Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

R.J. van Glabbeek, W.P. Weijland

Branching time and abstraction in bisimulation semantics
(extended abstract)

Computer Science/Department of Software Technology Report CS-R8911 April

---

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.).

Branching Time and Abstraction in Bisimulation Semantics

(extended abstract)

R.J. van Glabbeek and W.P. Weijland

Centre for Mathematics and Computer Science
Kruislaan 413,1098 SJ Amsterdam, The Netherlands

Abstract: In comparative concurrency semantics, one usually distinguishes between *linear time* and *branching time* semantic equivalences. Milner's notion of *observation equivalence* is often mentioned as the standard example of a branching time equivalence. In this paper we investigate whether observation equivalence really does respect the branching structure of processes, and find that in the presence of the unobservable action τ of CCS this is not the case.
Therefore the notion of *branching bisimulation equivalence* is introduced which strongly preserves the branching structure of processes, in the sense that it preserves computations together with the potentials in all intermediate states that are passed through, even if silent moves are involved. On closed terms, branching bisimulation can be completely axiomatized by the two laws:

$$x \cdot \tau = x$$
$$x \cdot (\tau \cdot (y + z) + y) = x \cdot (y + z).$$

For a large class of processes it turns out that branching bisimulation and observation equivalence are the same. All protocols known to the authors that have been verified in the setting of observation equivalence happen to fit in this class, and hence are also valid in the stronger setting of branching bisimulation equivalence.

Key words & phrases: semantics, process algebra, abstraction, bisimulation, branching time, concurrency.
1980 Mathematics subject classification: 68B10, 68C01, 68D25, 68F20
1985 Mathematics subject classification: 68Q55, 68Q45, 68Q10, 68N15
1987 CR Categories: F.3.2, F.4.3, F.1.2, D.3.1.

Note: This extended abstract will also appear in the proceedings of the 11th IFIP World Computer Congress, San Francisco 1989. It does not contain any proofs. They can be found in the full version of this paper, to appear as CWI-report.
This paper is sponsored in part by Esprit project no.432, METEOR.

INTRODUCTION

When comparing semantic equivalences for concurrency, it is common practice to distinguish between *linear time* and *branching time* equivalences (see for instance DE BAKKER, BERGSTRA, KLOP & MEYER [4], PNUELI [18]). In the former, a process is determined by its possible executions, whereas in the latter also the branching structure of processes is taken into account. The standard example of a linear time equivalence is *trace equivalence* as employed in HOARE [10] and REM [19]; the standard example of a branching time equivalence is *observation equivalence* or *bisimulation equivalence* as defined by MILNER [12] and PARK [16] (cf. [13, 14]). Furthermore,

Report CS-R8911
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

there are several *decorated trace equivalences* in between (cf. [2, 7, 8, 9, 11, 15, 17, 18]), preserving part of the branching structure of processes but for the rest resembling trace equivalence. Originally, the most popular argument for employing branching time semantics was the fact that it allows a proper modelling of *deadlock behaviour*, whereas linear time semantics does not. However, this advantage is shared with the decorated trace semantics which have the additional advantage of only distinguishing between processes that can be told apart by some notion of *observation* or *testing*. The main criticism on observation equivalence - and branching time equivalences in general - is that it is not an observational equivalence in that sense: distinctions between processes are made that cannot be observed or tested, unless observers are equipped with extraordinary abilities like that of a copying facility together with the capability of global testing as in ABRAMSKY [1].
Nevertheless, branching time semantics is of fundamental importance in concurrency, exactly because it is independent of the precise nature of observability. Which one of the decorated trace equivalences provides a suitable modelling of observable behaviour depends to a large extent on the tools an observer has, to test processes. And in general, a protocol verification in a particular decorated trace semantics, does not carry over to a setting in which observers are a bit more powerful. On the other hand, branching time semantics preserves the internal branching structure of processes and thus certainly their observable behaviour as far as it can be captured by decorated traces. A protocol, verified in branching time semantics, is automatically valid in each of the decorated trace semantics.

Probably one of the most important features in process algebra is that of abstraction, since it provides us with a mechanism to *hide* actions that are not observable, or not interesting for any other reason. By abstraction, some of the atomic steps in a process are made *invisible* or *silent*. Consequently, any consecutive execution of hidden steps cannot be recognized since we simply do not 'see' anything happen.
Algebraically, in ACP_τ of BERGSTRA & KLOP [6] abstraction has the form of a renaming operator which renames atomic steps into a *silent move* called τ. In MILNER's CCS [12] these silent moves result from synchronization. This new constant τ is introduced in the algebraic models as well: for instance in the *graph models* (cf. [12, 6]) we find the existence of τ-edges, and so the question was how to find a satisfactory extension of the original definition of *bisimulation equivalence* that we had on process graphs without τ.
One such possible extension is incorporated in Milner's notion of *observation equivalence*, which resembles ordinary bisimulation, but permits arbitrary sequences of τ-steps to precede or follow corresponding atomic actions. In a certain sense this notion of observation equivalence does not preserve the branching structure of a process. For instance, the processes $a\cdot(\tau\cdot b + c)$ and $a\cdot(\tau\cdot b + c) + a\cdot b$ are observation equivalent. However, in the first term, in each computation the

choice between b and c is made after the a-step, whereas the second term has a computation in which b is already chosen when the a-step occurs. For this reason one may wonder whether or not to accept the so-called third τ-law - $a\cdot(\tau\cdot x + y) = a\cdot(\tau\cdot x + y) + a\cdot x$ - (responsible for the former equivalence) and for similar reasons the second - $\tau\cdot x = \tau\cdot x + x$.
The previous example shows us that while preserving observation equivalence, we can introduce new paths in a graph that were not there before. To be precise: the *traces* are the same, but the sequences of intermediate nodes are different (modulo observation equivalence), since in the definition of observation equivalence there is no restriction whatsoever on the nature of the nodes that are passed through during the execution of a sequence of τ-steps, preceding or following corresponding atomic actions.
This is the key point in our new definition of abstraction: in two bisimilar processes every computation in the one process corresponds to a computation in the other, in such a way that all intermediate states of these computations correspond as well, due to the bisimulation relation. The equivalence which is thus obtained will be referred to as *branching bisimulation equivalence*. It turns out that it can be defined by a small change in the definition of observation equivalence.

1. ABSTRACTION

Assume that we work within the signature containing a set A of *atomic actions* a,b,c... which stand for observable actions, together with two infix written operators

- $+$ a binary operator for alternative composition
- $\cdot$ a binary operator for sequential composition.

As in regular algebra, we will often leave out brackets and $\cdot$, assuming that $\cdot$ will always bind stronger than +. In table 1 we find the algebraic theory BPA (cf. BERGSTRA & KLOP [5]) - which stands for *Basic Process Algebra* - which we will assume to hold for all processes in our signature:

$x + y = y + x$	A1
$(x + y) + z = x + (y + z)$	A2
$x + x = x$	A3
$(x + y)z = xz + yz$	A4
$(xy)z = x(yz)$	A5

Table 1. BPA.

The algebraic theory BPA is a logical theory from which we can derive certain statements about processes. In order to examine the implications of such equational theories we can build *models* for

BPA and try to achieve results about soundness and completeness. One such model is the *graph model*, which can be constructed by factoring out bisimulation equivalence on process graphs. Let us take a closer look at this particular model.

DEFINITION 1.1 A *process graph* (or *labelled transition system*) is a connected, rooted, edge-labelled and directed graph.

In an edge-labelled directed graph, edges go from one node to another (or the same) node and are labelled with elements from a given set. One can have more then one edge between two nodes as long as they carry different labels. A rooted graph has one special node which is indicated as the root node. Graphs need not be finite, but in a connected graph one must be able to reach every node from the root node by following a finite path. If r and s are nodes in a graph, then $r \rightarrow^a s$ denotes an edge from r to s with label a or it will be used as a *proposition* saying that such an edge exists. Process graphs represent concurrent systems in the following way: the edge-labels are *actions* a system may perform; the nodes of a process graph represent the states of a concurrent system; the root is the initial state and if $r \rightarrow^a s$, then the system can evolve from state r to state s by performing an action a.
For convenience, in this paper we will only consider *root unwound* process graphs, consisting of all those process graphs with no incoming edges at the root. This does not cause any loss of generality. Furthermore, we restrict ourselves to *non-trivial* graphs: graphs with at least one edge (which is a genuine restriction). The set of non-trivial, root unwound process graphs with labels from A will be denoted by $\boldsymbol{G}$.

DEFINITION 1.2 The constants $a \in A$ and the operators $+$ and $\cdot$ are defined on $\boldsymbol{G}$ as follows.

i. Constants $a \in A$ are denoted by one-edge graphs labelled with a;
ii. $g + h$ can be constructed by identifying the root nodes of g and h;
iii. $g \cdot h$ is constructed by identifying all endnodes (leaves) in g with the root of h. If g is without endnodes, then the result is just g.

In order to turn $\boldsymbol{G}$ into a *model* for BPA we need the notion of *bisimulation equivalence*, which originally was due to PARK [16] and used in MILNER [13] and in a different formulation already in MILNER [12]. Essentially, bisimulation equivalence is a congruence on $\boldsymbol{G}$ which preserves the branching structure of graphs. Its definition reads:

DEFINITION 1.3 Two graphs g and h in $\boldsymbol{G}$ are *bisimilar* (notation: $g \leftrightarroweq h$) if there exists a symmetric relation R between nodes of g and h (called the *bisimulation*) such that:

i. The roots of g and h are related by R

ii. If R(r,s) and $r \rightarrow^a r'$, then there exists a node s' such that $s \rightarrow^a s'$ and R(r',s').

A symmetric relation between nodes of g and h can be defined either as a relation $R \subseteq \text{nodes}(g) \times \text{nodes}(h) \cup \text{nodes}(h) \times \text{nodes}(g)$ such that $R(r,s) \Leftrightarrow R(s,r)$, or, alternatively, as a set of unordered pairs of nodes $R \subseteq \{\{r,s\}: r \in \text{nodes}(g), s \in \text{nodes}(h)\}$. In the latter case R(r,s) abbreviates $\{r,s\} \in R$.
Bisimilarity turns out to be an equivalence relation on $\boldsymbol{G}$ which is called *bisimulation equivalence*. Depending on the context we will sometimes use Milner's terminology and refer to bisimulation equivalence as *strong equivalence* or *strong congruence*.
It can be proved that $\leftrightarrows$ is a congruence on $\boldsymbol{G}$. Furthermore, let us say that a theory Γ is a *complete axiomatization* of a model $\boldsymbol{M}$ if for every pair of closed terms p and q we have: $\Gamma \vdash p=q$ if and only if $\boldsymbol{M} \models p=q$. Then the following theorem is from [6]:

THEOREM 1.1 BPA *is a complete axiomatization of* $\boldsymbol{G}/\leftrightarrows$.

A similar theorem for CCS occurred in [12]. Observe that in the presence of the trivial graph, BPA is not sound with respect to $\boldsymbol{G}/\leftrightarrows$. For this reason it was excluded from $\boldsymbol{G}$ from the beginning.

Now let us extend the signature of BPA with an extra constant $\tau \notin A$, and also introduce τ as a new edge-label. Since in this paper we want to avoid all complications connected with divergence, we restrict our attention to process graphs which are divergence free. Here, a process graph is *divergence free* if it does not contain an infinite path of τ-edges. Let $\boldsymbol{G}^*$ be the set of non-trivial, root unwound and divergence free process graphs with labels from $A \cup \{\tau\}$.
The definition of *strong congruence* was the starting point of MILNER [12] when he first considered abstraction in CCS. Having in mind that τ-steps are not observable, he suggested to simply require that if g and h are equivalent, (i) every possible a-step ($a \in A$) in the one graph should correspond with an a-step in the other (as in usual bisimulation semantics), apart from some arbitrary long sequences of τ-steps that are allowed to precede or follow, and (ii) every τ-step should correspond to some arbitrary long (≥ 0) τ-sequence. In this way he obtained his notion of *observation equivalence*. However, this equivalence turned out not to be a congruence, which he repaired by simply taking the closure of observation equivalence under contexts, thereby obtaining *observation congruence*. BERGSTRA & KLOP [6] found an additional condition in the definition of bisimulation - which is referred to as the *root condition* - turning observation equivalence into observation congruence or *rooted τ-bisimulation equivalence*.
Write $s \Rightarrow t$ for a path from s to t consisting of an arbitrary number (≥ 0) of τ-steps. Then MILNER's notion of observation equivalence (see [13, 14]) - or τ-bisimulation equivalence - can be defined as follows:

DEFINITION 1.4 Two graphs g and h are τ-*bisimilar* (notation: $g \leftrightarrows_\tau h$) if there exists a symmetric relation R between the nodes of g and h (called a τ-*bisimulation*) such that:

i. The roots are related by R

ii. If R(r,s) and $r \rightarrow^a r'$ ($a \in A \cup \{\tau\}$), then either $a=\tau$ and R(r',s), or there exists a path $s \Rightarrow s_1 \rightarrow^a s_2 \Rightarrow s'$ such that R(r',s').

Furthermore, g and h are *rooted* τ-*bisimilar* (notation: $g \leftrightarrows_{rt} h$) if we also have that

iii. (root condition) Root nodes are related to root nodes only.

In BERGSTRA & KLOP [6] it is proved that $\leftrightarrows_\tau$ is an equivalence and $\leftrightarrows_{rt}$ a congruence on G^*. Hence the relations are called *(rooted)* τ-*bisimulation equivalence*. Moreover, τ-bisimulation equivalence is also known as *observation equivalence*, and it is not difficult to see that rooted τ-bisimulation equivalence is the coarsest congruence contained in τ-bisimulation equivalence and hence coincides with *observation congruence*.
Milner proved that the resulting graph model $G^*/\leftrightarrows_{rt}$ satisfies the following three important τ-laws:

$x\tau = x$	T1
$\tau x = \tau x + x$	T2
$a(\tau x + y) = a(\tau x + y) + ax$	T3

Table 2. τ-laws ($a \in A \cup \{\tau\}$).

So we have $G^*/\leftrightarrows_{rt} \models$ T1-T3 which is an argument to include these three axioms in our system. Once again we are able to state a completeness theorem for $G^*/\leftrightarrows_{rt}$ saying that T1-T3 is *all* that we have extra in $G^*/\leftrightarrows_{rt}$ with respect to closed terms. It was first stated by MILNER [12] in the setting of CCS. In the setting of ACP_τ it was proved by BERGSTRA & KLOP [6].

THEOREM 1.2 BPA + T1-T3 *is a complete axiomatization of* $G^*/\leftrightarrows_{rt}$.

These τ-laws T1-T3 show us the implications of the particular choice of observation equivalence. Especially the laws T2 and T3 are quite surprising as a direct consequence of just eliminating sequences of τ-steps from computation paths. The reason for this is the fact that at least one basic feature in the construction of a bisimulation (see PARK [16]) is skipped, which is the property that any computation in the one process corresponds to a computation in the other process, in such a way that all intermediate states of these computations correspond as well due to the bisimulation relation. But look, in MILNER's observation equivalence, when satisfying the second requirement of

definition 1.4, one may execute arbitrarily many τ-steps in a graph without worrying about the status of the nodes that are passed in the meantime.

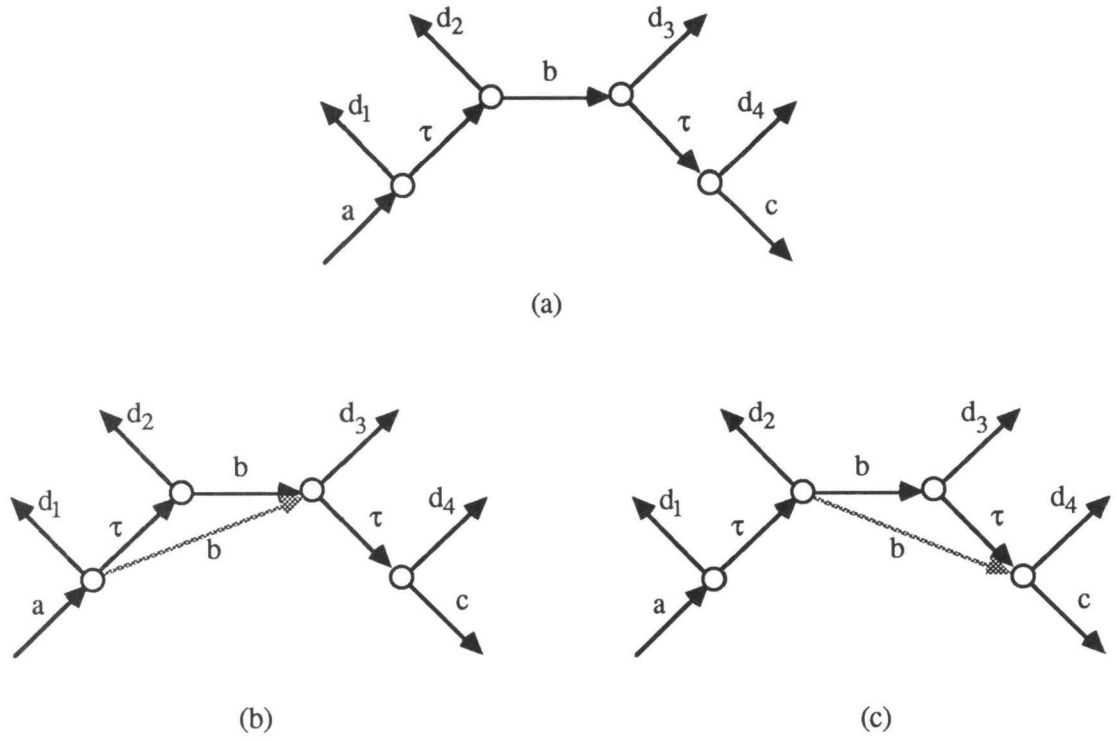

Figure 1. Observation equivalence.

As an illustration, in figure 1 we have a path $a\cdot\tau\cdot b\cdot\tau\cdot c$ with outgoing edges $d_1,\ldots,d_4$, and it follows easily that all three graphs are observation equivalent. Note that one may add extra b-edges as in (b) and (c) without disturbing equivalence. However, in both (b) and (c) a new computation path is introduced - in which the outgoing edge d_2 (or d_3 respectively) is missing - and such a path did not occur in (a). Or - to put it differently - in the path introduced in (b) the options d_1 and d_2 are discarded simultaneously, whereas in (a) it corresponds to a path containing a state where the option d_1 is already discarded but d_2 is still possible. Also in the path introduced in (c) the choice not to perform d_3 is already made with the execution of the b-step, whereas in (a) it corresponds to a path in which this choice is made only after the b-step. Thus we argue that observation equivalence does not preserve the branching structure of processes and hence lacks one of the main characteristics of bisimulation semantics.

Another observation tells us that the constructions (b) and (c) are highly fundamental for the behaviour of τ in the graph model. For instance, by simplifying figure 1(b) one immediately finds the second τ-law T2, whereas T3 can be easily found from figure 1(c). This shows us that the extra

τ-laws T2 and T3 actually originate from the fact that observation equivalence does not preserve branching structures.
Consider the following alternative definition of bisimulation in order to see how we can overcome this deficit.

DEFINITION 1.5 Two graphs g and h are *(rooted) branching bisimilar* if there exists a (rooted) τ-bisimulation R between the nodes of g and h with as an extra condition in definition 1.4 (ii) that $R(r,s_1)$ and $R(r',s_2)$.

Let us write $g \leftrightarrows_b h$ if g and h are branching bisimilar and $g \leftrightarrows_{rb} h$ if they are rooted branching bisimilar. In a picture, the difference between branching and τ-bisimulation can be characterized as follows:

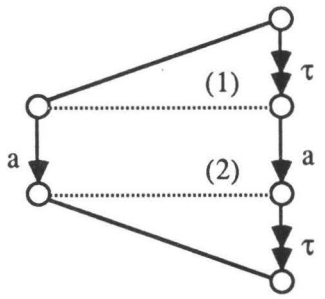

Figure 2. Bisimulations with τ.

The double arrow corresponds to the symbol $\Rightarrow$. Ordinary τ-bisimulation (see definition 1.4) says that every a-step $r \rightarrow^a r'$ corresponds with a path $s \Rightarrow s_1 \rightarrow^a s_2 \Rightarrow s'$ and so we obtain figure 2 *without* the lines marked with (1) and (2). Branching bisimulation moreover requires relations between r and s_1 and between r' and s_2 and thus we obtain figure 2 *with* (1) and (2). Note that in the presence of (2), i.e. $R(r',s_2)$, the existence requirement of a node s' with $s_2 \Rightarrow s'$ and R(r',s') is redundant. It follows immediately from the definitions that we have $g \leftrightarrows_{rb} h \Rightarrow g \leftrightarrows_{r\tau} h$.
Note that if $g \leftrightarrows_b h$ or $g \leftrightarrows_{rb} h$ then there exists a *largest* (rooted) branching bisimulation between g and h, since the set of (rooted) branching bisimulations is closed under arbitrary union.
Obviously, a branching bisimulation more strongly preserves the branching structure of a graph since the starting and endnodes of the τ-paths $s \Rightarrow s_1$ and $s_2 \Rightarrow s$ are related to the same nodes. Equivalently, we could have strengthened definition 1.4 (ii) by requiring *all* intermediate nodes in $s \Rightarrow s_1$ and $s_2 \Rightarrow s$ to be related with r and r' respectively. The fact that this alternative definition yields the same equivalence relation can be seen by use of the following lemma:

LEMMA 1.3 *Let* R *be the largest (rooted) branching bisimulation between* g *and* h.
If $r \to^\tau r_1 \to^\tau \cdots \to^\tau r_m \to^\tau r'$ $(m \geq 0)$ *is a path such that* R(r,s) *and* R(r',s) *then* $\forall 1 \leq i \leq m$: $R(r_i,s)$.

THEOREM 1.4 $\leftrightarrows_{rb}$ *is a congruence on* G^* *with respect to* + *and* $\cdot$. *Moreover it is the coarsest congruence contained in* $\leftrightarrows_b$.

The equivalence relation which is thus induced by a (rooted) branching bisimulation will be called *(rooted) branching bisimulation equivalence* or *(rooted) branching equivalence* for short.
Observe that in figure 1 there are no branching bisimulations among the graphs (a), (b) and (c). In particular, adding extra edges as in (b) and (c) no longer preserves branching bisimulation. For this reason we expect that the laws T2 and T3 will no longer hold. As it turns out, axiom T3 is completely dropped and T2 is considerably weakened to axiom H2 from the following table:

$x\tau = x$	H1 (T1)
$x(\tau(y + z) + y) = x(y + z)$	H2

Table 3. τ-laws for branching bisimulation.

H1 is the same axiom as T1 whereas H2 is a weaker version of T2 as one can check easily. Both axioms refer to the axiomatization of η, a constant for abstraction from BAETEN & VAN GLABBEEK [3] similar to τ. In fact, they are a variation on the first two η-laws in the sense that in [3] the second law H2 was only introduced for atomic actions x, instead of taking x as a general variable ranging over all processes. On the domain of closed terms, the two variants are equally powerful.
One can easily prove that H1 and H2 are preserved by rooted branching bisimulation. But we can do better, as is shown in the following theorem:

THEOREM 1.5 BPA + H1,H2 *is a complete axiomatization of* $G^*/\leftrightarrows_{rb}$.

SKETCH OF PROOF Let p and q be closed terms in BPA_τ such that $G^*/\leftrightarrows_{rb} \models p=q$. By applying the axioms A4 and A5, p and q can be rewritten into *basic* terms p' and q', only containing prefixing $a\cdot x$ instead of general sequential composition $x\cdot y$. We still have $G^*/\leftrightarrows_{rb} \models p'=q'$, or in other words: $[p'] \leftrightarrows [q']$, where [p] is the denotation of p in the graph domain G^*. Now one can show that by identifying isomorphic subgraphs and contracting inert τ-steps the graphs [p'] and [q'] can be transformed into the same normal form (up to graph isomorphism). Here, a τ-step $r \to^\tau s$ is *inert* if all options available in r are also available in s. On basic terms modulo axioms A1 and A2, these graph transformations correspond to applications of A3, H1 and H2.
Hence, $BPA+H1,H2 \vdash p=q$. □

2. BRANCHES AND TRACES

As we saw in figure 1, while preserving observation equivalence we are able to introduce 'new paths' in a graph. To be more precise: in these new paths alternative options may branch off at different places than in any of the old paths. So far, we claimed to have solved this problem by defining a new kind of bisimulation, but as of yet we still have to prove that our solution solves the problem in a fundamental way. In this section we will establish an alternative characterization of branching bisimulation. In fact, we will show the way in which branching bisimulation preserves the branching structure of graphs. Let us first consider ordinary bisimulation on process graphs without τ.

DEFINITION 2.1 A *trace* in a process graph $g \in \boldsymbol{G}$ is a finite sequence $(a_1, a_2, a_3, \ldots, a_k)$ of atoms from A, such that there exists a path $r_0 \rightarrow^{a_1} r_1 \rightarrow^{a_2} r_2 \rightarrow \cdots \rightarrow^{a_k} r_k$ from the root node r_0.

Two graphs g and h are said to be *trace equivalent*, notation $g \equiv_t h$, if their trace sets are equal. It is easily checked that $\equiv_t$ is a congruence on $\boldsymbol{G}$ and $g \leftrightarrows h \Rightarrow g \equiv_t h$. Consequently, we find that $\boldsymbol{G}/\equiv_t$ is a model for BPA. Compared to bisimulation, trace equivalence is much more rigid in its identifications. For example, we find that $\boldsymbol{G}/\equiv_t$ satisfies the equation $x(y + z) = xy + xz$ which cannot be proved from BPA.

The main reason for this is that in a trace we lose information about the potentials in the intermediate nodes. Therefore we cannot distinguish between processes $a(b + c)$ and $(ab + ac)$. In the following we will use *colours* at the nodes to indicate these potentials.

DEFINITION 2.2 A *coloured graph* is a process graph with colours $C \in \mathbf{C}$ as labels at the nodes.

Obviously, in a coloured graph we have traces which have colours in the nodes:

DEFINITION 2.3 A *coloured trace* in a coloured graph $g \in \boldsymbol{G}$ is a sequence

$$(C_0, a_1, C_1, a_2, C_2, \ldots, a_k, C_k)$$

for which there is a path $r_0 \rightarrow^{a_1} r_1 \rightarrow^{a_2} r_2 \rightarrow \cdots \rightarrow^{a_k} r_k$ in g, starting from the root node r_0, such that r_i has colour C_i.

The coloured traces of a node r in a graph g are the coloured traces of the subgraph of g that has r as its root node. This graph is obtained from g by deleting all nodes and edges which are inaccessible from r.

The question remains how to detect the colour of a node in a graph, or - to put it differently - how to define the concept of 'potential in a node' properly. There are several ways to do this. Probably the shortest definition is the following:

DEFINITION 2.4 A *consistent colouring* on a set of graphs is a colouring of their nodes with the property that two nodes have the same colour only if they have the same coloured trace set.

Obviously, the *trivial* colouring - in which every node has a different colour - is consistent on any set of graphs. Note that - even apart from the choice of the colours - a set of graphs can have more than one consistent colouring. For instance, consider a set containing only an infinite graph representing a^ω or $a{\cdot}a{\cdot}a\cdots$ then obviously the *homogeneous* colouring - in which every node has the same colour - is a consistent one, as well as the *alternating* or the trivial colouring.
Let us say two graphs $g,h \in \boldsymbol{G}$ are *coloured trace equivalent* - notation: $g \equiv_c h$ - if for some consistent colouring on $\{g,h\}$ they have the same coloured trace set. Then we have the following important characterization:

THEOREM 2.1 $g \leftrightarrows h$ *if and only if* $g \equiv_c h$.

Hence, on $\boldsymbol{G}$ the notion of coloured trace equivalence precisely coincides with bisimulation equivalence.

Next we will extend our definitions to $\boldsymbol{G}^* \supset \boldsymbol{G}$. In the following definition, we find how to abstract from τ-steps. The idea is simple: τ-steps can only be left out if they are *inert*, which says that they are between two nodes that have the same colour (potential). Thus it is not only that the inert steps are not observable but even more, they do not cause any change in the overall state of the machine.

DEFINITION 2.5 A *concrete coloured trace* in a coloured graph $g \in \boldsymbol{G}^*$ is defined as in definition 2.3, but treating τ exactly as an atom from A.
An *(abstract) coloured trace* in a coloured graph $g \in \boldsymbol{G}^*$ is a sequence

$$(C_0, a_1, C_1, a_2, C_2, \ldots, a_k, C_k)$$

which is obtained from a concrete coloured trace by replacing all subsequences of the form $(C, \tau, C, \tau, \ldots, \tau, C)$ by C.

DEFINITION 2.6 On subsets of $\boldsymbol{G}^*$ a consistent colouring is defined exactly as on subsets of $\boldsymbol{G}$.
Furthermore, such a colouring is *rooted* if all root nodes have the same unique colour, not occurring anywhere else in the graphs.

For two graphs $g,h \in \boldsymbol{G}^*$ let us also write $g \equiv_c h$ if for some consistent colouring on $\{g,h\}$ they have the same coloured trace set, and $g \equiv_{rc} h$ if moreover this colouring is rooted. Then we find the following characterization for (rooted) branching bisimulation:

THEOREM 2.2

i. $g \leftrightarrows_b h$ *if and only if* $g \equiv_c h$

ii. $g \leftrightarrows_{rb} h$ *if and only if* $g \equiv_{rc} h$.

This characterization provides us with a clear intuition about what branching bisimulation actually is, since the difference between *inert* steps - not changing the state of the machine - and relevant τ-steps - that behave as common atomic actions - is visualized immediately by the (change of) colours at the nodes. It follows that branching bisimulation equivalence preserves computations together with the potentials in all intermediate nodes that are passed through.

As a tool for further analysis we have the following proposition:

PROPOSITION 2.3 *For every process graph* $g \in \boldsymbol{G}^*$ *the largest rooted branching autobisimulation - relating* g *with itself - induces a rooted consistent colouring of its nodes, the so-called canonical colouring.*

The maximal rooted branching autobisimulation on g is an equivalence relation on the nodes. Proposition 2.3 says that every node can be labelled with its equivalence class as a colour, in order to obtain a consistent colouring on $\{g\}$.

DEFINITION 2.7 Let $g \in \boldsymbol{G}^*$ be a process graph and consider its canonical colouring with colour set **C**. Let N(g) - the *normal form* of g - be the graph which can be found from g by contracting all nodes with the same colour. To be precise:

1. N(g) has colours $C \in \mathbf{C}$ as its nodes.
2. N(g) has an edge $C \rightarrow^a C'$ ($a \in A$) iff g has an edge $r \rightarrow^a r'$ such that $C(r)=C$ and $C(r')=C'$, where C(r) denotes the colour of the node r.
3. N(g) has an edge $C \rightarrow^\tau C'$ iff $C \neq C'$ and g has an edge $r \rightarrow^\tau r'$ with $C(r)=C$ and $C(r')=C'$.

PROPOSITION 2.4 *For all process graphs* $g \in \boldsymbol{G}^*$: $g \leftrightarrows_{rb} N(g)$.

So every graph $g \in \boldsymbol{G}^*$ can be turned into normal form preserving rooted branching equivalence. In every rooted branching equivalence class, these normal forms turn out to be unique up to graph isomorphisms as is shown by the following:

DEFINITION 2.8 A *graph isomorphism* is a bijective relation R between the nodes of g and h such that:

1. the roots of g and h are related by R
2. if R(r,s) and R(r',s') then $r \rightarrow^a r'$ is an edge in g iff $s \rightarrow^a s'$ is an edge in h ($a \in A \cup \{\tau\}$).

Two graphs are isomorphic - notation $g \cong h$ - iff there exists an isomorphism between them. In that case g and h only differ with respect to the identity of the nodes. Note that $\cong$ is a congruence relation on process graphs.

THEOREM 2.5 (normal form theorem)

$g \leftrightarrows_{rb} h$ *if and only if* $N(g) \cong N(h)$.

3. CORRESPONDENCE

Let us end with a theorem which tells us that in quite a number of cases observation and branching equivalence are the same. For instance, consider the practical applications where implementations are verified by proving them equal to some specification (after having abstracted from a set of unobservable actions of course). In many such cases, the *specification* does not involve any τ-steps at all: in fact all τ-steps that occur in the verification process originate from the abstraction procedure which is carried out on the implementation.

As it turns out, in all such cases there is no difference between observation and branching bisimulation equivalence. For this reason we may expect many verifications involving observation equivalence to be valid in the stronger setting of branching bisimulation as well. In particular this is the case for all protocol verifications in τ-bisimulation semantics known to the authors.

THEOREM 3.1 *Suppose* g *and* h *are two graphs, and* g *is without edges labelled with* τ. *Then:*

i. $g \leftrightarrows_{\tau} h$ *if and only if* $g \leftrightarrows_{b} h$

ii. $g \leftrightarrows_{r\tau} h$ *if and only if* $g \leftrightarrows_{rb} h$.

Finally, observe that there exists a close relationship between rooted and non-rooted branching bisimulation, since the root condition (definition 1.4(iii)) only works on the root nodes:

THEOREM 3.2 *For all graphs g and h we have:*

i. $g \leftrightarrows_{\tau} h$ *if and only if* $\tau \cdot g \leftrightarrows_{r\tau} \tau \cdot h$

ii. $g \leftrightarrows_{b} h$ *if and only if* $\tau \cdot g \leftrightarrows_{rb} \tau \cdot h$.

From theorem 3.2 we easily find that for graphs g and h:

$$g \text{ is without } \tau\text{-edges} \Rightarrow (\tau \cdot g \leftrightarrows_{r\tau} \tau \cdot h \Rightarrow \tau \cdot g \leftrightarrows_{rb} \tau \cdot h).$$

REFERENCES

[1] S.ABRAMSKY, *Observation equivalence as a testing equivalence*, TCS 53, pp.225-241, 1987.

[2] J.C.M.BAETEN, J.A.BERGSTRA & J.W.KLOP, *Ready trace semantics for concrete process algebra with the priority operator*, The Computer Journal 30 (6), pp.498-506, 1987.

[3] J.C.M.BAETEN & R.J.VAN GLABBEEK, *Another look at abstraction in process algebra*, proc. 14th ICALP (Th.Ottman ed.) Karlsruhe, Springer LNCS 267, pp.84-94, 1987.

[4] J.W.DE BAKKER, J.A.BERGSTRA, J.W.KLOP & J.-J.Ch.MEYER, *Linear time and branching time semantics for recursion with merge*, proc. 10th ICALP (J.Diaz ed.) Barcelona, Springer LNCS 154, pp.39-51, 1983.

[5] J.A.BERGSTRA & J.W.KLOP, *Process algebra for synchronous communication*, Information & Control 60 (1-3), pp. 109-137, 1984.

[6] J.A.BERGSTRA & J.W.KLOP, *Algebra of communicating processes with abstraction*, TCS 37 (1), pp.77-121, 1985.

[7] B.BLOOM, S.ISTRAIL & A.R.MEYER, *Bisimulation can't be traced: preliminary report*, conf. record of the 15th POPL, San Diego, California, pp.229-239, 1988.

[8] S.D.BROOKES, C.A.R.HOARE & A.W.ROSCOE, *A theory of communicating sequential processes*, Journal ACM 31 (3), pp.560-599, 1984.

[9] R.DE NICOLA & M.C.B.HENNESSY, *Testing equivalences for processes*, TCS 34, pp.83-133, 1984.

[10] C.A.R.HOARE, *Communicating sequential processes*, On the construction of programs (R.M.McKeag & A.M. Macnaghten eds.), Cambridge University Press, pp.229-254, 1980.

[11] C.A.R.HOARE, *Communicating sequential processes*, Prentice Hall, London 1985.

[12] R.MILNER, *A calculus of communication systems*, Springer LNCS 92, 1980.

[13] R.MILNER, *Calculi for synchrony and asynchrony*, TCS 25, pp.267-310, 1983.

[14] R.MILNER, *Lectures on a calculus for communicating systems*, Seminar on concurrency (S.D.Brookes, A.W.Roscoe & G.Winskel eds.), Springer LNCS 197, pp.197-220, 1985.

[15] E.-R. OLDEROG & C.A.R.HOARE, *Specification-oriented semantics for communicating processes*, Acta Informatica 23, pp.9-66, 1986.

[16] D.M.R.PARK, *Concurrency and automata on infinite sequences*, proc. 5th GI conf. on Theor.Comp.Sci. (P.Deussen ed.), Springer LNCS 104, pp.167-183, 1981.

[17] I.C.C.PHILLIPS, *Refusal testing*, TCS 50, pp.241-284, 1987.

[18] A.PNUELI, *Linear and branching structures in the semantics and logics of reactive systems*, proc. 12th ICALP (W.Brauer ed.) Nafplion, Springer LNCS 194, pp.15-32, 1985.

[19] M.REM, *Trace theory and systolic computations*, proc. PARLE conf. (J.W.de Bakker, A.J.Nijman & P.C.Treleaven eds.) Eindhoven, Vol.1, Springer LNCS 258, pp.14-33, 1987.