\mu \varphi$. ♦

As a corollary of the above theorem, we have that our version of stuttering equivalence coincides with that of [BCG88] for finite state Kripke Structures without deadlocked states.

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*)
Let **AP** be a fixed set of atomic proposition names.
An L^2TS is a structure $(S, A, \rightarrow, \mathcal{L})$ where $(S, A, \rightarrow)$ is a LTS and $\mathcal{L}: S \rightarrow 2^{AP}$ is a labelling function which associates a set of atomic propositions to each state. ♦

This definition 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.

Definition 3.18. (*Consistent L^2TS*)
A L^2TS $(S, A, \rightarrow, \mathcal{L})$ is *consistent* if there exist two functions
- *effect*: $2^{AP} \times A_\tau \rightarrow 2^{AP}$ and
- *action*: $2^{AP} \times 2^{AP} \rightarrow A_\tau$

such that
i) *effect*$(l, \tau) = l$
ii) *action*$(l, l) = \tau$
iii) $s \xrightarrow{\alpha} r$ implies $(\mathcal{L}(r) = $ *effect*$(\mathcal{L}(s), \alpha)$ and
$\alpha = $ *action*$(\mathcal{L}(s), \mathcal{L}(r)))$. ♦

What this definition amounts to saying is that states which are connected by an invisible action have the same labels and the labels of adjacent states are consistent with the label of the transition connecting them. The above restriction on L^2TS, 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. We do not give here the formal definitions of the two equivalences over the new structure; they are exactly the same as the original one for LTS and KS and completely ignore the label of the states and of the transitions respectively.

Theorem 3.19. (*Divergence blind stuttering and branching bisimulations agree on consistent L^2TS's)*
If $\mathcal{AK} = (S, A, \rightarrow, \mathcal{L})$ is a consistent L^2TS then for any pair of state r, s in S we have:

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

An example of how to build a L^2TS from a given LTS can be found in [CLM89]. There, a given LTS is extended by labelling each state with the set of the labelling 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 and is not able to cope with systems whose states can be reached via two paths which contain the same action an even and an odd number of times. Indeed, the authors restrict attention to those LTS's which lead to unique labelling. This restricted class of LTS's gives rise to consistent L^2TS's only.

We now propose a new transformation function which permits building a consistent L^2TS from any LTS. The transformation involves the introduction of new states, but a simple example below shows that this is unavoidable.

Definition 3.20. (*From LTS's to Consistent Doubly Labelled LTSs*)
Let $\underline{A} = \underline{a} \mid a \in A$ and $L = A \cup \underline{A}$.
tr: LTS $\rightarrow$ L^2TS is a function which given an LTS $\mathcal{A} =$ $(S, A, \rightarrow)$ yields a L^2TS
$\mathcal{AK} = (S', A, \rightarrow', \mathcal{L})$ where
- $S' = s_l \mid s \in S, l \in L$;
- $\rightarrow'$ is the least relation induced by the following rules

i) $s \xrightarrow{b} r$ and $a \neq b$ implies $s_a \xrightarrow{b}' r_b$ and $s_{\underline{a}} \xrightarrow{b}' r_{\underline{b}}$

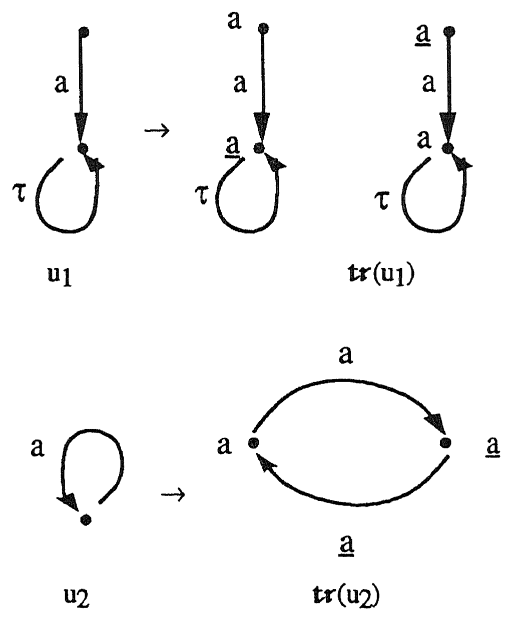
ii) $s \xrightarrow{a} r$ implies $s_a \xrightarrow{a}' r_{\underline{a}}$ and $s_{\underline{a}} \xrightarrow{a}' r_a$

iii) $s \xrightarrow{\tau} r$ implies $s_a \xrightarrow{\tau}' r_a$ and $s_{\underline{a}} \xrightarrow{\tau}' r_{\underline{a}}$

- $\mathcal{L}(s_l) = \{l\}$ for every $s \in S$, for every $l \in L$. ♦

Now, we give two examples of translation of labelled transition systems into doubly labelled ones. The two translations should evidence how, by means of the underlined labels, we avoid labelling states with invisible actions but are still able to give different labels to states of systems which are intuitively different. Had we not introduced the underlined labels, the only consistent labelling for the translation of u_1 was one which would associate an a also to the state in the τ-cycle, but this would have lead to identifying, via stuttering equivalence, the translations of u_1 and of u_2.

Example 3.21 (*Translating LTS's into L^2TS's*)



♦

Proposition 3.22. (***tr** yields consistent L^2TS's*)
Given an LTS $\mathcal{A}$, $\mathbf{tr}(\mathcal{A})$ is a consistent L^2TS. ♦

Proposition 3.23. (*The structure of $\mathcal{A}$ and $\mathbf{tr}(\mathcal{A})$ are very similar*)
If s is a state of a LTS $\mathcal{A}$ and $s_l \in \mathbf{tr}(\mathcal{A})$ then s and s_l give rise to isomorphic unfoldings. ♦

Now, Proposition 3.23 and Theorem 3.19, together with Theorem 3.9, allow us to prove the main theorem of this section.

Theorem 3.24.
If $\mathcal{A} = (S, A, \rightarrow)$ is a finite state LTS then for any pair of states r, s in S we have
$\mathcal{A}$: $s \approx_b r$ if and only if $\forall \varphi \in$ CTL*-X, $\forall l \in L$, $\mathbf{tr}(\mathcal{A})$: $s_l \models \varphi \Leftrightarrow \mathbf{tr}(\mathcal{A})$: $r_l \models \varphi$. ♦

Example 3.25.
Let p, q, r and s be as in Example 2.11, and suppose d is an element of A not in a, b, c.
If we define $\varphi = \exists\, (\exists\, Fb)\ U\ a$ then $s_d \models \varphi$ but $r_d \not\models \varphi$.
If we define $\varphi' = \exists\, ((d \vee \forall G \neg\, c)\ U\ b)$ then $q_d \models \varphi'$ but $p_d \not\models \varphi'$. ♦

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

We conclude this section by introducing a new version of branching bisimulation which 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 3.26. (*Extending LTS's with livelocked state*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a Labelled Transition System, let s_0 be a state not in S and let δ be a distinct action not in

A. Define the Labelled Transition System $\mathcal{A}_\delta = (S', A', \rightarrow')$ where $S' = S \cup s_0$, $A' = A \cup \delta$ and $\rightarrow' = \rightarrow \cup <s, \delta, s_0> \mid s$ has no outgoing transition or occurs in a τ-cycle. ♦

Definition 3.27. (*Divergence sensitive branching bisimulation*)
Let $\mathcal{A} = (S, A, \rightarrow)$ be a Labelled Transition System. Two states r, s in S are *divergence sensitive branching bisimilar*, abbreviated $\mathcal{A}$: $r \approx_{dsb} s$ or $r \approx_{dsb} s$, if and only if $\mathcal{A}_\delta$: $r \approx_b s$. ♦

Theorem 3.28. (*Stuttering and divergence sensitive branching bisimulation agree on consistent L^2TS' s)*
If $\mathcal{AK} = (S, A, \rightarrow, \mathcal{L})$ is a consistent L^2TS then for any pair of states r, s in S we have

$r \approx_s s$ if and only if $r \approx_{dsb} s$ and $\mathcal{L}(r) = \mathcal{L}(s)$. ♦

Theorem 3.29.
If $\mathcal{A} = (S, A, \rightarrow)$ is a LTS then for any pair of states r, s in S we have: $\mathcal{A}$: $s \approx_{dsb} r$ if and only if $\forall \varphi \in$ CTL*-X, $\forall l \in L$, $\mathbf{tr}(\mathcal{A})$: $s_l \models_\mu \varphi \Leftrightarrow \mathbf{tr}(\mathcal{A})$: $r_l \models_\mu \varphi$. ♦

4. 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.

5. 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. Mcmillan: 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] De Nicola, R., Inverardi, P. and Nesi, M. Using Axiomatic Presentation of Behavioural Equivalences for Manipulating CCS Specifications. In *Automatic Verification Methods for Finite State Machines* (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. Submitted for Publication.

[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.

[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; to appear in proc. ICALP '90.

[vGl90] R.J. Van Glabbeek: The Linear Time - Branching Time Spectrum; 1990. Submitted for Publication.

[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.

[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: Modal and Temporal Logics, In Handbook of Logic in Computer Science, Vol I, (Abramsky ed.), to appear, (1989).