1991

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

Branching time and abstraction in bisimulation semantics

Computer Science/Department of Software Technology Report CS-R9120 March

CWI, nationaal instituut voor onderzoek op het gebied van wiskunde en informatica

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

Rob van Glabbeek, Peter Weijland

Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB 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 CCS-terms branching bisimulation can be completely axiomatized by the single axiom scheme:

$$a.(\tau.(y + z) + y) = a.(y + z)$$

(where a ranges over all actions) and the usual laws for strong congruence.
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, action refinement.
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.

Notes: This paper appeared before as Chapter III of VAN GLABBEEK (1990). The first two sections partly appeared in WEIJLAND (1989), an extended abstract appeared as VAN GLABBEEK & WEIJLAND (1989a) and section 5 as VAN GLABBEEK & WEIJLAND (1989b).
The Research of the authors was supported by ESPRIT project 432 (METEOR). The first author was also supported by Sonderforschungsbereich 342 of the TU München and by the ESPRIT Basic Research Action 3148 (DEMON); the second author by ESPRIT project 3006 (CONCUR). This paper was finalized when the first author was employed at the Technical University of Munich. The current affiliations of the authors are:
R.J. van Glabbeek, Computer Science Department, Stanford University, Stanford, California 94305, USA. rvg@cs.stanford.edu
W.P. Weijland, Software Engineering Research Center, PO Box 424, 3500 AK Utrecht, The Netherlands. weijland@serc.nl.

Report CS-R9120
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

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 (1983), PNUELI (1985)). 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 (1980); the standard example of a branching time equivalence is *observation equivalence* or *bisimulation equivalence* as defined by MILNER (1980) and PARK (1981) (cf. MILNER (1983), MILNER (1989)). Furthermore, there are several *decorated trace equivalences* in between (cf. BAETEN, BERGSTRA & KLOP (1987b), BLOOM, ISTRAIL & MEYER (1988), BROOKES, HOARE & ROSCOE (1984), DE NICOLA & HENNESSY (1984), HOARE (1985), OLDEROG & HOARE (1986), PHILLIPS (1987), PNUELI (1985), POMELLO (1986)), 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 (1987).

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 in great extend 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 actions 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 (1985) abstraction has the form of a renaming operator which renames actions into a *silent move* called τ. In MILNER's CCS (MILNER (1980))

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. BERGSTRA & KLOP (1985), MILNER (1980)) 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.
It turns out that there exist many possibilities for extending bisimulation equivalence to process graphs with τ-steps. One such possible extension is incorporated in Milner's notion of *observation equivalence* - called *τ-bisimulation equivalence* in BERGSTRA & KLOP (1985) -, which resembles ordinary bisimulation, but permits arbitrary sequences of τ-steps to precede or follow corresponding atomic actions. A different notion of so-called *η-bisimulation* was suggested by BAETEN & VAN GLABBEEK (1987) invoking a weaker set of abstraction axioms. In MILNER (1981) another notion of observational equivalence was introduced which in this paper is referred to as *delay bisimulation equivalence.* As we will show, the treatments of Milner and Baeten & Van Glabbeek fit into a natural structure of four possible variations of bisimulation equivalence involving silent steps. The structure is completed by defining *branching bisimulation equivalence.* As it turns out, observation equivalence is the coarsest equivalence of the four, in the sense of identifying most processes. η- and delay bisimulation equivalence are two incomparable finer notions whereas branching bisimulation equivalence is the finest of all.
In a certain sense the usual 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 we should accept the so-called third τ-law - $a\cdot(\tau\cdot x + y) = a\cdot(\tau\cdot x + y) + ax$ - (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 definition of branching bisimulation equivalence: 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. It turns out that it can be defined by a small change in the definition of observation equivalence.
The fact that observation equivalence is too rigid in its identifications is even stronger illustrated by the problems that it may cause in practical applications and analysis. As an example, it can be shown (cf. GRAF & SIFAKIS (1987)) that there is no modal logic with eventually operator $\blacklozenge$ which is adequate for observation equivalence. Here $\blacklozenge\phi$ means that all paths will eventually pass to a state were ϕ holds. Indeed, suppose that such a logic *would* exist, then this means that two processes are

observation equivalent iff they satisfy the same modal formulas. For instance, with respect to processes in CCS there exists a formula f such that: (c.nil + τ.b.nil) ⊨ ϕ and b.nil ⊭ ϕ since obviously both processes are not observation equivalent. However, from (c.nil + τ.b.nil) ⊨ ϕ it follows that we have a.(c.nil + τ.b.nil) ⊨ ♦ϕ whereas from b.nil ⊭ ϕ we find a.(c.nil + τ.b.nil) + a.b.nil ⊭ ♦ϕ although both processes are observation equivalent. Obviously, this inconsistency is due to the third τ-law.
Another paper by JONSSON & PARROW (1989) on deciding bisimulation equivalence shows a different kind of struggle with the third τ-law. In this paper, infinite data flow is turned into a finite state representation by considering symbolic transitions. This provides us with a method to decide on the equivalence of infinite data flow programs. It turns out to work easily for strong equivalence, but in observation equivalence there is no straightforward generalization of the former results and a less intuitive transition system is needed to fix this problem. Using branching bisimulation may serve as a key to a more natural solution of this problem.

Having at least four options for the definition of bisimulation congruence involving τ-steps, in any particular application it becomes important to have a clear intuition about which kind of abstraction is preferable. In an important class of problems one can prove however, that all four notions of bisimulation yield the same equivalence. In particular this is the case if one of the two bisimulating graphs does not have any τ-steps. It is interesting to observe that all case studies on protocol verification known to the authors fit into this class of problems, hence all of their proofs that have been given in the setting of observation equivalence still hold in branching bisimulation semantics .

1. BRANCHING AND ABSTRACTION

In this section we define the semantic equivalences that we want to discuss on a domain of *process graphs*. Since we focus on branching and abstraction, we have chosen to abstain from a proper modelling of divergence, concurrency, real-time behaviour and stochastical aspects of processes. Moreover, we will disregard the nature of the actions that our processes may perform: they will be modelled as uninterpreted symbols a,b,c,... from a given set Act. We have chosen process graphs (or labeled transition systems) to represent processes, since they clearly visualize the aspects of the modelled systems' behaviour we are interested in. The nodes in our graphs (or states in our transition systems) remain anonymous. A common alternative is to use closed expressions in a process specification language like CCS or ACP as nodes in process graphs, but here we prefer to separate the semantic issues from the treatment of a particular language. In the next section, however, we will give an interpretation of certain subsets of CCS and ACP in (parts of) the graph model and discuss the algebraic aspects of our equivalences.

DEFINITION 1.1 A *process graph* is a connected, rooted, edge-labeled and directed graph.

In an edge-labeled directed graph, edges go from one node to another (or the same) node and are labeled with elements from a certain set Act. One can have more than 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. We require process graphs to be connected: they need not be finite, but 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 elements of Act 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. The domain of process graphs will be denoted by G.
On G we consider the notion of *bisimulation equivalence*, which originally was due to PARK (1981) and used in MILNER (1983, 1985, 1989) and in a different formulation already in MILNER (1980). On the domain of process graphs, a bisimulation usually is defined as a relation $R \subseteq \text{nodes}(g) \times \text{nodes}(h)$ on the nodes of graphs g and h satisfying:

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

Equivalently - as is done in this paper - one can obtain bisimulation equivalence from a *symmetric* relation R between nodes of g and h, only satisfying (i) and (ii). Such a symmetric relation can be defined 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 $\{r,s\} \in R$. Note that this restriction to symmetric relations does not cause any loss of generality.

DEFINITION 1.2 Two graphs g and h in G are *bisimilar* - notation: $g \leftrightarrows h$ - if there exists a symmetric relation R between the nodes of g and h (called a *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')

Bisimilarity turns out to be an equivalence relation on 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*.
Now let us postulate the existence of a special action $\tau \in$ Act, that represents an unobservable, internal move of a process. We write $r \Rightarrow s$ for a path from r to s consisting of an arbitrary number (≥ 0) of τ-steps.
The definition of strong congruence was the starting point of MILNER (1980) when he considered abstraction in CCS. Having in mind that τ-steps are not observable, he suggested to simply require that for g and h to be equivalent, (i) every possible a-step ($a \neq \tau$) in the one graph should correspond with an a-step in the other (as for usual bisimulation equivalence), apart from some arbitrary long

sequences of τ-steps that are allowed to precede or follow, and (ii) every τ-step should correspond to an arbitrary long (≥ 0) τ-sequence. This way he obtained his notion of *observation equivalence* (cf. MILNER (1980, 1983, 1985, 1989) - or *τ-bisimulation equivalence* - which can be defined as follows:

DEFINITION 1.3 Two graphs g and h are *τ-bisimilar* - notation: $g \leftrightarroweq_\tau h$ - if there exists a symmetric relation R (called a *τ-bisimulation*) between the nodes of g and h such that:

i. The roots are related by R
ii. If R(r,s) and $r \rightarrow^a r'$, 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').

Again, $\leftrightarroweq_\tau$ is an equivalence on $\boldsymbol{G}$ which is called *τ-bisimulation equivalence*, also known as *observation equivalence* or *weak equivalence.*

To some extend, the notion of τ-bisimulation cannot be regarded as the natural generalization of ordinary bisimulation to an abstract setting with hidden steps. The reason for this is the fact that an important feature of a bisimulation is missing for τ-bisimulation, which is the property that any 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. When HENNESSY & MILNER (1980) introduced the first version of observation equivalence, they also insisted on relating the intermediate states of computations, as they tell us: "... any satisfactory comparison of the behaviour of concurrent programs must take into account their intermediate states as they progress through a computation, because differing intermediate states can be exploited in different program contexts to produce different overall behaviour ..." and: "If we consider a computation as a sequence of experiments (or communications), then the above remarks show that the intermediate states are compared. In fact, if p is to be equivalent to q, there must be a strong relationship between their respective intermediate states. At each intermediate stage in the computations, the respective "potentials" must also be the same". However, in Milner's observation equivalence, when satisfying the second requirement of definition 1.3 one may execute arbitrarily many τ-steps in a graph without worrying about the status of the nodes that are passed through in the meantime.

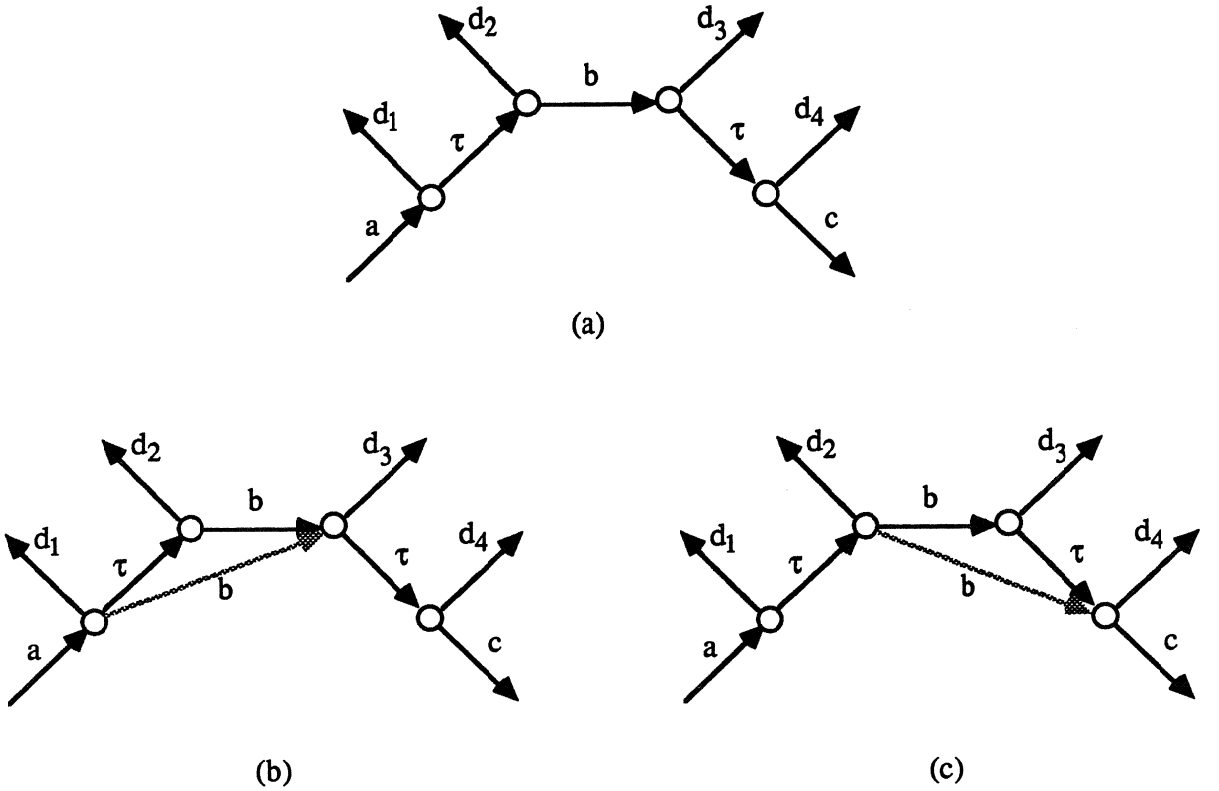


FIGURE 1. Observation equivalence.

As an illustration, in figure 1 we consider a path $a \cdot \tau \cdot b \cdot \tau \cdot c$ with outgoing edges $d_1,...,d_4$, 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.

Consider the following alternative definition of bisimulation in order to see how we can overcome this deficit.

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

i. The roots are related by R
ii. If R(r,s) and $r \rightarrow^a r'$, 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 $R(r,s_1)$, $R(r',s_2)$ and R(r',s').

In a picture, the difference between branching and τ-bisimulation can be characterized as follows:

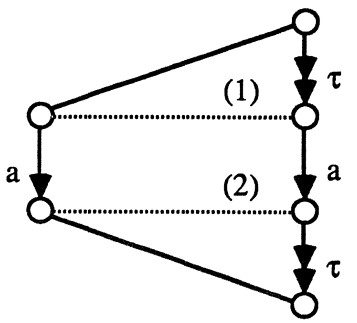


FIGURE 2. Bisimulations with τ.

The double arrow corresponds to the symbol $\Rightarrow$. Ordinary τ-bisimulation (definition 1.3) 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 if $g \leftrightarrows_b h$ then there exists a *largest* branching bisimulation between g and h, since the set of branching bisimulations is closed under arbitrary union. One can easily check that branching bisimilarity is an equivalence on *G*, referred to as *branching bisimulation equivalence* or *branching equivalence* for short.

Obviously, branching equivalence 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. Observe that in figure 1 there are no branching bisimulations between any of the graphs (a), (b) and (c). In particular, adding extra edges as in (b) and (c) no longer preserves branching equivalence. Equivalently, we could have strengthened definition 1.3 (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 1.1 (stuttering lemma) *Let* R *be the largest branching bisimulation between* g *and* h.
If $r \rightarrow^\tau r_1 \rightarrow^\tau \cdots \rightarrow^\tau r_m \rightarrow^\tau r'$ $(m \geq 0)$ *is a path such that* R(r,s) *and* R(r',s) *then* $\forall 1 \leq i \leq m$: $R(r_i,s)$.

PROOF First we prove lemma 1.1 for a slightly different kind of bisimulation, defined as follows:

DEFINITION A *semi branching bisimulation* between two graphs g and h is a symmetric relation R between the nodes of g and h such that:

i. The roots are related by R
ii. If R(v,w) and $v \rightarrow^a v'$ then either
   (a) $a=\tau$ and there exists a path $w \Rightarrow w'$ such that R(v,w') and R(v',w'), or:
   (b) there exists a path $w \Rightarrow w_1 \rightarrow^a w_2 \Rightarrow w'$ such that $R(v,w_1)$, $R(v',w_2)$ and R(v',w').

The difference with branching bisimulation is in case (a), which can be illustrated by:

FIGURE 3. Semi branching (left) and branching bisimulation.

Now let (*) denote the property, mentioned in the lemma. Observe that (a) any branching bisimulation is a semi branching bisimulation and (b) any semi branching bisimulation satisfying (*) is a branching bisimulation.

CLAIM *The largest semi branching bisimulation between* g *and* h *satisfies* (*).

Let R be the largest semi branching bisimulation between g and h, let s be a node and let $r \rightarrow^\tau r_1 \rightarrow^\tau \cdots \rightarrow^\tau r_m \rightarrow^\tau r'$ ($m \geq 0$) be a path such that R(r,s) and R(r',s). Then we prove that $R' = R \cup \{\{r_i,s\}: 1 \leq i \leq m\}$ is a semi branching bisimulation. We check the conditions:
(i) Clearly, the root nodes of g and h are related by R' (since by R).
(ii) Suppose R'(v,w) and $v \rightarrow^a v'$. If R(v,w) then it follows that either (a) $a=\tau$ and there exists a path $w \Rightarrow w'$ such that R(v,w') and R(v',w'), or (b) there exists a path $w \Rightarrow w_1 \rightarrow^a w_2 \Rightarrow w'$ such that $R(v,w_1)$, $R(v',w_2)$ and R(v',w'). Hence, from $R \subseteq R'$ we find that R' satisfies the requirements in the definition above.
So assume *not* R(v,w), then we find that either (1) v=s and $w=r_i$ or (2) $v=r_i$ and w=s.
(1) If $s \rightarrow^a s'$ then it follows from R(r',s) that
either: $a=\tau$ and there is a path $r' \Rightarrow r''$ such that R(r'',s) and R(r'',s'). Hence there is a path $r_i \Rightarrow r' \Rightarrow r''$ such that R'(r'',s) and R'(r'',s') as required.
or: there is a path $r' \Rightarrow t_1 \rightarrow^a t_2 \Rightarrow r''$ such that $R(t_1,s)$, $R(t_2,s')$ and R(r'',s') and hence $r_i \Rightarrow r' \Rightarrow t_1 \rightarrow^a t_2 \Rightarrow r''$ with $R'(t_1,s)$, $R'(t_2,s')$ and R'(r'',s').
(2) If $r_i \rightarrow^a r''$ then $r \rightarrow^\tau r_1 \rightarrow^\tau \cdots \rightarrow^\tau r_i \rightarrow^a r''$ and since R(r,s) we find that there exists a sequence $s \Rightarrow s_1 \Rightarrow \cdots \Rightarrow s_i$ such that $R(r_1,s_1),\ldots,R(r_i,s_i)$. It follows from $R(r_i,s_i)$ that

either: $a=\tau$ and there exists a path $s_i \Rightarrow s'$ such that $R(r_i,s')$ and $R(r'',s')$. Hence $s \Rightarrow s'$ with $R'(r_i,s')$ and $R'(r'',s')$ as required.
or: there exists a path $s_i \Rightarrow t_1 \rightarrow^a t_2 \Rightarrow s''$ such that $R(r_i,t_1)$, $R(r'',t_2)$ and $R(r'',s'')$, and hence $s \Rightarrow s_i \Rightarrow t_1 \rightarrow^a t_2 \Rightarrow s''$ with $R'(r_i,t_1)$, $R'(r'',t_2)$ and $R'(r'',s'')$.
This proves that R' is a semi branching bisimulation. Since R is the largest we find R=R'.

So we proved the claim. Finally, conclude that the largest semi branching bisimulation is equal to the largest branching bisimulation, and thus we proved the lemma. □

The stuttering lemma will play a crucial role in some of the results we will present later.
It follows from figure 2 that we can find two more kinds of bisimulation with τ, since we can leave out (1) while still having (2) and vice versa. Consider the following two definitions:

DEFINITION 1.5 Two graphs g and h are η-*bisimilar* - notation $g \leftrightarrows_\eta h$ - if there exists a symmetric relation R (called an η-*bisimulation*) between the nodes of g and h such that:
i. The roots are related by R
ii. If $R(r,s)$ and $r \rightarrow^a r'$, 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 $R(r,s_1)$ and $R(r',s')$.

DEFINITION 1.6 Two graphs g and h are *delay bisimilar* - notation $g \leftrightarrows_d h$ - if there exists a symmetric relation R (called a *delay bisimulation*) between the nodes of g and h such that:
i. The roots are related by R
ii. If $R(r,s)$ and $r \rightarrow^a r'$, 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 $R(r',s_2)$ and $R(r',s')$.

Notice the subtle differences between both definitions (and definition 1.4). In definition 1.5 the notion of η-bisimulation corresponds to figure 2 without the relation (2) but with (1). Similarly, with delay bisimulation we have (2) but not (1). It is easy to see that in the definition of both branching and delay bisimulation the existence requirement of a node s' such that $s_2 \Rightarrow s'$ and $R(r',s')$ is redundant.
From the definitions we find immediately that $g \leftrightarrows_b h \Rightarrow g \leftrightarrows_\eta h \Rightarrow g \leftrightarrows_\tau h$ and similarly $g \leftrightarrows_b h \Rightarrow g \leftrightarrows_d h \Rightarrow g \leftrightarrows_\tau h$. Observe that in figure 1 we find an η-bisimulation between (a) and (c) and a delay bisimulation between (a) and (b). Conversely, there is no η-bisimulation between (a) and (b) and no delay bisimulation between (a) and (c), so all implications are strict.
The notion of η-bisimulation was first introduced by BAETEN & VAN GLABBEEK (1987) as a finer version of observation equivalence. A variant of delay bisimulation - only differing in the treatment of divergence - first appeared in MILNER (1981), also under the name observational equivalence.

HISTORICAL NOTE:
The first semantic equivalence preserving the branching structure of processes was defined in HENNESSY & MILNER (1980) and MILNER (1980). In MILNER (1980) it was called *strong equivalence* or *strong congruence*. It was defined in terms of a decreasing sequence $\sim_0, \sim_1, ..., \sim_k, ...$ of equivalence relations. Originally, these relations where defined on CCS expressions that figured as states in transition systems, but one can also define them on nodes of (possibly different) process graphs.

DEFINITION 1.7 Let r and s be nodes of process graphs. Then:
$r \sim_0 s$ is always true
$r \sim_{k+1} s$ iff for all $a \in Act$
(i) if $r \rightarrow^a r'$ then there exists a node s' such that $s \rightarrow^a s'$ and $r' \sim_k s'$
(ii) if $s \rightarrow^a s'$ then there exists a node r' such that $r \rightarrow^a r'$ and $r' \sim_k s'$
$r \sim s$ iff for all $k \in \mathbf{N}$: $r \sim_k s$.
Two graphs g and h are *strongly equivalent*, notation $g \sim h$, if root(g) ~ root(h).

A process graph is *finitely branching* if each node has only finitely many outgoing edges. In HENNESSY & MILNER (1980) and MILNER (1980) strong congruence was defined only on CCS expressions corresponding with finitely branching graphs. On this domain, as was shown in MILNER (1980), strong congruence 'satisfies its definition' in the following sense:

PROPOSITION 1.2 *Let* r *and* s *be nodes of finitely branching process graphs.*
Then $r \sim s$ *iff for all* $a \in$ Act:
i. if $r \rightarrow^a r'$ *then there exists a node* s' *such that* $s \rightarrow^a s'$ *and* $r' \sim s'$
ii. if $s \rightarrow^a s'$ *then there exists a node* r' *such that* $r \rightarrow^a r'$ *and* $r' \sim s'$.

Strong equivalence is closely related to the notion of *bisimulation*, introduced by PARK (1981) (cf. definition 1.2). It is easy to verify that any bisimulation is included in each of the relations $\sim_k$ for $k \in \mathbf{N}$. Hence bisimulation equivalence is at least as discriminating as strong equivalence. On the other hand, from the former proposition it follows that with respect to finitely branching process graphs strong equivalence is a bisimulation, and hence the two notions coincide. With respect to infinitely branching graphs, ~ is strictly coarser than bisimulation equivalence as can be seen from the following example. Consider the graphs

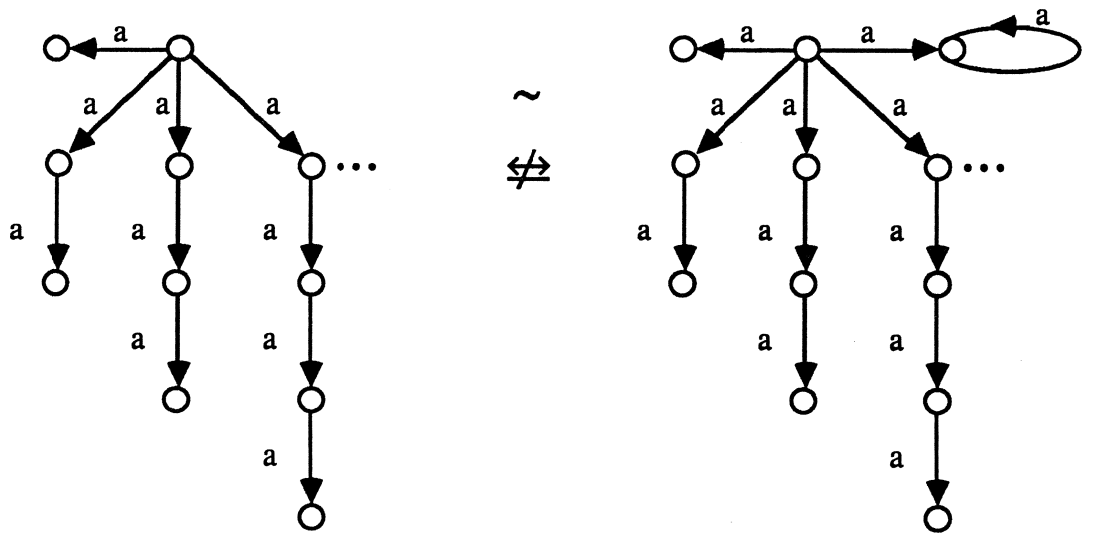


FIGURE 4. 'Strongly equivalent' graphs that are not bisimilar.

One can easily verify that these graphs are strongly equivalent in the sense of definition 1.7, but not bisimilar.

PROPOSITION 1.3

i. With respect to finitely branching process graphs the notions ~ and ⇆ coincide;

ii. With respect to infinitely branching process graphs ⇆ is strictly contained in ~.

Starting from this observation, there are two different ways in which the notion of strong equivalence (in HENNESSY & MILNER (1980) and MILNER (1980) defined on finitely branching processes only) can be extended to infinitely branching process graphs. In MILNER (1983) strong equivalence is chosen to be the relation of definition 1.2, so strong equivalence and bisimulation equivalence are synonyms.

In the presence of a special action τ, representing an unobservable move of a process, one looks for a semantic equivalence that abstracts from internal moves in a process and for the rest resembles bisimulation equivalence. Such an abstract equivalence has to satisfy requirements such as:

- it is coarser than bisimulation equivalence
- it is equal to bisimulation equivalence with respect to processes not containing τ-edges
- it does not discriminate between the graphs

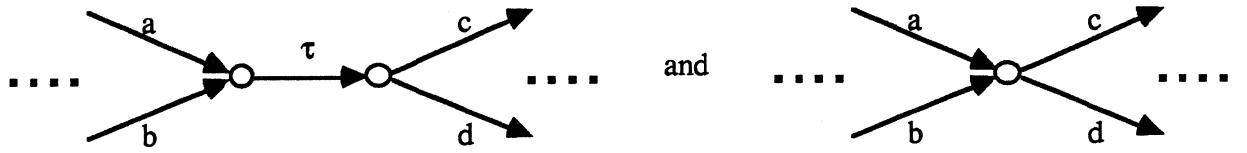


FIGURE 5. Contraction of internal moves.

The definition of strong congruence (~) was the starting point of HENNESSY & MILNER (1980) when they introduced abstraction in CCS. Having in mind that τ-steps are not observable, they proposed that two process graph g and h are equivalent if every visible step in the one graph corresponds with a similar step in the other, apart from some arbitrarily long sequences of τ-steps that are allowed to precede or follow. This way they obtained a notion of *observational equivalence*. Originally, this relation was defined in the style of definition 1.7, but in order to facilitate comparison with the other equivalences, we will present it in bisimulation style.

DEFINITION 1.8 Two graphs g and h are *observational equivalent* in the sense of HENNESSY & MILNER if there exists a symmetric relation R between the nodes of g and h such that:

i. The roots are related by R
ii. If R(r,s) and $r \rightarrow^a r'$ ($a \neq \tau$), then there exists a path $s \Rightarrow s_1 \rightarrow^a s_2 \Rightarrow s'$ such that R(r',s').

Unfortunately, this type of observational equivalence turned out not to be a congruence for the CCS parallel composition operator, the free merge, or any other operator representing concurrent activity (cf. HENNESSY & MILNER (1980)). Furthermore, we argue that it is not resistent against *refusal testing* as developed in PHILIPS (1987) (Refusal testing is essentially the testing notion of MILNER (1981), but without replication facility).

EXAMPLE Consider the following two process graphs:

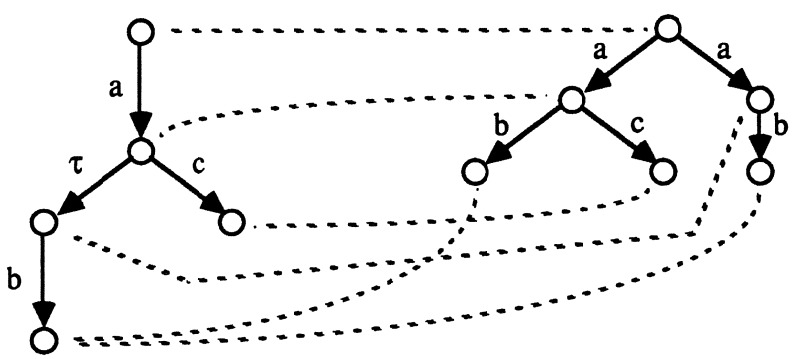


FIGURE 6. Observational equivalence does not respect refusal testing.

These graphs are observational equivalent in the sense of HENNESSY & MILNER (1980); the relation R has been indicated in the figure above. Now expose them to the experiments a, d and c (in this order). The process on the right may respond as follows: a is accepted, d is refused and c is accepted (another possible respons would be: a accepted, d refused and c refused). However, this respons would not be possible in the process on the left: the attempt to execute the action d would cause the τ-edge to be executed, and then c cannot happen anymore.

Hence MILNER's version of observation equivalence (MILNER (1980)) (which we call τ-bisimulation equivalence) can be regarded as an improvement. Both notions satisfy the requirements mentioned above, but additionally τ-bisimulation equivalence is a congruence for the CCS parallel composition operator and is resistent against refusal testing. Since observational equivalence in the sense of HENNESSY & MILNER (1980) is coarser than τ-bisimulation equivalence, the criticism that τ-bisimulation equivalence does not preserve the branching structure of processes also applies to the variant of HENNESSY & MILNER (1980).

2. AXIOMS

In this section we will turn several parts of our graph domain G into algebras, by defining some operations on them. This will enable us to give equational characterizations of the equivalences studied in the previous section. In the first subsection we use the operators of the axiom system BPA_τ (cf. BERGSTRA & KLOP (1985)): action constants, alternative and sequential composition. In the second subsection we take the operators inaction, prefixing and alternative composition of CCS (cf. MILNER (1980)). Finally, in the third subsection we combine the features of the previous two approaches, thereby obtaining the kernel of the extended algebra ACP_τ (cf. BERGSTRA & KLOP (1985)). We will not consider parallel composition, restriction (or encapsulation), hiding and relabeling. However, we claim that these can be added without problem.

2.1. BASIC PROCESS ALGEBRA

For sake of convenience, in this subsection we will only consider *root unwound* process graphs, i.e. process graphs with no incoming edges at the root. Since each bisimulation equivalence class of process graphs contains a root unwound graph, this does not cause any loss of generality. Furthermore, we restrict ourselves to *non-trivial graphs* - having at least one edge - and we assume our graphs to be *divergence free*, meaning that they do not contain infinite τ-paths. The latter restriction will be cancelled later, but for the time being it suits us since having it we can stay closer to CCS in our presentation. (NOTE: apart from arguments about presentation, one may argue that there is still discussion about the role of divergence in bisimulation equivalence on processes, such as the dichotomy between explicit divergence MILNER (1981), WALKER (1990) and fair abstraction MILNER (1980), BAETEN, BERGSTRA & KLOP (1987a), see also section 6.3). The domain of root

unwound, non-trivial and divergence free process graphs will be denoted by G_{BPA}. Clearly $G_{BPA} \subseteq G$.
In order to equip G_{BPA} with some structure, we introduce two binary infix written operators + and · and constants for every element in Act.

DEFINITION 2.1 The constants $a \in$ Act and the operators + and · are defined on G_{BPA} as follows:

(i) Constants $a \in$ Act are interpreted by one-edge graphs labeled with a
(ii) (g + h) can be constructed by identifying the root nodes of g and h
(iii) (g·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.

As in regular algebra we will often leave out brackets and ·, assuming that · will always bind stronger than +.
The operators + and · are well-defined, even after deviding out *bisimulation equivalence* on G_{BPA}, as follows from the following proposition, the proof of which is straightforward and omitted.

PROPOSITION 2.1 *Bisimulation equivalence is a congruence with respect to the operators + and ·.*

Hence the structure $(G_{BPA}/\leftrightarrows,+,\cdot,\text{Act})$ is a well-defined algebra. Considering its first order equational theory we find the axiom system BPA (cf. BERGSTRA & KLOP (1984)) which stands for *Basic Process Algebra.*

$x + y = y + x$	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.

As usual, we assume the axioms from table 1 to be universally quantified.
Now let us say that a theory Γ is a *complete axiomatization* of a model M if for every pair of closed terms p and q we have: $\Gamma \vdash p=q$ if and only if $M \models p=q$. This definition deviates from the standard one, since usually also open terms are considered. Then the following theorem is due to BERGSTRA & KLOP (1985):

THEOREM 2.2 BPA *is a complete axiomatization of* $(G_{BPA}/\leftrightarrows,+,\cdot,\text{Act})$.

Observe that in the presence of the trivial graph, BPA is not sound with respect to ($G_{BPA}/\leftrightarrows$,+,·,Act): axiom A4 no longer holds, with the trivial graph substituted for the variable y. For this reason it was excluded from G_{BPA} from the beginning.
In the same way one may wish to find axiomatizations for algebras resulting from deviding out the other equivalences of section 1. However, as it turns out these equivalences are no congruences with respect to the operator +. In the case of observation equivalence this problem was solved by MILNER (1980) by simply taking the closure of $\leftrightarrows_\tau$ with respect to all contexts in CCS, thereby obtaining *observation congruence*. Similarly in HENNESSY & MILNER (1980) *observational congruence* was defined as the CCS-closure of their variant of obsevational equivalence (definition 1.8) and this congruence coincides with the one of MILNER (1980). BERGSTRA & KLOP (1985) formulated an additional condition, yielding an immediate definition of observation congruence by means of bisimulation relations.

DEFINITION 2.2 (root condition) A relation R between nodes of process graphs is called *rooted* if root nodes are related with root nodes only.

Observe that every bisimulation (see definition 1.2) is rooted, but this is not necessarily the case for the relations defined in definitions 1.3-1.6. For any two process graphs g and h and $* \in \{\tau,b,\eta,d\}$ we write R: $g \leftrightarrows_{r*} h$ if R is a rooted $*$-bisimulation between g and h, and $g \leftrightarrows_{r*} h$ if such a relation exists.

THEOREM 2.3 *For* $* \in \{\tau,b,\eta,d\}$, $\leftrightarrows_{r*}$ *is a congruence on* G_{BPA} *with respect to* + *and* ·.

PROOF We prove theorem 2.3 for $\leftrightarrows_{rb}$. The other proofs proceed in the same way.
$\leftrightarrows_{rb}$ is reflexive since the identity relation is a rooted branching bisimulation between any graph and itself, and it is symmetric by definition. Furthermore, assume that R: $g \leftrightarrows_{rb} g'$ and S: $g' \leftrightarrows_{rb} g''$ and define: $T(r,r'') :\Leftrightarrow$ for some r' in g': R(r,r') and S(r',r''). Now one can easily prove that T: $g \leftrightarrows_{rb} g''$, and so $\leftrightarrows_{rb}$ is transitive. Thus we proved that $\leftrightarrows_{rb}$ is an equivalence. So it is left to prove that $\leftrightarrows_{rb}$ respects the operators. Suppose that R: $g \leftrightarrows_{rb} g'$ and S: $h \leftrightarrows_{rb} h'$.
$\pm$: We prove that $(R \cup S)$: $(g + h) \leftrightarrows_{rb} (g' + h')$.
(i) Obviously the roots of (g + h) and (g' + h') are related.
(ii) Assume that in (g + h) we have an edge $r \rightarrow^a r'$ and suppose we have $(R \cup S)(r,s)$ then from the construction of (g + h) it follows that this edge either originates from g or from h; let us say it is g. It follows from $(R \cup S)(r,s)$ that we have either R(r,s) or S(r,s), so we have two distinct cases:
Firstly, suppose that R(r,s). Then either $a=\tau$ and R(r',s) - hence $(R \cup S)(r',s)$ and $(R \cup S)$ satisfies definition 1.4 - or there exists a path $s \Rightarrow s_1 \rightarrow^a s_2 \Rightarrow s'$ in g' such that $R(r,s_1)$, $R(r',s_2)$ and R(r',s'). Obviously, we can find the same path in (g' + h') and we have that $(R \cup S)(r,s_1)$, $(R \cup S)(r',s_2)$ and $(R \cup S)(r',s')$ as required.

Secondly, suppose that we do *not* have R(r,s). Then we have S(r,s), and since we assumed that the edge $r \rightarrow^a r'$ came from g, we find that r has to be the (joint) root node of g and h. However, in S root nodes are related with root nodes only (the root condition), and so s must be the joint root node of g' and h'. Hence R(r,s), which is a contradiction.
(iii) Obviously, the root nodes of (g + h) and (g' + h') are uniquely related by (R∪S).
∴ we prove R∪S: $(g \cdot h) \leftrightarrows_{rb} (g' \cdot h')$.
(i) Clearly, the roots of both graphs are related by R, hence by R∪S.
(ii) Assume that in (g·h) we have an edge $r \rightarrow^a r'$ and suppose we have (R∪S)(r,s) then from the construction of (g·h) it follows that this edge either originates from g or from h.
(1) Firstly, let us say it is from g. From (R∪S)(r,s) we find that either R(r,s) or S(r,s). Since r cannot be an endnode in g we have R(r,s). It follows from the fact that R is a rooted branching bisimulation that either $a=\tau$ and R(r',s) - hence (R∪S)(r',s) as required - or there is a path $s \Rightarrow s_1 \rightarrow^a s_2 \Rightarrow s'$ in g' such that $R(r,s_1)$, $R(r',s_2)$ and R(r',s') and thus we find the same path in (g'·h') such that $(R \cup S)(r,s_1)$, $(R \cup S)(r',s_2)$ and (R∪S)(r',s'), as is required.
(2) Secondly, assume $r \rightarrow^a r'$ is from h.
- In case R(r,s), we find that r is an endnode in g (since those are the only nodes of g that are identified with nodes from h). Suppose s is an endnode in g', then it is identified with the root node of h', and since S is a rooted branching bisimulation we find:
either $a=\tau$ and S(r',s), hence (R∪S)(r',s);
or there exists a path $s \Rightarrow s_1 \rightarrow s_2 \Rightarrow s'$ such that $S(r,s_1)$, $S(r',s_2)$ and S(r',s') and hence $(R \cup S)(r,s_1)$, $(R \cup S)(r',s_2)$ and (R∪S)(r',s') as required.
So let us assume that s is *not* an endnode in g', then it has at least one outgoing edge $s \rightarrow^b s_1$. Since R is a rooted branching bisimulation and R(r,s), we find that $b=\tau$ and $R(r,s_1)$. The same argument holds for s_1 and thus we find a path $s=s_0 \rightarrow^\tau s_1 \rightarrow^\tau s_2 \rightarrow^\tau \ldots$ such that $R(r,s_i)$. Since all graphs in G_{BPA} are divergence free we have that all nodes s_i are distinct and furthermore the sequence $s=s_0 \rightarrow^\tau s_1 \rightarrow^\tau s_2 \rightarrow^\tau \ldots$ has bounded length. Hence there exists a path $s \Rightarrow s'$ to an endnode s' in g', such that R(r,s') (and hence (R∪S)(r,s')). Note that s' is identified with the root node of h'. Combining this result with the former part, we find that the conditions of definition 1.4 are satisfied as required.
- In case *not* R(r,s), then S(r,s) and both r and s are from h and h' respectively. Now the requirement follows immediately from the fact that S is a branching bisimulation.
(iii) Clearly, the root nodes are uniquely related by (R∪S). □

THEOREM 2.4 *Provided that there exists al least one action* $a \in$ Act *with* $a \neq \tau$, $\leftrightarrows_{r*}$ *is the coarsest congruence on* G_{BPA} *with respect to* + *that is contained in* $\leftrightarrows_*$, *for* $* \in \{\tau,b,\eta,d\}$. *Hence* $\leftrightarrows_{r\tau}$ *coincides with observation congruence.*

PROOF The idea for this proof is due to J.W. Klop (personal communication). Let g and $h \in G_{BPA}$ and suppose $g+k \leftrightarrows_* h+k$ for any graph $k \in G_{BPA}$. Suppose there is an action $a \in$ Act ($a \neq \tau$) that

does not occur in g and h. Then $g+a \leftrightarrows_* h+a$. Let R be a *-bisimulation between g+a and h+a, then R must be rooted. Therefore the restriction of R to the nodes of g and h is a rooted *-bisimulation between g and h.
If no 'fresh atom' $a \in Act$ can be found a variant of this method still works. First note that for each infinite cardinal κ there are at least κ *-bisimulation equivalence classes of graphs with less then κ nodes. (Choose an action $a \in Act$ ($a \neq \tau$) and define for each ordinal $\lambda > 0$ the graphs g_λ as follows: $g_1 = a$, $g_{\lambda+1} = g_\lambda + a g_\lambda$ and for λ a limit ordinal g_λ is contructed from all graphs g_μ for $\mu < \lambda$ by identifying their roots. Then with transfinite induction it follows that no two different g_λ's are *-bisimilar. Furthermore, for infinite λ, the cardinality of the nodes of g_λ is the cardinality of λ.) Thus for any two graphs g and h there must be a graph $k \in G_{BPA}$ with the same cardinality such that k is not bisimilar with any subgraph corresponding with a node in g or h. Now take a *-bisimulation between $g+\tau k$ and $h+\tau k$. □

The equivalence relations $\leftrightarrows_{r*}$ are called *rooted *-bisimulation equivalence* or *-bisimulation congruence*. As a consequence of theorem 2.3, we find that all structures $(G_{BPA}/\leftrightarrows_*,+,\cdot,Act)$ are well-defined algebras, every one of which may satisfy a different equational theory. In a slightly different setting, MILNER (1980) found that the algebra $(G_{BPA}/\leftrightarrows_{r\tau},+,\cdot,Act)$ can be completely axiomatized by BPA together with the following three equations:

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

Table 2. τ-laws ($a \in Act$).

THEOREM 2.5 BPA + T1-T3 *is a complete axiomatization of* $(G_{BPA}/\leftrightarrows_{r\tau},+,\cdot,Act)$.

In the setting of BPA and process graphs, this theorem was first established in BERGSTRA & KLOP (1985). Its proof will be given in section 4, together with the proofs of the theorems 2.6-2.8.

From figure 1 one can observe 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 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 originate from the fact that observation equivalence does not preserve branching structures. Since branching bisimulation equivalence distinguishes between all three graphs in figure 1, we expect that the laws T2 and T3 will no longer hold in $(G_{BPA}/\leftrightarrows_{rb},+,\cdot,Act)$. 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 derivable from T1 and T2 as one can check easily. Both axioms refer to the axiomatization of η, a constant for abstraction from BAETEN & VAN GLABBEEK (1987) similar to τ. In fact they are a variation on the first two η-laws in the sense that in BAETEN & VAN GLABBEEK (1987) 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.

THEOREM 2.6 BPA + H1-H2 *is a complete axiomatization of* $(G_{BPA}/\leftrightarrows_{rb},+,\cdot,Act)$.

Obviously, $\leftrightarrows_{r\eta}$ is a coarser notion than $\leftrightarrows_{rb}$ and it respects the axioms H1-H2. As it turns out we have the additional axiom H3 which was introduced earlier as T3.

$x\tau = x$	H1 (T1)
$x(\tau(y + z) + y) = x(y + z)$	H2
$a(\tau x + y) = a(\tau x + y) + ax$	H3 (T3)

Table 4. η-laws ($a \in Act$).

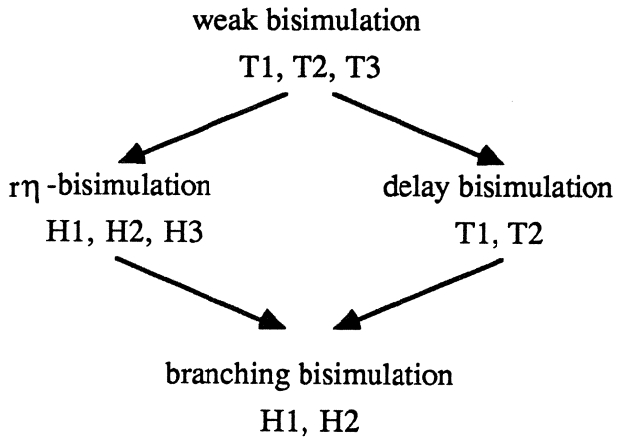
BAETEN & VAN GLABBEEK (1987) established a completeness theorem for rooted η-bisimulation:

THEOREM 2.7 BPA + H1-H3 *is a complete axiomatization of* $(G_{BPA}/\leftrightarrows_{r\eta},+,\cdot,Act)$.

So, on closed terms, the difference between H2 and T2 is precisely all the difference there is between the usual τ-laws and η. Finally a completeness theorem for delay bisimulation was (in the setting of CCS) established by WALKER (1990).

THEOREM 2.8 BPA + T1-T2 *is a complete axiomatization of* $(G_{BPA}/\leftrightarrows_{r\Delta},+,\cdot,Act)$.

Resuming we have the following diagram (see figure 7):



T1	$x\tau = x$	H1
	$x(\tau(y + z) + y) = x(y + z)$	H2
T2	$\tau x = \tau x + x$	
T3	$a(\tau x + y) = a(\tau x + y) + ax$	H3

FIGURE 7. Four notions of bisimulation with τ ($a \in Act$).

REMARK
In case we do not restrict to root unwound process graphs the definitions of the various bisimulations become a little more complicated. In particular the root conditions will have a different form (cf. BAETEN & VAN GLABBEEK (1987)) and the definition of the operator + on process graphs has to be changed.

2.2 CCS

In the setting of CCS we extend the graph domain G_{BPA} to G_{CCS} consisting of the root unwound process graphs, thus no longer excluding the trivial graph (the one-node graph without edges) nor any of the graphs with divergences (i.e. infinite τ-paths). We obtain: $G_{BPA} \subseteq G_{CCS} \subseteq G$.
We introduce a constant 0 for inaction, a binary infix written operator + for alternative composition, and unary operators a. for prefixing ($a \in Act$).

DEFINITION 2.3 The constant 0 and the operators + and a. are defined on G_{CCS} as follows:

(i) The constant 0 is interpreted as the trivial graph
(ii) (g + h) can be constructed by identifying the root nodes of g and h

(iii) (a.g) is constructed from g by adding a new node which will be the root of a.g, and a new a-labeled edge from the root of a.g to the root of g.

We will often leave out brackets, assuming that + will be the weakest operator symbol. For agents p we will often write ap instead of a.p in order to avoid non-essential distinctions between CCS and ACP. Similarly, we write Act for the set of prefix operators {a.: a∈ Act}. MILNER (1980) proved that the operators from Act and + all are well-defined on $G_{CCS}/\leftrightarrows$:

PROPOSITION 2.9 *Bisimulation equivalence is a congruence with respect to the operators from* Act *and* +.

So again, the structure $(G_{CCS}/\leftrightarrows,0,+,Act)$ is a well-defined algebra, and as in the case of $(G_{BPA}/\leftrightarrows,+,\cdot,Act)$ we can find a complete axiomatization of its equalities with respect to closed terms:

$x + y = y + x$	A1
$(x + y) + z = x + (y + z)$	A2
$x + x = x$	A3
$x + 0 = x$	A6

Table 5. Basic CCS.

Let us call the theory from table 5 *Basic CCS*, and write BCCS := A1-A3,A6. Then the following theorem is due to HENNESSY & MILNER (1980) and MILNER (1980).

THEOREM 2.10 *BCCS is a complete axiomatization of* $(G_{CCS}/\leftrightarrows,0,+,Act)$.

As before, we have four other equivalences $\leftrightarrows_{r*}$ for $* \in \{\tau,b,\eta,d\}$ on G_{CCS} which can be considered. First we establish that they are congruences.

THEOREM 2.11 *For* $* \in \{\tau,b,\eta,d\}$, $\leftrightarrows_{r*}$ *is a congruence on* G_{CCS} *with respect to* + *and* Act.

PROOF We prove it for $\leftrightarrows_{rb}$. The other proofs proceed in the same way.

The proof that $\leftrightarrows_{rb}$ is an equivalence and respects + can be copied from the proof of theorem 2.3. So it is left to prove that it respects the operators in Act. So suppose that R: g $\leftrightarrows_{rb}$ g' and p, p' are the root nodes of a.g and a.g'. Put R* := R∪{p,p'}. Then we prove R*: (a.g) $\leftrightarrows_{rb}$ (a.g').

(i) Clearly, the roots of both graphs are related by R*.

(ii) Assume that in (a.g) we have an edge $r \rightarrow^b r'$ and suppose we have R*(r,s) then from the construction of (a.g) it follows that either r=p or this edge originates from g.
If r=p then by the definition of R* we have s=p'. Furthermore, b=a and r' is the root node of g and by the construction of prefixing we find that in g' there exists an edge $s \rightarrow^a s'$ to the root node s' of g'. Since R is a branching bisimulation we find R(r',s') and hence R*(r',s').
If $r \rightarrow^b r'$ originates from g then it follows from the definition of R* that R(r,s), from which the requirement follows immediately.
(iii) Clearly, the root nodes are uniquely related by R*. □

It follows from theorem 2.4 that $\leftrightarrows_{r*}$ is moreover the coarsest BCCS-congruence contained in $\leftrightarrows_*$.
Now consider the axioms from the following table:

H1'	$a\tau x = ax$	T1'
H2'	$a(\tau(y + z) + y) = a(y + z)$	
	$\tau x = \tau x + x$	T2'
H3'	$a(\tau x + y) = a(\tau x + y) + ax$	T3'

Table 6. τ-laws in CCS (a∈ Act).

The only difference between these axioms and the ones introduced in the previous section is the replacement of sequential composition by prefixing in the axioms T1 (H1) and H2. The prime accents (') refer to this replacement. Note that H1' is derivable from H2. We find the following completeness results:

THEOREM 2.12
(i) BCCS *is a complete axiomatization of* $(G_{CCS}/\leftrightarrows,0,+,Act)$
(ii) BCCS + T1'-T3' *is a complete axiomatization of* $(G_{CCS}/\leftrightarrows_{r\tau},0,+,Act)$.
(iii) BCCS + H2' *is a complete axiomatization of* $(G_{CCS}/\leftrightarrows_{rb},0,+,Act)$.
(iv) BCCS + H2'-H3' *is a complete axiomatization of* $(G_{CCS}/\leftrightarrows_{r\eta},0,+,Act)$.
(v) BCCS + T1'-T2' *is a complete axiomatization of* $(G_{CCS}/\leftrightarrows_{rd},0,+,Act)$.

For the proof of theorem 2.12, we refer to section 4.

2.3. TERMINATION

In the previous two subsections, we presented two models: the model $(G_{BPA}/\leftrightarrows_{r*},+,\cdot,Act)$ for BPA with sequential composition, and $(G_{CCS}/\leftrightarrows_{r*},0,+,Act)$ for BCCS with prefixing. As we

argued before, including the trivial graph in $G_{BPA}/\leftrightarrows_{r*}$ would destroy the soundness of BPA in the corresponding model, i.e. of the axiom A4. Furthermore, from $G_{BPA}/\leftrightarrows_{r*}$ we have to exclude graphs containing infinite τ-paths since otherwise sequential composition no longer respects the equivalences - i.e. the equivalences $\leftrightarrows_{r*}$ are no longer congruences with respect to $\cdot$. For consider the following example:

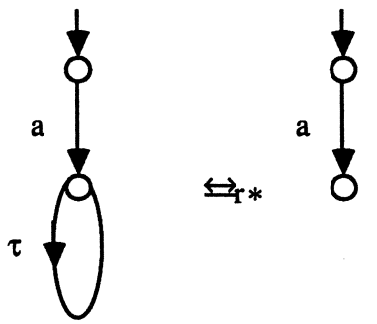


FIGURE 8. Equivalent graphs with and without divergence.

In figure 8, we find two equivalent graphs, one with and one without divergence, which we informally denote by $a\cdot\tau^{\omega}$ and a. So: $a\cdot\tau^{\omega} \leftrightarrows_{r*} a$, for $*\in\{\tau,b,\eta,d\}$. However, since $a\cdot\tau^{\omega}$ does not contain any endnodes we find that $(a\cdot\tau^{\omega})\cdot b = a\cdot\tau^{\omega}$ and $a\cdot\tau^{\omega} \not\leftrightarrows ab$. So in the presence of divergence $\leftrightarrows_{r*}$ no longer is a congruence with respect to $\cdot$.

The question arises whether the virtues of $(G_{BPA}/\leftrightarrows_{r*},+,\cdot,Act)$ and $(G_{CCS}/\leftrightarrows_{r*},0,+,Act)$ can be conbined, i.e. whether it is possible to define inaction and general sequential composition in one model (without destroying the intuitively plausable axiom A4) as well as to define general sequential composition on graphs with possible divergence paths, while respecting the equivalences. We will give a positive answer to this question by once again extending G_{CCS} to a larger domain G_{ACP} (so: $G_{BPA} \subseteq G_{CCS} \subseteq G_{ACP}$).

Let us extend the set Act with an additional label, written as $\surd$. Then, in G_{ACP} we will distinguish between successful and unsuccessful termination of a process by adding a *termination edge* to the endnodes which are considered to terminate successfully. Such termination edges consist of an outgoing edge labeled with $\surd$ to a new endnode. Let G_{ACP} consist of all graphs that can be obtained from non-trivial, root unwound graphs from G_{BPA} by adding termination edges to some of their endnodes. Next we add the trivial graph to G_{ACP} but assume that G_{ACP} is without the graph consisting of a single termination edge, i.e. the graph representing instant termination.

Observe that in graphs from G_{ACP} every node has at most one outgoing termination edge and if it has one, then it does not have any other outgoing edges. Furthermore, if a node has an incoming termination edge then it is an endnode and it does not have any other incoming edges. We immediately find that $G_{CCS} \subseteq G_{ACP}$ and $G_{ACP} \subseteq G^{\surd}$, where $G^{\surd}$ is the set of process graphs with

$\surd$ as a possible edge-label. The difference between G_{CCS} and G_{ACP} is that the latter distinguishes between two kinds of termination.

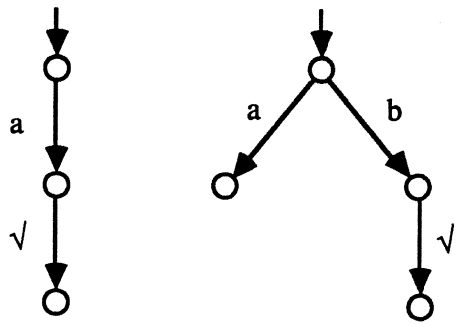


FIGURE 9. Process graphs with termination edges.

With respect to the algebraic operators, we simple combine the operators from BPA and ACP, but we adapt the definitions of action constants and sequential composition to the presence of $\surd$-labels. This is done in the following definition. The new operator for sequential composition will again be denoted by $\cdot$, and similarly for action constants. It will appear from the context (whether it is about G_{BPA} or G_{ACP}) which one of the definitions 2.1 and 2.4 presents their current interpretation. In case of doubt we underline the BPA operators.

DEFINITION 2.4 On G_{ACP} the constants 0 and a (for $a \in$ Act) and the operators + and $\cdot$ are defined as follows:

(i) 0 is the trivial graph
(ii) Constants $a \in$ Act are interpreted by the left hand side of figure 9
(iii) (g + h) can be constructed by identifying the root nodes of g and h
(iv) (g·h) is constructed by identifying every node in g with an outgoing termination edge with the root node of h while deleting its termination edge. The root node of (g·h) is that of g. If g is without termination edges, then (g·h) is just g.

The prefixing operator of CCS can now be defined by: a.g=a·g. In the subdomain G_{CCS} of G_{ACP} all processes end in deadlock (unsuccesful termination), so g·h=g. This explains the absence of sequential composition on G_{CCS}. Let G'_{BPA} be the subdomain of G_{ACP} consisting of all divergence free graphs from G_{ACP} only ending with succesful termination. Then $(G'_{BPA},+,\cdot,Act)$ and $(G_{BPA},+,\underline{\cdot},\underline{Act})$ are isomorphic algebras and the latter can be interpreted as a notational abbreviation of the former, where all $\surd$-labels have been left out.

On the new graph domain G_{ACP} we can define the bisimulation relations from definition 1.2-1.6 and 2.2, taking into account that $\surd \in$ Act. That is, termination edges are not treated anyhow different from other edges. The relations on G_{BPA}, inherited through the isomorphism from G'_{BPA}, coincide with the relations considered in subsection 2.1. However, this is no longer true if divergent graphs would be added to G_{BPA}; in that case all relations need an additional clause:

- If R(r,s) and r is an endnode than there is a path $s \Rightarrow s'$ to an endnode s'.

In order to prevent this complication in section 2.1, there we treated divergence free graphs only. The fact that definition 2.4 provides us with a proper algebraic structure on G_{ACP} follows from the following theorem:

THEOREM 2.13 *All equivalences* $\leftrightarrows$, $\leftrightarrows_{r\tau}$, $\leftrightarrows_{rb}$, $\leftrightarrows_{r\eta}$ *and* $\leftrightarrows_{rd}$ *are congruences with respect to the operators* + *and* · *on* G_{ACP}.

PROOF Again we prove the theorem for $\leftrightarrows_{rb}$. The fact that on G_{ACP} they are conguences with respect to + can be found from the proof of theorem 2.3. Considering the proof for ·, suppose that R: $g \leftrightarrows_{rb} g'$ and S: $h \leftrightarrows_{rb} h'$. Let R' be the restriction of R to the nodes in g that also appear in g·h (i.e. the nodes without incoming $\surd$-edges). We prove that R'$\cup$S: $(g \cdot h) \leftrightarrows_{rb} (g' \cdot h')$.

(i) Clearly the roots of (g·h) and (g'·h') are related by R'$\cup$S.

(ii) Assume that in (g·h) we have an edge $r \rightarrow^a r'$ and suppose (R'$\cup$S)(r,s), then from the construction of (g·h) it follows that this edge either originates from g or from h. If it is from g, then the proof proceeds as in the proof of theorem 2.3. So assume $r \rightarrow^a r'$ is from h.

- In case R'(r,s), we find that in g the node r has an outgoing termination edge $r \rightarrow^{\surd} r''$ to an endnode r'' (since those are the only nodes of g that are identified with nodes from h). Since R is a branching bisimulation, we find that in g' there exists a path $s \Rightarrow s' \rightarrow^{\surd} s''$ such that R(r,s') and R(r'',s''). By applying the definition of $\leftrightarrows_{rb}$ we even find that all nodes in $s \Rightarrow s'$ are related with r. Furthermore, by construction of (g'·h') the node s' is identified with the root node of h', and since S is a rooted branching bisimulation between h and h', we find:
either $a=\tau$ and S(r',s'), hence (R'$\cup$S)(r',s');
or there exists a path $s' \Rightarrow s_1 \rightarrow s_2 \Rightarrow s_3$ such that $S(r,s_1)$, $S(r',s_2)$ and $S(r',s_3)$ and hence $(R'\cup S)(r,s_1)$, $(R'\cup S)(r',s_2)$ and $(R'\cup S)(r',s_3)$ as required.
- In case *not* R'(r,s), then S(r,s) and both r and s are from h and h' respectively. Now the requirement follows immediately from the fact that S is a branching bisimulation.

(iii) In G_{ACP} the root node cannot have an outgoing termination edge, and hence the root nodes of (g·h) and (g'·h') are only related by R' (they are not identified with nodes from h or h'). Hence (R'$\cup$S) is rooted since R is. □

As a consequence, we find a well-defined algebra ($G_{ACP}/\leftrightarrows$,0,+,·,Act), and four others with domain $G_{ACP}/\leftrightarrows_{r*}$ ($* \in \{\tau,b,\eta,d\}$). To start with, we find that the following basic theory is valid in all five algebras (see table 7):

$x + y = y + x$	A1
$(x + y) + z = x + (y + z)$	A2
$x + x = x$	A3
$(x + y)z = xz + yz$	A4
$(xy)z = x(yz)$	A5
$x + 0 = x$	A6
$0 \cdot x = 0$	A7

Table 7. BPA_0.

The theory BPA_0 is the kernel of the axiom system ACP, introduced in BERGSTRA & KLOP (1984), where 0 was called δ. As before, we have the following completeness theorem for the five respective algebras:

THEOREM 2.14

(i) BPA_0 *is a complete axiomatization of* $(G_{ACP}/\leftrightarrows,0,+,\cdot,Act)$

(ii) BPA_0 + T1-T3 *is a complete axiomatization of* $(G_{ACP}/\leftrightarrows_{r\tau},0,+,\cdot,Act)$

(iii) BPA_0 + H1-H2 *is a complete axiomatization of* $(G_{ACP}/\leftrightarrows_{rb},0,+,\cdot,Act)$

(iv) BPA_0 + H1-H3 *is a complete axiomatization of* $(G_{ACP}/\leftrightarrows_{r\eta},0,+,\cdot,Act)$

(v) BPA_0 + T1-T2 *is a complete axiomatization of* $(G_{ACP}/\leftrightarrows_{rd},0,+,\cdot,Act)$.

Again for the proof of this completeness theorem we refer to section 4.

3. 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 in places different from any in 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 how branching bisimulation preserves the branching structure of graphs. Let us first consider ordinary bisimulation.

DEFINITION 3.1 A *concrete trace* of a process graph is a finite sequence $(a_1, a_2, a_3, \ldots, a_k)$ of actions from Act, 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 *concrete trace equivalent*, notation $g \equiv_t h$, if their *concrete trace sets* (i.e. the sets of their concrete traces) are equal. It is easily checked that $\equiv_t$ is a congruence on G_{BPA} and $g \leftrightarrows h \Rightarrow g \equiv_t h$. Consequently we find that $G_{BPA}/\equiv_t$ is a model for BPA. Compared to bisimulation, concrete trace equivalence makes much more identifications. For example, we find that $G_{BPA}/\equiv_t$ satisfies the equation $x(y + z) = xy + xz$ which cannot be proved from BPA.
The main reason for this is that a concrete trace does not provide us with information about the branching 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 3.2 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 3.3 A *concrete coloured trace* of a coloured graph g is a sequence of the form $(C_0, a_1, C_1, a_2, C_2, \ldots, a_k, C_k)$ for which there exists 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 concrete coloured traces of a node r in a graph g are the concrete coloured traces of the subgraph $(g)_r$ 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 differences of the nodes, or - to put it differently - how to define the concept of 'branching potential in a node' properly. There are several ways to do this. Probably the shortest definition is the following:

DEFINITION 3.4 A *concrete consistent colouring* of 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 concrete 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 and h are *concrete coloured trace equivalent* - notation: $g \equiv_{cc} h$ - if for some concrete consistent colouring on $\{g,h\}$ they have the same concrete coloured trace set, or equivalently, if for some concrete consistent colouring on $\{g,h\}$ the root nodes have the same colour. Then we have the following important characterization:

THEOREM 3.1 $g \leftrightarrows h$ *if and only if* $g \equiv_{cc} h$.

PROOF $\Rightarrow$: Suppose R is the largest bisimulation relation between g and h. Let $\underline{R}$ be the transitive closure of R, then $\underline{R}$ is an equivalence relation on the set of nodes from g and h. Let C be the set of equivalence classes induced by $\underline{R}$ and label every node with its own equivalence class. Then this colouring is consistent on g and h.
To see this let r_0 be a node in g say, and $(C_0, a_1, C_1, a_2, C_2, \ldots, a_k, C_k)$ be a concrete coloured trace which corresponds to a path $r_0 \rightarrow^{a_1} r_1 \rightarrow^{a_2} r_2 \rightarrow \cdots \rightarrow^{a_k} r_k$ starting from r_0. Now suppose for some node s_0 in h we have $R(r_0,s_0)$, then we find from definition 1.2 that $s_0 \rightarrow^{a_1} s_1$ for some s_1 such that $R(r_1,s_1)$. Thus r_1 and s_1 have the same colour C_1. By induction we find that s_0 has the same concrete coloured trace $(C_0, a_1, C_1, a_2, C_2, \ldots, a_k, C_k)$. So R preserves concrete coloured trace sets, hence so does $\underline{R}$.
Since the roots of g and h are related we find $g \equiv_{cc} h$.
$\Leftarrow$: Suppose that g and h have the same concrete coloured trace sets. Then consider the relation R which relates two nodes of g and h iff they are labeled with the same colour. It is easy to prove that R is a bisimulation between g and h. □

So far we did not have any notion of abstraction in the definition of coloured traces, so if a coloured graph has τ-labels then these are treated as if they were ordinary actions. In the following definition we find how to abstract from these τ-steps. The idea is simple: τ-steps can only be left out if they are *inert*, meaning that they are between two nodes that have the same colour. Thus it is not only that inert steps are not observable, but even more, they do not cause any change in the overall state of the machine.

DEFINITION 3.5 A *coloured trace* of a coloured graph 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 of this graph by replacing all subsequences $(C, \tau, C, \tau, \ldots, \tau, C)$ by C.

DEFINITION 3.6 A *consistent colouring* of 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. Furthermore such a colouring is *rooted* if no root-node has the same colour as a non-root node.

For two root unwound graphs g and h let us 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 3.2

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

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

PROOF $\Rightarrow$: Suppose R is the largest (rooted) branching bisimulation between g and h. Let $\underline{R}$ be its transitive closure and C the set of equivalence classes induced by $\underline{R}$. Then the colouring in which every node is labeled with its own equivalence class is consistent (and rooted) on g and h. To see this, let us write C(r) for the colour of the node r and assume that, for certain nodes r_0 and s_0, $R(r_0,s_0)$ and r_0 has an coloured trace $(C_0, a_1, C_1, a_2, C_2,..., a_k, C_k)$. Then there exists a path of the form $r_0 \rightarrow^\tau u_1 \rightarrow^\tau ... \rightarrow^\tau u_m \rightarrow^{a_1} r_1$ $(m \geq 0)$ such that $C(r_1) = C_1$ and for all i: $C(u_i) = C(r_0) = C_0$. For every edge $u_i \rightarrow^\tau u_{i+1}$ $(0 \leq i < m, u_0 = r_0)$ there exists a path $v_i \Rightarrow v_{i+1}$ $(v_0 = s_0)$ such that $R(u_i,v_i)$, and all intermediate nodes are related to either u_i or u_{i+1} (by lemma 1.1), hence all v_i have the same colour C_0. So we find a path $s_0 \Rightarrow v_m$ with only one colour in the nodes such that $R(u_m,v_m)$.

Next, since $u_m \rightarrow^{a_1} r_1$ and $R(u_m,v_m)$ we find that either $a_1 = \tau$ and $R(r_1,v_m)$ - in which case $C_1 = C_0$ in contradiction with $(C_0, a_1, C_1, a_2, C_2,..., a_k, C_k)$ being a coloured trace - or there is a path $v_m \Rightarrow t_1 \rightarrow^{a_1} s_1$ such that $R(u_m,t_1)$ and $R(r_1,s_1)$. Again by lemma 1.1 we find that t_1 and all the intermediate nodes in $\Rightarrow$ have the same colour as v_m and so we find a coloured trace (C_0, a_1, C_1) of s_0. By repeating this argument k times, we find that s_0 has a coloured trace $(C_0, a_1, C_1, a_2, C_2,..., a_k, C_k)$ and so R preserves coloured trace sets. Thus $\underline{R}$ induces a consistent colouring and since the roots are related we find $g \equiv_c h$. If moreover R is rooted, then so is the induced colouring.

$\Leftarrow$: Consider a (rooted) consistent colouring such that the coloured trace sets of g and h are equal with respect to that colouring. Let R be the relation between nodes of g and h relating two nodes iff they have the same colour, then it is easy to see that R is a (rooted) branching bisimulation. □

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 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 states that are passed through.

Another way of looking at the colouring of a graph is the following. Since trace-equivalence is too weak to characterize branching bisimilarity we can add more information to traces in order to distinguish between processes. Consider the following definition:

DEFINITION 3.7 For ordinals α the *α-trace set* of a graph g is defined as follows:

1. The α-trace set of a node r of g is the set of all γ-traces of r, for $\gamma < \alpha$.
2. An α-trace of r is made of a sequence $(T_0, a_1, T_1, a_2,..., a_k, T_k)$, where a_i are actions from Act and T_i are α-trace sets such that g has a path $r_0 \rightarrow^{a_1} r_1 \rightarrow^{a_2} \cdots \rightarrow^{a_k} r_k$ and r_i has α-trace set T_i, by replacing all subsequences $(T, \tau, T, \tau, ..., \tau, T)$ by T.
3. The α-trace set of g is the α-trace set of its root.

Note that all 0-trace sets are empty, and the 1-trace set of g is just the set of its concrete traces from which τ's have been left out. Two graphs g and h are α-*trace equivalent* - notation $g \approx_\alpha h$ - if they have the same α-trace set. Let us say that they are *hypertrace equivalent* - notation $g \approx h$ - if $g \approx_\alpha h$ for all ordinals α. Note that if $\lambda<\alpha$ then $g \approx_\alpha h$ implies $g \approx_\lambda h$. From this it immediately follows that if $G' \subseteq G$ is a *set* of process graphs then on G' the notion of α-trace equivalence stabelizes for some ordinal - i.e. there exists a *closure ordinal* α such that, for $g,h \in G'$, $g \approx h$ iff $g \approx_\alpha h$. It will follow from the proof of theorem 3.3 that the smallest ordinal with $(g \approx_\alpha h \Leftrightarrow g \approx_{\alpha+1} h)$ is a closure ordinal. Furthermore if G has cardinality β then β must be a closure ordinal. Next we prove that hypertrace equivalence coincides with coloured trace equivalence:

THEOREM 3.3 $g \approx h$ *if and only if* $g \equiv_c h$.

PROOF $\Rightarrow$: Let G' be a set of process graphs containing g,h and all their subgraphs and let α be the smallest ordinal such that, for $g',h' \in G'$, $g' \approx_\alpha h'$ iff $g' \approx_{\alpha+1} h'$. If $g' \approx_{\alpha+1} h'$ then from definition 3.7 we find that g' and h' have the same α-traces. Consider the colouring on g and h in which every node is coloured with its own α-trace set. Now a coloured trace $(C_0, a_1, C_1, a_2,..., a_k, C_k)$ of a node r is precisely an α-trace and by definition of α we have that r and r' have the same α-trace set only if they have the same α-traces, i.e. they have the same colour only if they have the same coloured traces. Hence the colouring is consistent.
Now $g \approx h \Rightarrow g \approx_{\alpha+1} h \Rightarrow$ g and h have the same coloured traces $\Rightarrow g \equiv_c h$.
$\Leftarrow$: Take a consistent colouring on g and h such that the roots of g and h have the same colour. Then with transfinite induction on γ it is easy to prove that equally coloured nodes have the same γ-traces for all ordinals γ. □

Hence we find that $\approx$ is equivalent to $\equiv_c$, and hence to $\leftrightarrows_b$ (theorem 3.2). Note that compared to *readiness semantics* (cf. OLDEROG & HOARE (1986)), *possible-futures semantics* (cf. ROUNDS & BROOKES (1981)) and *ready trace semantics* (cf. BAETEN, BERGSTRA & KLOP (1987b)) in an α-trace ($\alpha \geq 1$) we keep track of a lot more information. Apart from just all one-step exits from the endstate of a partial execution we are now able to see all traces (and higher traces) that can be chosen at every intermediate state during the execution.
The notion of hypertrace equivalence gives us an indication of the amount of extra information that is needed to turn trace equivalence into branching bisimulation equivalence. Furthermore, it provides us with an idea of how to build a consistent colouring on a set of graphs by distinguishing more and more between nodes. A construction similar to definition 3.7 was used by MILNER (1985) to characterize observation equivalence in the spirit of definition 1.7.

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

PROPOSITION 3.4 *It is possible to colour the nodes of a root unwound process graph g in such a way that two nodes have the same colour iff they can be related by a rooted branching autobisimulation on g (relating* g *with itself). This colouring is rooted and consistent.*

PROOF For every root unwound process graph g the largest rooted branching autobisimulation on g is an equivalence relation on the nodes. It follows from the proof of theorem 3.2 that every node can be labeled with its equivalence class as a colour, in order to obtain a rooted consistent colouring. □

This colouring of a graph is called its *canonical colouring*. Note that two nodes r and s of a root unwound process graph g have the same colour with respect to its canonical colouring if and only if r,s ≠ root(g) and $(g)_r \leftrightarrows_b (g)_s$ (the subgraph $(g)_r$ of g with root r is defined in the beginning of this section; furthermore, remember that in the canonical colouring root nodes have colours different from those of internal nodes). We say that r and s are *rooted branching bisimilar*. A root unwound graph is said to be in *normal form* if each node has a different colour with respect to its canonical colouring and it has no τ-loops $r \rightarrow^{\tau} r$. Next we show that each root unwound process graph is rooted branching bisimilar with exactly one normal form (up to isomorphism).

DEFINITION 3.8 Let g be a root unwound 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 and removing τ-loops. To be precise:

1. N(g) has colours C∈**C** as its nodes.
2. N(g) has an edge $C \rightarrow^{a} C'$ ($a \neq \tau$) 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≠C' and g has an edge $r \rightarrow^{\tau} r'$ with C(r)=C and C(r')=C'.

PROPOSITION 3.5 *For all root unwound process graphs* g: $g \leftrightarrows_{rb} N(g)$.

PROOF Consider the canonical colouring on g, and the trivial colouring on N(g) in which each node (being a colour from **C**) is labeled by itself. Let R be the relation relating nodes from g and N(g) iff they have the same colour. Now it follows directly from the construction above that R is a rooted branching bisimulation between g and N(g). □

So in every rooted branching bisimulation equivalence class of root unwound process graphs there is a normal form. We proceed by showing that up to isomorphism there is only one.

DEFINITION 3.9 A *graph isomorphism* is a bijective relation R between the nodes of two process graphs 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 $r \to^a r'$ is an edge in g iff $s \to^a s'$ is an edge in h ($a \in Act$).

Note that a graph isomorphism is just a bijective bisimulation, or a bijective branching bisimulation for that matter. 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 3.6 (normal form theorem) *Let* g *and* h *be root unwound graphs that are in normal form. Then* $g \leftrightarrows_{rb} h$ *if and only if* $g \cong h$.

PROOF This follows since any bisimulation R: $g \leftrightarrows_{rb} h$ must be bijective:
(i) it is surjective because every node in g or h can be reached from the root; hence by definition 1.4 every node is related to some node in the other graph;
(ii) it is injective since every node is related with at most one other node: if two different nodes in g are related to the same node in h, then these two are also related by a branching autobisimulation on g, and so with respect to the canonical colouring they have the same colour. But then by definition 3.8 the nodes are identical, which is a contradiction. □

Theorem 3.6 says that each equivalence class in $G/\leftrightarrows_{rb}$ can be represented by one special element: its normal form. It follows that $g \leftrightarrows_{rb} h$ if and only if $N(g) \cong N(h)$.

4. COMPLETENESS PROOFS

In this section we will present the proofs of the completeness theorems 2.6, 2.7, 2.8 and 2.5. By means of a rather trivial adaptation of the contents of this section one obtains the completeness theorems for CCS and ACP_τ (theorems 2.12 and 2.14). The basic idea in these proofs is to establish a *graph rewriting system* on finite process graphs, which is confluent and terminating. We prove that (i) normal forms with respect to the graph rewriting system are normal forms in the sense of definition 3.8, hence two normal forms are bisimilar iff they are equal (i.e. isomorphic). Furthermore we prove that every rewriting step in the system (ii) preserves bisimulation, and (iii) corresponds to a proof step in the theory. Then we conclude:
- two finite graphs are bisimilar iff they have the same normal form
- if two graphs have the same normal form then the corresponding terms can be proved equal.
To start with, let us consider some definitions.

DEFINITION 4.1 Let $\boldsymbol{H} \subseteq \boldsymbol{G}$ be the set of *finite* process graphs and $\boldsymbol{H}^+ \subseteq \boldsymbol{G}_{BPA}$ the set of finite, non-trivial process graphs. Here a process graph is *finite* if it has only finitely many paths.

Note that finite process graphs are acyclic and thus certainly root-unwound, and contain only finitely many nodes and edges. Later on, we will establish a correspondence between graphs from H^+ and closed terms in BPA_τ, i.e. the signature of BPA together with the extra constant τ. Below we will use the results from the previous section, starting from proposition 3.4.

DEFINITION 4.2 A pair (r,s) of nodes in a process graph g is called a pair of *double nodes* if $r \neq s$, $r,s \neq root(g)$ and for all nodes t and labels $a \in Act$: $r \rightarrow^a t \Leftrightarrow s \rightarrow^a t$.

DEFINITION 4.3 An edge $r \rightarrow^\tau s$ in a process graph g is called *manifestly inert* if $r \neq root(g)$ and for all nodes t and labels $a \in Act$ such that $(a,t) \neq (\tau,s)$: $r \rightarrow^a t \Rightarrow s \rightarrow^a t$.

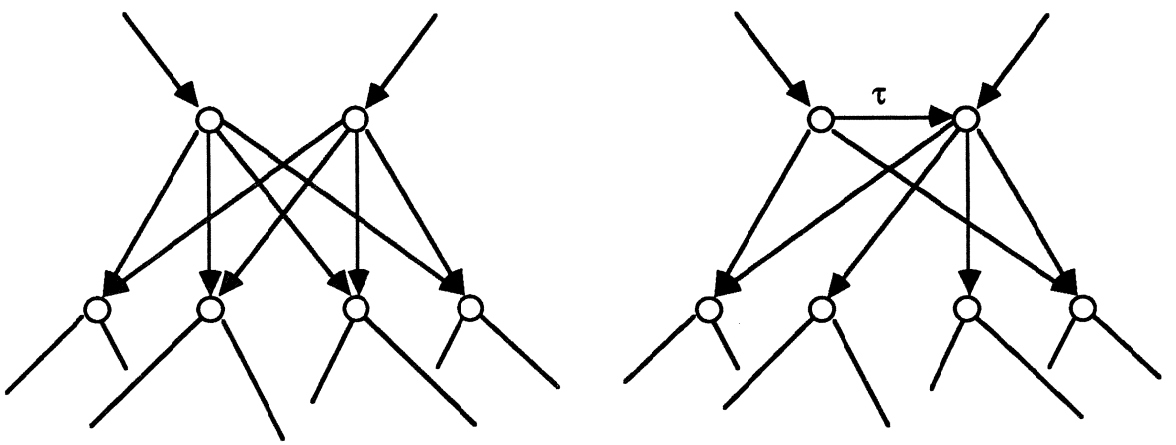


Figure 10. A pair of double nodes (left) and a manifestly inert τ-step.

Note that for finite process graphs g, the requirement $r,s \neq root(g)$ in definition 4.2 is redundant. A τ-edge in a root unwound graph g is *inert* if it is between two rooted branching bisimilar nodes (i.e. nodes that have the same colour in the canonical colouring of g). For root unwound graphs it is easily checked that if (r,s) is a pair of double nodes or if $r \rightarrow^\tau s$ is manifestly inert, then r and s are rooted branching bisimilar. As one can see from figure 10, it is essential in the definitions 4.2 and 4.3 that this can be found by investigating the outgoing edges only up to one level. For this reason, in definition 4.3 the τ-step is called *manifestly* inert, since it can be recognized as such. On H, sharing of double nodes and contraction of manifestly inert τ-steps turns out to be strong enough to reduce a graph to its normal form. This means that in the reduction process all rooted branching bisimilar nodes become *manifestly* rooted branching bisimilar.

THEOREM 4.1 *A graph* $g \in \boldsymbol{H}$ *without double nodes or manifestly inert edges is in normal form.*

PROOF Let $g \in \boldsymbol{H}$ be a finite graph which is not in normal form. Then with respect to its canonical colouring (proposition 3.4) it has at least one pair of different nodes with the same colour. Now define the *depth* d(s) of a node s as the number of edges in the longest path starting from s, and the *combined depth* of two nodes as the sum of their depths. Choose a pair (r,s) of different equally coloured nodes in g with minimal combined depth. Consequently we have:

(*) if r' and s' have the same colour and $d(r') + d(s') < d(r) + d(s)$ then r'=s'.

Without loss of generality assume $d(s) \leq d(r)$. Then we prove the following two statements:

1. *if* $r \rightarrow^a t$ ($a \in Act$) *is an edge in* g *and* $(a,t) \neq (\tau,s)$, *then* $s \rightarrow^a t$ *is an edge in* g

2. *if* $s \rightarrow^a t$ ($a \in Act$) *is an edge in* g, *then either* $r \rightarrow^\tau s$ *or* $r \rightarrow^a t$ *is an edge in* g.

From these two statements we find that if $r \rightarrow^\tau s$ is an edge in g then it is manisfestly inert, and if $r \rightarrow^\tau s$ is not an edge in g, then (r,s) is a pair of double nodes, which proves our theorem. Note that since r and s are different equally coloured nodes, they both must be different from the root.

ad 1: Let $r \rightarrow^a t$ be an edge in g and $(a,t) \neq (\tau,s)$. Since r and s have the same colour (hence the same coloured traces) we find that either $a=\tau$ and t has the same colour as r and s, or s has the coloured trace (C(r), a, C(t)). In the first case it follows from $d(t) < d(r)$ and (*) that t=s, which is in contradiction with our assumption $(a,t) \neq (\tau,s)$. So s has a coloured trace (C(r), a, C(t)). Suppose that $s \rightarrow^\tau u$ for a node u with colour C(u)=C(s)=C(r), then it follows from $d(u)<d(s)$ and (*) that u=r, contradicting $d(u)<d(s) \leq d(r)$. Hence there is a node u such that $s \rightarrow^a u$ and C(t)=C(u), and since $d(t) + d(u) < d(r) + d(s)$ we conclude from (*) that t=u. Hence $s \rightarrow^a t$ is an edge in g.

ad 2: Let $s \rightarrow^a t$ be an edge in g. If C(t)=C(s)=C(r) then it follows from (*) and $d(t)<d(s)$ that r=t, in contradiction with $d(t)<d(s) \leq d(r)$. So (C(s), a, C(t)) is a coloured trace of s, and since r and s have the same colour (C(s), a, C(t)) is a coloured trace of r as well. Now if r has an outgoing τ-edge $r \rightarrow^\tau u$ to a node with the same colour C(r), then it follows from $d(u) + d(s) < d(r) + d(s)$ and (*) that u=s. If r has no such edge, then it has an edge $r \rightarrow^a u$ with C(u)=C(t), and since $d(u) + d(t) < d(r) + d(s)$ we find that u=t. Thus we proved that either $r \rightarrow^\tau s$ or $r \rightarrow^a t$, which proves (2). □

Theorem 4.1 tells us that all we need do in order to turn a finite graph g into its normal form is to repeatedly unify its pairs of double nodes and contract its manifestly inert edges. In the case of finite graphs this can be done in finitely many steps as follows:

DEFINITION 4.4 For any graph $g \in \boldsymbol{H}$ the rewriting relation $\rightarrow_{\boldsymbol{H}}$ is defined by the following two one-step reductions:

1. *sharing* a pair of double nodes (r,s): replace all edges $t \rightarrow^a r$ by $t \rightarrow^a s$ (if not already there, otherwise just remove $t \rightarrow^a r$) and remove r together with all its outgoing edges from g;

2. *contracting* a manifestly inert step $r \to^\tau s$: replace all edges $t \to^a r$ by $t \to^a s$ (if not already there, otherwise just remove $t \to^a r$) and remove r together with all its outgoing edges from g.

PROPOSITION 4.2 *The rewriting relation $\to_H$ has the following properties:*

i. ***H*** *as well as* H^+ *are closed under applications of* $\to_H$

ii. if $g \to_H h$ *then* $g \leftrightarrows_{rb} h$

iii. $\to_H$ *is confluent and terminating.*

PROOF (i) In applications of $\to_H$ the root is never removed and in the resulting graph all nodes remain accessible from the root. Never two edges with the same label appear between the same two nodes. The graph also remains finite (and non-trivial).

(ii) Suppose (r,s) is a pair of double nodes or $r \to^\tau s$ is a manifestly inert edge in g, and $g \to_H h$ identifies the nodes r and s (= removes the node r). Let I be the identity relation on the nodes of h then $I \cup \{\{r,s\}\}$ is a rooted branching bisimulation between g and h. This is easy to prove from the definitions 4.2 and 4.3.

(iii) $\to_H$ is terminating since it decreases the number of nodes, and every finite process graph has finitely many nodes. Next, suppose g has two normal forms n and n', then by the definition of $\to_H$ n and n' are without pairs of double nodes and without manisfestly inert edges. Thus by theorem 4.1 n and n' are in normal form. By (ii) it follows that $n \leftrightarrows_{rb} n'$ and hence by theorem 3.6 (normal form theorem) we have $n \equiv n'$. □

Next we will establish a correspondence between finite non-trivial graphs and closed BPA_τ-terms, such that the graph reductions of definition 4.4 correspond to proof steps in BPA + H1,2.
Write $s \equiv_\Gamma t$ for $\Gamma \vdash s=t$ saying that s and t are equal *modulo* applications of axioms from Γ and the standard axioms for equality (reflexivity, commutativity, transitivity and replacement). It is quite easy to turn finite non-trivial graphs into BPA_τ-terms as follows. Let $T(BPA_\tau)$ be the set of closed BPA_τ terms.

DEFINITION 4.5 Let $\langle\cdot\rangle: H^+ \to T(BPA_\tau)$ be a mapping that satisfies

$$\langle g \rangle = \sum_{\substack{r(g) \to^a s \text{ is an edge in g;} \\ s \text{ not an endnode}}} a \cdot \langle (g)_s \rangle + \sum_{\substack{r(g) \to^b s \text{ is an edge in g;} \\ s \text{ is an endnode}}} b.$$

Here r(g) denotes the root node of $g \in H^+$ and if p_i is a BPA_τ-term for $i \in I$, with $I=\{i_1,...,i_n\}$ a finite non-empty set of indices, then $\sum_{i \in I} p_i$ denotes a BPA_τ-term $p_{i_1} + ... + p_{i_n}$. Note that the notation $\sum_{i \in I} p_i$ does not determine the order and association of the terms p_i.

If $g \in \boldsymbol{H}^+$, $r(g) \rightarrow^a s$ is an edge in g, and s is not an endnode, then $(g)_s \in \boldsymbol{H}^+$. Furthermore, since $g \in \boldsymbol{H}^+$ is finite, r(g) has only finitely many outgoing edges. Moreover, with induction to the depth of its arguments, one easily proves that $\langle\cdot\rangle$ is well-defined. Observe that for $g \in \boldsymbol{H}^+$, $\langle g \rangle$ is only *uniquely* defined modulo A1-A2.

PROPOSITION 4.3 *If* $g,h \in \boldsymbol{H}^+$ *and* $g \leftrightarrows h$, *then* A1-A2 $\vdash \langle g \rangle = \langle h \rangle$.

PROOF Trivial.

DEFINITION 4.6 The denotation $[\![p]\!]$ of a BPA_τ-term p in the graph domain $\boldsymbol{G}$, is defined by:

$[\![a]\!] = a_{\boldsymbol{G}}$ for $a \in A_\tau$

$[\![x + y]\!] = [\![x]\!] +_{\boldsymbol{G}} [\![y]\!]$

$[\![x \cdot y]\!] = [\![x]\!] \cdot_{\boldsymbol{G}} [\![y]\!]$

where $a_{\boldsymbol{G}}$, $+_{\boldsymbol{G}}$ and $\cdot_{\boldsymbol{G}}$ are the interpretations in $\boldsymbol{G}$, of the constants and operators a, + and · of BPA_τ, as defined in definition 2.1.

Now it turns out that for terms of the form $\langle g \rangle$ (for $g \in \boldsymbol{H}^+$) $\langle\cdot\rangle$ is a left-inverse of $[\![\cdot]\!]$, modulo A1-A2. Consider the following definition:

DEFINITION 4.7 The set BT of *basic terms* over BPA_τ is inductively defined as follows:

1. For all $a \in$ Act we have $a \in$ BT;
2. If $p,q \in$ BT then $(p + q) \in$ BT and for all $a \in$ Act: $a \cdot p \in$ BT.

LEMMA 4.4 *For* $g \in \boldsymbol{H}^+$, $\langle g \rangle$ is a basic term and if $p \in$ BT, then $\langle [\![p]\!] \rangle \equiv_{A1,2} p$.

PROOF With induction to the structure of terms:

- If p = a ($a \in$ Act) then $[\![p]\!]$ is the one-edge graph labeled with a, and so $\langle [\![p]\!] \rangle = p$.
- If $p = a \cdot u$ for some basic term u, then $[\![p]\!]$ is the graph with an edge labeled a followed by $[\![u]\!]$. Then, $\langle [\![p]\!] \rangle = a \cdot \langle [\![u]\!] \rangle$ and so by induction we find that $\langle [\![p]\!] \rangle \equiv_{A1,2} a \cdot u$.
- Suppose $p = u + v$. One can easily see that for graphs g and h: $\langle g +_{\boldsymbol{G}} h \rangle \equiv_{A1,2} \langle g \rangle + \langle h \rangle$. Then: A1-A2 $\vdash \langle [\![u + v]\!] \rangle = \langle [\![u]\!] \rangle + \langle [\![v]\!] \rangle = u + v$ (by induction). □

LEMMA 4.5 (elimination)

For every closed BPA_τ*-term* p *there exists a basic term* q *such that* A4-A5 $\vdash p = q$.

PROOF By induction on the structure of p.

- If p = a ($a \in$ Act) then p is a basic term.
- If $p = u \cdot v$ and lemma 4.5 can be proved for all terms smaller than p, then there exist basic terms

u' and v' such that A4-A5 $\vDash$ u = u', v = v'. Now suppose u' has the form $(\Sigma_i a_i \cdot w_i + \Sigma_j b_j)$, then we find:

$$\begin{aligned} \text{A4-A5} \vDash p = u' \cdot v' &= (\Sigma_i a_i \cdot w_i + \Sigma_j b_j) \cdot v' = \Sigma_i (a_i \cdot w_i) \cdot v' + \Sigma_j b_j \cdot v' \text{ (by axiom A4)} \\ &= \Sigma_i a_i \cdot (w_i \cdot v') + \Sigma_j b_j \cdot v' \text{ (by axiom A5)} \\ &= \Sigma_i a_i \cdot q_i + \Sigma_j b_j \cdot v' \text{ for some basic terms } q_i \text{ (by induction)} \end{aligned}$$

which is a basic term again.

- If p = u + v then A4-A5 $\vDash$ p = u' + v' for basic terms u' and v', and the sum of two basic terms is again a basic term. □

PROPOSITION 4.6 *For all closed* BPA_τ*-terms* p *we have:* A1-A2 + A4-A5 $\vdash$ $\langle [p] \rangle = p$.

PROOF If 'p=q' is an instantiation of A4 or A5 (possibly in a context) then $\langle [p] \rangle \equiv_{A1,2} \langle [q] \rangle$. Now the proposition follows immediately from the lemma s 4.4 and 4.5. □

This concludes the establishment of a correspondence between $\boldsymbol{H}^+$ and $T(BPA_\tau)$. Next we will show that every rewriting step on $\boldsymbol{H}^+$ corresponds to a proof step in BPA + H1-H2.

LEMMA 4.7 *Let* (r,s) *be a pair of dubble nodes or* $r \rightarrow^\tau s$ *be a manifestly inert* τ*-step in a process graph* g, *such that neither r nor s are endnodes, and let* $a \in$ Act. *Then we have:* BPA + H1-H2 $\vdash$ $a \cdot \langle (g)_r \rangle = a \cdot \langle (g)_s \rangle$.

PROOF In case (r,s) is a pair of dubble nodes r has an edge $r \rightarrow^a t$ iff s has an edge $s \rightarrow^a t$ and so $\langle (g)_r \rangle \equiv_{A1,2} \langle (g)_s \rangle$, hence $a \cdot \langle (g)_r \rangle = a \cdot \langle (g)_s \rangle$.

In case $r \rightarrow^\tau s$ is a manifestly inert τ-step we distinguish two subcases: First assume that r has more outgoing edges than only $r \rightarrow^\tau s$. Then there must be basic terms p and q such that

(1) $\langle (g)_r \rangle \equiv_{A1,2} \tau \cdot \langle (g)_s \rangle + p$

(2) $\langle (g)_s \rangle \equiv_{A1,2} p + q$.

So we derive:

$$\begin{aligned} A1,2 + H2 \vdash a \cdot \langle (g)_r \rangle &= a \cdot (\tau \cdot \langle (g)_s \rangle + p) \text{ (by (1))} = \\ &= a \cdot (\tau \cdot (p + q) + p) \text{ (by (2))} = \\ &= a \cdot (p + q) \text{ (by applying H2)} \\ &= a \cdot \langle (g)_s \rangle \text{ (by (2))}. \end{aligned}$$

In case r has no more outgoing edges than $r \rightarrow^\tau s$ we have $\langle (g)_r \rangle = \tau \cdot \langle (g)_s \rangle$, hence

$$A5 + H1 \vdash a \cdot \langle (g)_r \rangle = a \cdot (\tau \cdot \langle (g)_s \rangle) = (a \cdot \tau) \cdot \langle (g)_s \rangle = a \cdot \langle (g)_s \rangle.$$ □

PROPOSITION 4.8 *If* $g \rightarrow_{\boldsymbol{H}} h$ *then* BPA + H1-H2 $\vdash \langle g \rangle = \langle h \rangle$.

PROOF On H the rewriting relation $\rightarrow_H$ can be decomposed in the following elementary reductions: Take a pair of double nodes (r,s) or a manifestly inert τ-step $r \rightarrow^\tau s$ and replace one edge $t \rightarrow^a r$ by $t \rightarrow^a s$ (if not already there, otherwise just remove $t \rightarrow^a r$) and if r has no more incoming edges remove r together with all its outgoing edges from g. So it suffices to proof that if h is obtained from g by means of such an elementary reduction, we have $\langle g \rangle \equiv_\Gamma \langle h \rangle$, where Γ = BPA + H1-H2. From definition 4.5 it follows that it even suffices to proof $\langle (g)_t \rangle \equiv_\Gamma \langle (h)_t \rangle$.

- First consider the case that neither r nor s are endnodes and there is no edge $t \rightarrow^a s$ in g. Then $\langle (g)_t \rangle \equiv_{A1,2} a\cdot\langle (g)_r \rangle + p$ for certain basic term p. Lemma 4.7 says $a\cdot\langle (g)_r \rangle \equiv_\Gamma a\cdot\langle (g)_s \rangle$, hence $\langle (g)_t \rangle \equiv_\Gamma a\cdot\langle (g)_s \rangle + p \equiv_{A1,2} \langle (h)_t \rangle$.
- In case $t \rightarrow^a s$ is an edge in g, and r,s are still assumed not to be endnodes we have $\langle (g)_t \rangle \equiv_{A1,2} a\cdot\langle (g)_r \rangle + a\cdot\langle (g)_s \rangle + p \equiv_\Gamma a\cdot\langle (g)_s \rangle + a\cdot\langle (g)_s \rangle + p \equiv_{A2,3} a\cdot\langle (g)_s \rangle + p \equiv_{A1,2} \langle (h)_t \rangle$.
- If (r,s) is a pair of double nodes than r is an endnode iff s is. In this case we have $\langle (g)_t \rangle \equiv_{A1,2} a + p \equiv_{A1,2} \langle (h)_t \rangle$ if $t \rightarrow^a s$ is not an edge in g and $\langle (g)_t \rangle \equiv_{A1,2} a + a + p \equiv_{A2,3} a + p \equiv_{A1,2} \langle (h)_t \rangle$ otherwise.
- Finally if $t \rightarrow^a s$ is a manifestly inert τ-edge and s is an endnode in g, we have $\langle (g)_t \rangle \equiv_{A1,2} a\cdot\tau + p \equiv_{H1} a + p \equiv_{A1,2} \langle (h)_t \rangle$ if $t \rightarrow^a s$ is not an edge in g and $\langle (g)_t \rangle \equiv_{A1,2} a\cdot\tau + a + p \equiv_{H1} a + a + p \equiv_{A2,3} a + p \equiv_{A1,2} \langle (h)_t \rangle$ otherwise. □

Now we are in the position to prove the completeness of BPA + H1-H2 with respect to $G_{BPA/\leftrightarrows_{rb}}$:

PROOF OF THEOREM 2.6: (soundness) The fact that $(G_{BPA/\leftrightarrows_{rb}},+,\cdot,Act)$ is a model for BPA + H1-H2 follows easily by inspection of the axioms of BPA + H1-H2.
(completeness) Let $(G_{BPA/\leftrightarrows_{rb}},+,\cdot,Act) \models p=q$ for two closed BPA_τ-terms p,q, then by definition $[\![p]\!] \leftrightarrows_{rb} [\![q]\!]$. Let g and h be the unique normal forms of $[\![p]\!]$ and $[\![q]\!]$ with respect to $\rightarrow_H$. By proposition 4.2 we find $g \leftrightarrows_{rb} [\![p]\!] \leftrightarrows_{rb} [\![q]\!] \leftrightarrows_{rb} h$. From theorem 4.1 it follows that g and h must be in normal form in the sense of section 3 and by the normal form theorem (theorem 3.6) it then follows that $g \cong h$. Thus we find BPA + H1-H2 $\vdash p = \langle [\![p]\!] \rangle = \langle g \rangle = \langle h \rangle = \langle [\![q]\!] \rangle = q$ using propositions 4.3, 4.6 and 4.8. So BPA + H1-H2 is a complete axiomatization of $G_{BPA/\leftrightarrows_{rb}}$. □

Next we will prove the other completeness theorems, using the earlier results in this section. In fact we will extend the graph rewriting system to one which is 'typical' for the corresponding bisimulation relation. The rewrite rules which are added to the system are derived from figure 1: in case of η-bisimulation we will *saturate* the graph by exhaustively adding edges of the kind of figure 1 (c), whereas in the case of delay bisimulation we add edges as in figure 1 (b). For τ-bisimulation we do both. This way we obtain normal forms which are saturated and which turn out to be unique modulo rooted branching bisimulation. From there we establish the completeness result precisely in the same way as before.

DEFINITION 4.8 Let $a \in Act$, then:

1. The rewriting relation $\rightarrow_\eta$ is defined on $\boldsymbol{H}$ by the rule:
   if a graph has a path $s \rightarrow^a s_1 \rightarrow^\tau s'$ without an edge $s \rightarrow^a s'$ then add $s \rightarrow^a s'$.
2. The rewriting relation $\rightarrow_d$ is defined on $\boldsymbol{H}$ by the rule:
   if a graph has a path $s \rightarrow^\tau s_1 \rightarrow^a s'$ without an edge $s \rightarrow^a s'$ then add $s \rightarrow^a s'$.
3. Furthermore, we set: $\rightarrow_\tau = \rightarrow_\eta \cup \rightarrow_d$.

Applications of $\rightarrow_\eta$, $\rightarrow_d$ or $\rightarrow_\tau$ are referred to as *saturation steps* (cf. BERGSTRA & KLOP (1988)).

PROPOSITION 4.9 *The relations* $\rightarrow_\eta$, $\rightarrow_d$ *and* $\rightarrow_\tau$ *satisfy the following properties:*

i. $\boldsymbol{H}$ *as well as* $\boldsymbol{H}^+$ *are closed under applications of* $\rightarrow_\eta$, $\rightarrow_d$ *and* $\rightarrow_\tau$

ii. $\rightarrow_\eta$, $\rightarrow_d$ *and* $\rightarrow_\tau$ *are confluent and terminating.*

PROOF (i) Directly from definition 4.8.

(ii) (termination) Let $g \in \boldsymbol{H}$. Let n(g) be the (finite) number of nodes in g, l(g) be the number of labels and e(g) be the number of edges in g. Note that n(g) and l(g) are not changed by $\rightarrow_\eta$, $\rightarrow_d$ and $\rightarrow_\tau$ whereas e(g) increases with every saturation step. Since g is finite we find that $e(g) < n(g) \times l(g) \times n(g)$ and so $n(g) \times l(g) \times n(g) - e(g)$ is positive and decreasing with the number of saturation steps.

(confluence) $\rightarrow_\eta$, $\rightarrow_d$ and $\rightarrow_\tau$ do not eliminate redexes. □

So from proposition 4.9 we find that any graph $g \in \boldsymbol{H}$ has unique normal forms with respect $\rightarrow_\eta$, $\rightarrow_d$ and $\rightarrow_\tau$. These are written as H(g), D(g) and T(g) and (in that order) are called η-, d- and τ-*saturated*. The latter is also often referred to as the *transitive closure* of τ-steps. Furthermore, saturation preserves the corresponding bisimulation:

PROPOSITION 4.10 *For all* $g,h \in \boldsymbol{H}$:

i. *if* $g \rightarrow_\eta h$ *then* $g \leftrightarrows_{r\eta} h$

ii. *if* $g \rightarrow_d h$ *then* $g \leftrightarrows_{rd} h$

iii. *if* $g \rightarrow_\tau h$ *then* $g \leftrightarrows_{r\tau} h$.

The proof of the proposition 4.10 is straightforward.

THEOREM 4.11 (normal form theorem) *Let* $g,h \in \boldsymbol{H}$, *then*

i. *if g and h are* η*-saturated process graphs, then* $g \leftrightarrows_{r\eta} h$ *if and only if* $g \leftrightarrows_{rb} h$

ii. *if g and h are d-saturated process graphs, then* $g \leftrightarrows_{rd} h$ *if and only if* $g \leftrightarrows_{rb} h$

iii. *if g and h are* τ*-saturated process graphs, then* $g \leftrightarrows_{r\tau} h$ *if and only if* $g \leftrightarrows_{rb} h$.

PROOF We will only prove (i). The other cases proceed in the same way.
Suppose that R: $g \leftrightarrows_{r\eta} h$ then it is sufficient to prove that R is a rooted branching bisimulation:
(i) The roots of H(g) and H(h) are related and (iii) R satisfies the root condition.
(ii) If R(r,s) and $r \to^a r'$ then either $a=\tau$ and R(r',s), or $s \Rightarrow s_1 \to^a s_2 \Rightarrow s'$ such that $R(r,s_1)$ and R(r',s'). Let $t_1,...,t_k$ be such that $s_2 = t_0 \to^\tau t_1 \to^\tau \cdots \to^\tau t_k = s'$ $(k \geq 0)$ then since g and h are η-saturated there are edges $s_1 \to^a t_i$ and so there is a path $s \Rightarrow s_1 \to^a s'$. □

COROLLARY
i. $g \leftrightarrows_{r\eta} h$ *if and only if* $H(g) \leftrightarrows_{rb} H(h)$ *if and only if* $N(H(g)) \cong N(H(h))$
ii. $g \leftrightarrows_{rd} h$ *if and only if* $D(g) \leftrightarrows_{rb} D(h)$ *if and only if* $N(D(g)) \cong N(D(h))$
iii. $g \leftrightarrows_{r\tau} h$ *if and only if* $T(g) \leftrightarrows_{rb} T(h)$ *if and only if* $N(T(g)) \cong N(T(h))$.

PROOF It follows by proposition 4.10 that $H(g) \leftrightarrows_{r\eta} g$, $D(g) \leftrightarrows_{rd} g$ and $T(g) \leftrightarrows_{r\tau} g$. Now apply the normal form theorems 4.11 and 3.6. □

So we find that in each r*-bisimulation equivalence class of finite process graphs for $* \in \{\tau,\eta,d\}$ there is exactly one *-saturated process graph up to rooted branching bisimulation and exactly one *-saturated normal form up to isomorphism. In order to prove the completeness theorems we still need to prove that rewriting steps correspond to proof steps.

PROPOSITION 4.12 *For finite graphs g and h:*
i. *If* $g \to_\eta h$ *then* A1-A3 + H1,3 $\vdash \langle g \rangle = \langle h \rangle$
ii. *If* $g \to_d h$ *then* A1-A3 + T2 $\vdash \langle g \rangle = \langle h \rangle$
iii. *If* $g \to_\tau h$ *then* A1-A3 + T1-3 $\vdash \langle g \rangle = \langle h \rangle$.

PROOF (i) If $r \to^a r' \to^\tau r'' \to$ is a path is g and $r \to^a r''$ is added in g to obtain h, then we find that $\langle (g)_r \rangle \equiv_{A1\text{-}3} \langle (g)_r \rangle + a\cdot\langle (g)_{r'} \rangle$ and $\langle (g)_{r'} \rangle \equiv_{A1\text{-}3} \tau\cdot\langle (g)_{r''} \rangle + \langle (g)_{r'} \rangle$ and hence:
A1-A3 + H3 $\vdash \langle (g)_r \rangle = \langle (g)_r \rangle + a\cdot(\tau\cdot\langle (g)_{r''} \rangle + \langle (g)_{r'} \rangle) =$
$= \langle (g)_r \rangle + a\cdot(\tau\cdot\langle (g)_{r''} \rangle + \langle (g)_{r'} \rangle) + a\cdot\langle (g)_{r''} \rangle$ (by H3) $= \langle (g)_r \rangle + a\cdot\langle (g)_{r''} \rangle = \langle (h)_r \rangle$.
In case $r \to^a r' \to^\tau r''$ and r'' is an endnode we find:
A1-A3 + H1,3 $\vdash \langle (g)_r \rangle = \langle (g)_r \rangle + a\cdot(\tau + \langle (g)_{r'} \rangle) = \langle (g)_r \rangle + a\cdot(\tau\cdot\tau + \langle (g)_{r'} \rangle)$ (by H1) =
$= \langle (g)_r \rangle + a\cdot(\tau\cdot\tau + \langle (g)_{r'} \rangle) + a\cdot\tau$ (by H3) $= \langle (g)_r \rangle + a = \langle (h)_r \rangle$.
From A1-A3 + H3 $\vdash \langle (g)_r \rangle = \langle (h)_r \rangle$ it easily follows that A1-A3 + H3 $\vdash \langle g \rangle = \langle h \rangle$.
(ii) If $r \to^\tau r' \to^a r'' \to$ is a path is g and $r \to^a r''$ is added in g to obtain h, then:
$\langle (g)_r \rangle \equiv_{A1\text{-}3} \langle (g)_r \rangle + \tau\cdot\langle (g)_{r'} \rangle$ and $\langle (g)_{r'} \rangle \equiv_{A1\text{-}3} a\cdot\langle (g)_{r''} \rangle + \langle (g)_{r'} \rangle$ and hence:
A1-A3 + T2 $\vdash \langle (g)_r \rangle = \langle (g)_r \rangle + \tau\cdot(a\cdot\langle (g)_{r''} \rangle + \langle (g)_{r'} \rangle) =$
$= \langle (g)_r \rangle + a\cdot\langle (g)_{r''} \rangle$ (by T2 and A3) $= \langle (h)_r \rangle$.
In case $r \to^a r' \to^\tau r''$ and r'' is an endnode we simply leave out $\cdot\langle (g)_{r''} \rangle$ in the argument above.
Hence A1-A3 + T2 $\vdash \langle g \rangle = \langle h \rangle$.
(iii) Immediately from (i) and (ii). Note that H1 = T1 and H3 = T3. □

PROOFS OF THE THEOREMS 2.5, 2.7 AND 2.8

The soundness theorems follow easily after inspection of the axioms. Of the completeness theorems we only prove theorem 2.7. The others proceed in the same way.

Let $(G_{BPA}/\leftrightarrows_{r\eta},+,\cdot,Act) \models p=q$ for two closed BPA_τ-terms p,q, then by definition $[\![p]\!] \leftrightarrows_{r\eta} [\![q]\!]$. Let g and h be the unique normal forms of $[\![p]\!]$ and $[\![q]\!]$ with respect to $\rightarrow_\eta$. By proposition 4.10 we find $g \leftrightarrows_{r\eta} [\![p]\!] \leftrightarrows_{r\eta} [\![q]\!] \leftrightarrows_{r\eta} h$. The graphs g and h must be η-saturated and by the normal form theorem (theorem 4.11) it then follows that $g \leftrightarrows_{rb} h$. Thus we find BPA + H1-H3 $\vdash$ $p = \langle [\![p]\!] \rangle = \langle g \rangle = \langle h \rangle = \langle [\![q]\!] \rangle = q$ using propositions 4.6 and 4.12 and theorem 2.6. So BPA + H1-H3 is a complete axiomatization of $G_{BPA}/\leftrightarrows_{r\eta}$. □

5. REFINEMENT

Virtually all semantic equivalences employed in theories of concurrency are - as in this paper - defined in terms of *actions* that concurrent systems may perform. Mostly, these actions are taken to be *atomic*, meaning that they are considered not to be divisible into smaller parts. In this case, the defined equivalences are said to be based on *action atomicity*.

However, in the top-down design of distributed systems it might be fruitful to model processes at different levels of abstraction. The actions on an abstract level then turn out to represent complex processes on a more concrete level. This methodology does not seem compatible with non-divisibility of actions and for this reason PRATT (1986), LAMPORT (1986) and others plead for the use of semantic equivalences that are not based on action atomicity.

As indicated in CASTELLANO, DE MICHELIS & POMELLO (1987), the concept of action atomicity can be formalized by means of the notion of *refinement of actions*. A semantic equivalence is *preserved under action refinement* if two equivalent processes remain equivalent after replacing all occurrences of an action a by a more complex process r(a). In particular, r(a) may be a sequence of two actions a_1 and a_2. An equivalence is strictly based on action atomicity if it is not preserved under action refinement.

In the previous sections in this paper we argued that MILNER's notion of observation equivalence does not respect the branching structure of processes, and proposed the finer notion of *branching bisimulation equivalence* which does. In this section we moreover find, that observation equivalence is not preserved under action refinement, whereas branching bisimulation equivalence is.

From the axioms T3 (see table 2), it is easy to show why the notion of observation congruence is not preserved under refinement of actions: replacing the action a by the term bc, we obtain $bc(\tau x + y) = bc(\tau x + y) + bcx$, which obviously is not valid in $G/\leftrightarrows_{r\tau}$. Applying T3, we *do* find $bc(\tau x + y) = b(c(\tau x + y) + cx)$, unfortunately denoting a different process however.

In this section we will prove that branching equivalence *is* preserved under refinement of actions, and so it allows us to look at actions as abstractions of much larger structures. We will present our result in the style of BPA, and indicate afterwards how our construction can be adapted to obtain refinement theorems in the style of CCS and ACP. Put A=Act\{0} (or A=Act\{0,√} if there are √-labels around). Consider the following definitions.

DEFINITION 5.1 (substitution) Let r: A $\rightarrow$ G_{BPA} be a mapping from observable actions to graphs, and suppose g$\in$ G_{BPA}. Then, the graph r(g) can be found as follows.
For every edge r $\rightarrow^a$ r' (a$\in$ A) in g, take a copy $\underline{r(a)}$ of r(a) ($\in G_{BPA}$). Next, identify r with the root node of $\underline{r(a)}$, and r' with all endnodes of $\underline{r(a)}$, and remove the edge r $\rightarrow^a$ r'.

Note that in this definition it is never needed to identify r and r', since r(g) is non-trivial. This way, the mapping r is extended to the domain G_{BPA}. Note that since $\tau \notin$ A, τ-edges cannot be substituted by graphs. Finally, observe that every node in g is a node in r(g).

DEFINITION 5.2 (preservation under action refinement) An equivalence $\approx$ on G_{BPA} is said to be *preserved under refinement of actions* if for every mapping r: A $\rightarrow$ G_{BPA}, we have: g $\approx$ h $\Rightarrow$ r(g) $\approx$ r(h).

In other words, an equivalence $\approx$ is preserved under refinement if it is a congruence with respect to every substitution operator r.
Starting from a relation R: g $\leftrightarrows_{rb}$ h, we construct a branching bisimulation r(R): r(g) $\leftrightarrows_{rb}$ r(h), proving that preserving branching congruence, every edge with a label from A can be replaced by a root unwound non-trivial graph.

DEFINITION 5.3 Let r: A $\rightarrow$ G_{BPA} be a mapping from observable actions to graphs, g,h$\in$ G_{BPA} and R: g $\leftrightarrows_{rb}$ h. Now r(R) is the smallest relation between nodes of r(g) and r(h), such that:

1. R $\subseteq$ r(R).
2. If r $\rightarrow^a$ r' and s $\rightarrow^a$ s' (a$\in$ A) are edges in g and h such that R(r,s) and R(r',s'), and both edges are replaced by copies $\underline{r(a)}$ and $\overline{r(a)}$ of r(a) respectively, then nodes from $\underline{r(a)}$ and $\overline{r(a)}$ are related by r(R) iff they are copies of the same node in r(a).

Edges r $\rightarrow^a$ r' and s $\rightarrow^a$ s' (a$\in$ A) such that R(r,s) and R(r',s'), will be called *related* by R, as well as the copies $\underline{r(a)}$ and $\overline{r(a)}$ that are substituted for them. Observe, that on nodes from g and h the relation r(R) is equal to R. Note that if r(R)(r,s), then r is a node in g iff s is a node in h.

THEOREM 5.1 (refinement) *Branching congruence is preserved under refinement of actions.*

PROOF We prove that R: g $\leftrightarrows_{rb}$ h $\Rightarrow$ r(R): r(g) $\leftrightarrows_{rb}$ r(h) by checking the requirements. For convenience, in the definition of branching equivalence (definition 1.4), we omit the

requirement of the existence of a path $s_2 \Rightarrow s'$, as it is redundant (see the remark just after definition 1.6). Then we find:

i. The root nodes of r(g) and r(h) are related by r(R).

ii. Assume r(R)(r,s) and in r(g) there is an edge $r \rightarrow^a r'$. Then there are two possibilities (similarly in case $r \rightarrow^a r'$ stems from r(h)):

(1) The nodes r and s originate from g and h. Then R(r,s), and by the construction of r(g) we find that either $a=\tau$ and $r \rightarrow^\tau r'$ was already an edge in g, or g has an edge $r \rightarrow^b r^*$ and $r \rightarrow^a r'$ is a copy of an initial edge from r(b).

In the first case it follows from R: $g \leftrightarrows_{rb} h$ that either R(r',s) - hence r(R)(r',s) - or in h there is a path $s \Rightarrow s_1 \rightarrow^\tau s'$ such that $R(r,s_1)$ and R(r',s'). By definition of refinement, the same path also exists in r(h), and thus we have $r(R)(r,s_1)$ and r(R)(r',s').

In the second case there must be a corresponding path $s \Rightarrow s_1 \rightarrow^b s^*$ in h such that $R(r,s_1)$ and R(r*,s*). Then, in r(h) we find a path $s \Rightarrow s_1 \rightarrow^a s'$ (by replacing $\rightarrow^b$ by r(b)) such that $r(R)(r,s_1)$ and r(R)(r',s').

(2) The nodes r and s originate from related copies $\underline{r(b)}$ and $\overline{r(b)}$ of a substituted graph r(b) (for some $b \in A$), and are no copies of root or endnodes in r(b). Then $r \rightarrow^a r'$ is an edge in $\underline{r(b)}$. From r(R)(r,s) we find that r and s are copies of the same node from r(b). So, there is an edge $s \rightarrow^a s'$ in $\overline{r(b)}$ where s' is a copy of the node in r(b), corresponding with r'. Clearly r(R)(r',s').

iii. Since for nodes from g and h we have r(R)(r,s) iff R(r,s), the root condition is satisfied. □

With respect to closed BPA_τ-terms, the refinement theorem can be proved much easier by syntactic analysis of proofs, instead of working with equivalences between graphs. For observe that the axioms A1-A5 + H1-H2, that form a complete axiomatization of branching congruence for closed terms, do *not* contain any occurrences of (atomic) actions from A. Now assume we have a proof of some equality s=t between closed terms, then this proof consists of a sequence of applications of axioms from A1-A5 + H1-H2. Since all these axioms are universal equations without actions from A, the actions from s and t can be replaced by general variables, and the proof will still hold. Hence, every equation is an instance of a universal equation *without* any actions. Immediately we find that we can substitute arbitrary closed terms for these variables, obtaining refinement for closed terms.

Nevertheless, the semantic proof of the refinement theorem is important since it also holds for larger graphs from G_{BPA} that cannot be represented by closed BPA_τ-terms.

In the setting of BCCS, a substitution should be a mapping $r: A \rightarrow G_{CCS}\backslash\{0\}$, where 0 denotes the trivial graph. Then the semantic proof of the refinement theorem goes exactly as in the setting of BPA. However the syntactic proof breaks down on the absence of general sequential composition and on the presence of actions in the axioms for branching congruence. In the setting of basic ACP, definition 5.1 should be adapted such that r' is identified not with all endnodes of $\underline{r(a)}$, but with all

nodes of r(a) that have an outgoing termination edge. These termination edges should then be deleted. Furthermore if certain parts in the resulting graph have become disconnected from the root, they should be deleted as well. Now both the semantic and the syntactic proof of the refinement theorem remain valid. Finally it should be noted that refinement as defined in this section is a sensible notion that can be used in the design of systems *only* if these system are assumed to be sequential (i.e. performing only one action at a time). In the presence of parallel composition, process graphs as presented here are not sufficiently expressive for defining a refinement operator. For this pupose one may better use causality based models of concurrency, such as event structures or Petri nets (cf. VAN GLABBEEK & GOLTZ (1990)).

6. COMPARISON

In this section we compare branching with η-, delay and τ-bisimulation and list the differences and similarities that occurred to us.

6.1. BRANCHING TIME

The main difference between branching and τ-bisimulation equivalence is that the former notion preserves the branching structure of processes whereas the latter does not. This has been elaborated in the sections 1 and 3. If one argues that branching equivalence is too fine, since it does not correspond to a natural testing scenario, the same argement can be used to move from τ-bisimulation to one of the decorated trace equivalences, which are even coarser. On the other hand, if one favours τ-bisimulation over the decorated trace semantics since it preserves the internal structure of processes and is therefore independent from any particular testing scenario, a consequent application of this argument points in the direction of the finer notion of branching bisimulation semantics.

6.2. EQUIVALENCE VERSUS CONGRUENCE

τ-bisimulation equivalence is not a congruence for +, and therefore τ-bisimulation congruence is defined as the closure of τ-bisimulation equivalence under contexts, or by means of the root condition. In this respect η-, delay and branching bisimulation behave exactly the same. However, each τ-bisimulation equivalence class consists of at most two τ-bisimulation congruence classes (this follows from exersice 7.6 of HENNESSY in MILNER (1980)), as is the case for delay bisimulation, whereas η- and branching bisimulation equivalence classes may contain many congruence classes. Nevertheless, for all four bisimulations there exists a close relationship between rooted and non-rooted bisimulation, since the root condition (definition 2.2) only works on the root nodes:

THEOREM 6.1 *For all root unwound graphs g and h and* $* \in \{\tau,b,\eta,d\}$ *we have:*
$g \leftrightarrows_* h$ *if and only if* $\tau{\cdot}g \leftrightarrows_{r*} \tau{\cdot}h$.

PROOF If R is a $*$-bisimulation between g and h and r,s are the roots of $\tau \cdot g$ and $\tau \cdot h$ then $R \cup \{r,s\}$ is a rooted $*$-bisimulation between $\tau \cdot g$ and $\tau \cdot h$. On the other hand, if R is a rooted $*$-bisimulation between $\tau \cdot g$ and $\tau \cdot h$, then the roots of g and h are related by R, so R restricted to the nodes of g and h is a $*$-bisimulation between g and h. □

This theorem provides us with a tool to decide upon $*$-bisimulation equivalence, using the axiom systems of $*$-bisimulation congruence.

6.3. DIVERGENCE

In the literature on bisimulation semantics roughly three ways are suggested for treating divergence (= infinite τ-paths). The original notion of τ-bisimulation equivalence (HENNESSY & MILNER (1980), MILNER (1980) and PARK (1981)) abstracted from all divergencies; the first two graphs of figure 11 are equivalent, as well as the two graphs of figure 8.

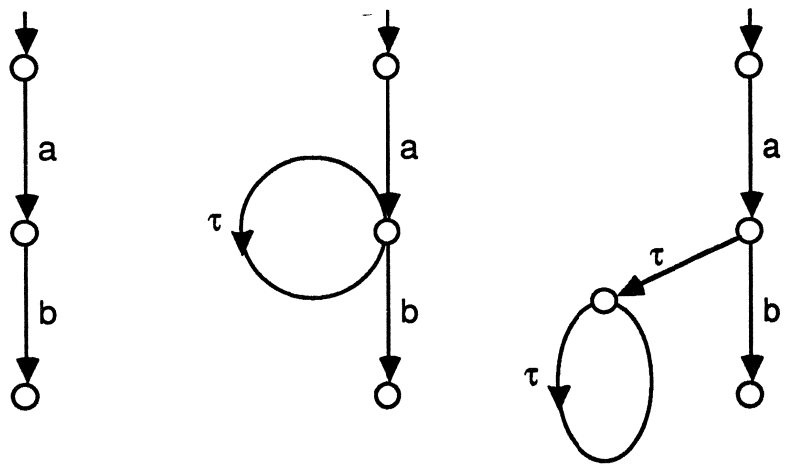


Figure 11. Three ways of modeling divergence.

These identifications can be justified by an appeal to fairness (MILNER (1980) and BAETEN, BERGSTRA & KLOP (1987a)), and play a crucial role in many protocol verifications. In BERGSTRA, KLOP & OLDEROG (1987) the corresponding semantics is refered to as *bisimulation semantics with fair abstraction.* A variant were divergence is taken into account, in the sence that the first two graphs of figure 11 are distinguished, as well as the two graphs of figure 8, was proposed in HENNESSY & PLOTKIN (1980) for τ-bisimulation and in MILNER (1981) for delay bisimulation. In both cases a complete axiomatization is provided in WALKER (1990). In these semantics the basic notion is a *preorder* rather then an equivalence, and divergence is identified with

underspecification. The induced equivalences identify the last two graphs of figure 11, which are distinghuished in τ-bisimulation semantics with fair abstraction. Hence the two notions are incomparable. A semantics that refines both notions was proposed in BERGSTRA, KLOP & OLDEROG (1987) under the name *bisimulation semantics with explicite divergence.*
η-, delay and branching bisimulation as presented in this paper are all based on the variant of τ-bisimulation with fair abstraction. However it is completely straightforward to generalize the *τ-bisimulation preorder* of HENNESSY & MILNER (1980) to a *η-bisimulation preorder*, and the *delay bisimulation preorder* of MILNER (1980) to a *branching bisimulation preorder.* Also it is not difficult to define *η-, delay* and *branching bisimulation with explicit divergence* in the spirit of BERGSTRA, KLOP & OLDEROG (1987). For branching bisimulation the definition can conveniently be given in terms of coloured traces.

DEFINITION 6.1 A node in a coloured graph is *divergent* if it is the starting point of an infinite path of which all nodes have the same colour. A colouring *preserves divergence* if no divergent node has the same colour as a non-divergent node. Two graphs g and h are *(rooted) branching bisimulation equivalent with explicit divergence* if there exists a (rooted) consistent divergence preserving colouring on g and h for which they have the same coloured trace set.

6.4. ADEQUACY FOR MODAL LOGICS

As mentioned in the introduction, τ-bisimulation semantics is not adequate for a modal logic with eventually operator. From the example in the introduction one can see that the problem originates from the circumstance that τ-bisimulation equivalence does not preserve the branching structure of processes, and indeed one can easily proof that such an operator would cause no problems in branching bisimulation semantics, at least not in the variant with explicit divergence. In fact, a much stronger result has been proved in DE NICOLA & VAANDRAGER (1990).

The Computation Tree Logic CTL* (EMERSON & HALPERN (1986)) is a very powerful logic, combining both branching time and linear time operators. It is a generalization of CTL (CLARKE & EMERSON (1981)), that contains only branching time operators. CTL* is interpreted on Kripke structures (directed graphs of which the nodes are labeled with sets of atomic propositions). DE NICOLA & VAANDRAGER (1990) established a translation from process graphs to Kripke structures, so that CTL* can also be regarded as a logic on process graphs. One of the operators of CTL/CTL*, the nexttime operator X, makes it possible to see when an (invisible) action takes place, and is therefore incompatible with abstraction. This operator was also criticized by LAMPORT (1983). BROWNE, CLARKE & GRÜMBERG (1988) found that CTL-X and CTL*-X induce the same equivalence on Kripke structures, which they characterized as *stuttering equivalence.* In DE NICOLA & VAANDRAGER (1990) branching bisimulation, after being translated to Kripke structures, is shown to coincide with stuttering equivalence. (To be precise, they consider two variants of CTL*, that correspond to two variants of stuttering equivalence and two variants of branching bisimulation, namenly *divergence blind branching bisimulation* (our notion with fair

abstraction) and *divergence sensitive branching bisimulation* (defined as branching bisimulation with fair abstraction above, but also considering endnodes to be divergent). The stuttering equivalence of BROWNE, CLARKE & GRÜMBERG (1988) is the divergence sensitive variant.) Hence (divergence sensitive) branching bisimulation is adequate for CTL*-X. Since the eventually operator of GRAF & SIFAKIS (1987) can be expressed in CTL*, this implies that it causes no problems in branching bisimulation semantics.

6.5. MODAL CHARACTERIZATIONS

It is well known (cf. HENNESSY & MILNER (1985)) that observation equivalence can be characterized by means of a simple modal language, called Hennessy-Milner logic (HML). The question arises if such a result can also be obtained for branching equivalence. As pointed out above, CTL-X characterizes branching equivalence, but this language is rather strong. Another possibility is adding the eventually operator to HML. It remains to be determined for which classes of process graphs HML + 'eventually' is adequate. In DE NICOLA & VAANDRAGER (1990) it has been shown that adding an 'until' operator to HML is sufficient.

6.6. BACK AND FORTH BISIMULATIONS

In DE NICOLA, MONTANARI & VAANDRAGER (1989) it has been established that if in the definition of $*$-bisimulation, for $* \in \{\tau,b,\eta,d\}$, it is required that moves in the one process can be simulated by the other process, not only when going forward but also when going back in history, these modified notions all coincide with branching bisimulation. This also yields another modal characterization of branching bisimulation, namely HML with backward modalities.

6.7. PRACTICAL APPLICATIONS OF BRANCHING TIME

The extra identifications made in τ-bisimulation semantics on top of branching bisimulation semantics can be cumbersome in certain applications of the theory. See the remark in the introduction.

6.8. REFINEMENT OF ACTIONS

For sequential processes branching bisimulation is preserved under refinement of actions, whereas τ-bisimulation is not. This was established in section 5, which appeared before as VAN GLABBEEK & WEIJLAND (1989). A proof can also be found in DARONDEAU & DEGANO (1989). Delay bisimulation is also preserved under action refinement, and η-bisimulation is not. Moreover delay bisimulation is the coarsest equivalence that is preserved under refinement and finer then τ-bisimulation, and branching bisimulation is the coarsest equivalence that is preserved under refinement and finer then η-bisimulation. This was established in CHERIEF & SCHNOEBELEN (1990).

6.9. AXIOMATIZATIONS AND REWRITE SYSTEMS

All $*$-bisimulations ($* \in \{\tau,b,\eta,d\}$) have relatively simple equational characterizations (see section 2), but the axiom system for branching bisimulation can easily be turned in a complete term rewriting system, which is not the case for the other notions.

6.10. COMPLEXITY

In GROOTE & VAANDRAGER (1990) an algorithm is presented for deciding branching bisimulation equivalence between finite-state processes, with (time) complexity $O(l+n\cdot m)$. Here l is the size of Act, n is the number of nodes in the investigated process graphs and m the number of edges. The fastest algorithm for τ-bisimulation equivalence up till now has complexity $O(l\cdot n^{2.376})$. In general $n \leq m \leq l\cdot n^2$, so it depends on the density of edges in a graph which algorithm is faster. In a trial implementation of the scheduler of MILNER (1980), reported in GROOTE & VAANDRAGER (1990), branching bisimulation turned out to be much faster. Furthermore, it turned out that in such automatic verifications the space complexity was a much more serious handicap then the time complexity (the τ-bisimulation tools suffered from lack of memory already when applied to processes with 15.000 states). The space complexity of the algorithm of GROOTE & VAANDRAGER (1990) is $O(n+m)$, which is less than the space complexity of the τ-bisimulation algorithm.

6.11. CORRESPONDENCE

Finally we present a theorem which tells us that in quite a number of cases observation and branching bisimulation 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 6.2 *Suppose* g *and* h *are two graphs, and* g *is without edges labeled 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* $g \leftrightarrows_{rb} h$.

PROOF Let R be the *largest* (rooted) τ-bisimulation between g and h. We show that R is even a (rooted) branching bisimulation. Assume that R(r,s) and $r \rightarrow^a r'$ is an edge in g, then either $a=\tau$ and R(r',s) - contradicting the absence of τ-edges in g - or in h there is a path $s \Rightarrow s_1 \rightarrow^a s_2 \Rightarrow s'$ and R(r',s'). Assume $s \Rightarrow s_1$ has the form $s = v_0 \rightarrow^\tau v_1 \rightarrow^\tau \cdots \rightarrow^\tau v_m = s_1$ ($m \geq 0$) then it

follows from $s \rightarrow^{\tau} v_1$ and R(r,s) that for some r_1: $r \Rightarrow r_1$ and $R(r_1,v_1)$. Since g has no τ-edges we find that $r=r_1$. Repeating this argument m times we find that $R(r,v_i)$ and $R(r,s_1)$.
Furthermore, since $R(r,s_1)$ and $s_1 \rightarrow^{a} s_2$ we find that $r \rightarrow^{a} r''$ (g has no τ-steps) such that $R(r'',s_2)$. Since $s_2 = w_0 \rightarrow^{\tau} w_1 \rightarrow^{\tau} \cdots \rightarrow^{\tau} w_n = s'$ it follows from the same argument as before that $R(r'',w_i)$ and R(r'',s'). Thus we find R(r',s'), R(s',r'') and $R(r'',s_2)$ and since R is the largest rooted τ-bisimulation we have $R(r',s_2)$.
On the other hand, if R(r,s) and $r \rightarrow^{a} r'$ is an edge in h, then either $a=\tau$ and R(r',s) or directly $s \rightarrow^{a} s'$ such that R(r',s'), since g contains no τ-edges. □

For η- instead of branching bisimulation equivalence this theorem was already proven in BAETEN & VAN GLABBEEK (1987). From theorem 6.1 we easily find that for graphs g and h:

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

REFERENCES

S.ABRAMSKY (1987), *Observation equivalence as a testing equivalence*, TCS 53, pp. 225-241.

J.C.M.BAETEN, J.A.BERGSTRA & J.W.KLOP (1987a), *On the consistency of Koomen's fair abstraction rule.* TCS 51(1/2), pp. 129-176.

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

J.C.M.BAETEN & R.J.VAN GLABBEEK (1987), *Another look at abstraction in process algebra*, proc. ICALP 87 (Th.Ottman ed.) Karlsruhe, LNCS 267, Springer-Verlag, pp.84-94.

J.C.M.BAETEN & W.P.WEIJLAND (1990), *Process Algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, Cambridge.

J.W.DE BAKKER, J.A.BERGSTRA, J.W.KLOP & J.-J.Ch.MEIJER (1984), *Linear time and branching time semantics for recursion with merge*, TCS 34, pp. 135-156.

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

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

J.A.BERGSTRA & J.W.KLOP (1988), *A complete inference system for regular processes with silent moves*, proc. Logic Colloquium 1986 (F.R.Drake & J.K.Truss, eds) Hull, North Holland, pp. 21-81.

J.A.BERGSTRA, J.W.KLOP & E.-R.OLDEROG (1987), *Failures without chaos: a new process semantics for fair abstraction*, Formal Description of Programming Concepts - III, proc. third IFIP WG 2.2 working conference (M.Wirsing ed.) Ebberup 1986, North Holland, pp. 77-103.

B.BLOOM, S.ISTRAIL & A.R.MEYER (1988), *Bisimulation can't be traced: preliminary report*, Conference Record of the 15th ACM Symposium on Principles of Programming Languages (POPL), San Diego, California, pp. 229-239.

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

M.C.BROWNE, E.M.CLARKE & O.GRÜMBERG (1988), *Characterizing finite Kripke structures in propositional temporal logic*, TCS 59(1/2), pp. 115-131.

L.CASTELLANO, G.DE MICHELIS & L.POMELLO (1987), *Concurrency vs Interleaving: an instructive example*, Bulletin of the EATCS 31, pp. 12-15.

F. CHERIEF & PH. SCHNOEBELEN (1990), *Tau-bisimulations and full abstraction for refinement of actions,* unpublished manuscript.

E.M.CLARKE & E.A. EMERSON (1981), *Design and synthesis of synchronization skeletons using branching-time temporal logic*, Proceedings of the Workshop on Logics of Programs (D. Kozen, ed.) Yorktown Heights, LNCS 131, Springer-Verlag, pp. 52-71.

PH.DARONDEAU & P.DEGANO (1989), *About semantic action refinement*, to appear in Fundamenta Informaticae.

R.DE NICOLA & M.HENNESSY (1984), *Testing equivalence for processes*, TCS 34, pp. 83-133.

R.DE NICOLA, U.MONTANARI & F.W.VAANDRAGER (1990), *Back and forth bisimulations*, proc. CONCUR 90 (J.C.M. Baeten & J.W. Klop, eds.) Amsterdam, LNCS 458, Springer-Verlag, pp. 152-165.

R.DE NICOLA & F.W.VAANDRAGER (1990), *Three logics for branching bisimulation*, Proc. LICS 90, Philadelphia, IEEE Computer Society Press, Washington, pp. 118-129.

E.A.EMERSON & J.Y.HALPERN (1986), *'Sometimes' and 'Not Never' revisited: on branching time versus linear time temporal logic.* JACM 33(1), pp. 151-178.

R.J.VAN GLABBEEK (1990), *Comparative concurrency semantics and refinement of actions*, Ph.D. Thesis, Free University, Amsterdam.

R.J.VAN GLABBEEK & U.GOLTZ (1990), *Refinement of actions in causality based models*, proc. REX workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness (J.W. de Bakker, W.-P. de Roever & G. Rozenberg, eds.) Mook, The Netherlands, LNCS 430, Springer-Verlag, pp. 267-300.

R.J.VAN GLABBEEK & W.P.WEIJLAND (1989a), *Branching time and abstraction in bisimulation semantics (extended abstract)*, Information Processing 89, proc. IFIP 11th World Computer Congress (G.X.Ritter, ed.) San Francisco, North Holland, pp. 613-618.

R.J.VAN GLABBEEK & W.P.WEIJLAND (1989b), *Refinement in branching time semantics*, Report CS-R8922, Centrum voor Wiskunde en Informatica, Amsterdam, and proc. AMAST conference, May 1989, Iowa, pp. 197-201.

S.GRAF & J.SIFAKIS (1987), *Readiness semantics for regular processes with silent actions*, proc. ICALP 87 (Th.Ottman, ed.) Karlsruhe, LNCS 267, Springer-Verlag, pp.115-125.

J.F.GROOTE & F.W.VAANDRAGER (1990), *An efficient algorithm for branching bisimulation and stuttering equivalence*, proc. ICALP 90 (M.S.Paterson, ed.) Warwick, LNCS 443, Springer-Verlag, pp. 626-638.

M.HENNESSY & R.MILNER (1980), *On observing nondeterminism and concurrency*, in: Proceedings ICALP 80 (J.W.de Bakker & J.van Leeuwen, eds.), LNCS 85, Springer Verlag, pp. 299-309, a preliminary version of:

M.HENNESSY & R.MILNER (1985), *Algebraic laws for nondeterminism and concurrency*, Journal of the ACM 32(1), pp. 137-161.

M. HENNESSY & G.D.PLOTKIN (1980), *A term model for CCS*, proc. MFCS 80 (P.Dembiński, ed.), LNCS 88, Springer-Verlag, pp. 261-274.

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

C.A.R.HOARE (1985), *Communicating sequential processes*, Prentice Hall, London.

B.JONSSON & J.PARROW (1989), *Deciding bisimulation equivalences for a class of non-finite-state programs*, in: Proceedings STACS 89 (B.Monien & R.Cori, eds.), Paderborn, LNCS 347, Springer Verlag.

L.LAMPORT (1983), *What good is temporal logic?*, Information Processing 83 (R.A.Mason, ed.), North Holland, pp. 657-668.

L.LAMPORT (1986), *On interprocess communication. Part 1: Basic formalism*, Distributed Computing 1 (2), pp. 77-85.

R.MILNER (1980), *A calculus of communication systems*, LNCS 92, Springer-Verlag.

R.MILNER (1981), *A modal characterisation of observable machine-behaviour*, Proceedings CAAP 81, (G.Astesiano & C.Böhm, eds.), LNCS 112, Springer-Verlag, pp. 25-34.

R.MILNER (1983), *Calculi for synchrony and asynchrony*, TCS 25, pp. 267-310.

R.MILNER (1985), *Lectures on a calculus for communicating systems*, Seminar on concurrency (S.D.Brookes, A.W.Roscoe & G.Winskel eds.), pp. 197-220, LNCS 97, Springer-Verlag.

R.MILNER (1989), *Communication and concurrency*, Prentice Hall, London.

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

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

I.C.C.PHILLIPS (1987), *Refusal testing*, TCS 50, pp. 241-284.

A.PNUELI (1985), *Linear and branching structures in the semantics and logics of reactive systems*, proc. 12th ICALP conf., LNCS 194, Springer-Verlag, pp. 15-32.

L.POMELLO (1986), *Some equivalence notions for concurrent systems. An overview*, Advances in Petri Nets 1985, (G.Rozenberg,ed.), LNCS 222, Springer-Verlag, pp. 381-400.

V.R.PRATT (1986), *Modeling concurrency with partial orders*, International Journal of Parallel Programming 15 (1), pp. 33-71.

W.C.ROUND & S.D.BROOKES (1981), *Possible futures, acceptances, refusals and communicating processes*, Proceedings 22nd Annual Symposium on Foundations of Computer Science, Nashville 1981, IEEE, New York, pp. 140-149.

D.J.WALKER (1990), *Bisimulation and divergence*, Information & Computation 85(2), pp. 202-241.

W.P.WEIJLAND (1989), *Synchrony and asynchrony in process algebra*, Ph.D. Thesis, University of Amsterdam.