Branching Bisimulation as a Tool in the Analysis of Weak Bisimulation†

R.J. van Glabbeek*
Computer Science Department, Stanford University
Stanford, CA 94305, USA.
`rvg@cs.stanford.edu`

It is shown that the analysis of weak congruence can sometimes be simplified by means of a similar analysis of branching congruence as an intermediate step. This point is made through a completeness proof for an equational axiomatization of basic CCS with prefix iteration.

1 Introduction

A fundamental semantic equivalence in the study of concurrent systems is *bisimulation equivalence* [19]. The concept of bisimulation equivalence on labelled transition systems without a notion of invisible action is completely canonical. When invisible actions, or τ-steps, are introduced however, a certain degree of freedom in the definition of bisimulation equivalence emerges, even without considering *divergence*. The most prominent version of bisimulation is MILNER's notion of *weak (bisimulation) congruence* or *observation congruence* [19]. An alternative is the notion of *branching (bisimulation) congruence* of VAN GLABBEEK & WEIJLAND [14].

Since the inception of the branching bisimulation eleven advantages of the use of branching congruence over other equivalences in general and weak congruence in particular have been found:

1. Verifications in branching bisimulation semantics are sound independent of ones notion of observable behaviour [14, 13].
2. No coarser semantics (like weak bisimulation) has this property [17].
3. In abstract interleaving semantics no finer notion of bisimulation is suitable [12].
4. There is a reasonable operator for which branching bisimulation is a congruence and weak bisimulation or coarser notions are not. On the other hand no examples testifying for the opposite are known [17].
5. Branching bisimulation equivalence is the only known equivalence in the linear time – branching time spectrum that supports an 'eventually' operator as part of a temporal logic on transition systems [16]. It even supports all the operators of CTL* and corresponds with stuttering equivalence of Kripke structures [8].
6. There are practical applications in which weak bisimulation poses a problem that can be solved by moving to branching bisimulation [16]. No applications have been found in which the reverse holds [17].

*This work was supported by ONR under grant number N00014-92-J-1974.

†I don't intend to publish this paper separately, as it will be integrated with FOKKINK [11] and ACETO & INGÓLFSDÓTTIR [2] into ACETO, FOKKINK, VAN GLABBEEK & INGÓLFSDÓTTIR [1].

7. Branching congruence has a lower complexity than any other abstract semantic equivalence used in concurrency theory [18].

8. For sequential processes, branching congruence is preserved under action refinement, whereas weak congruence is not [15].

9. Branching congruence has a very appealing complete axiomatization [12]

10. and better term rewriting properties than other (abstract) bisimulations [16, 3, 6].

11. Finally, it has a nice characterization as back-and-forth bisimulation [7].

On the other hand there is one advantage of weak over branching congruence, namely that only the former one can be understood as strong bisimulation congruence on an alternative transition system, obtainable in a simple way from the original one (namely as a kind of reflexive and transitive closure) [17].

The purpose of this paper is to add a twelfth advantage of branching bisimulation to the list above. I claim that even when branching bisimulation would not be found interesting in its own right, it may serve as a better tool to prove certain properties about weak bisimulation than weak bisimulation itself.

This claim is substantiated by means of a comparison between two proofs of the completeness of an axiomatization for weak bisimulation congruence over basic CCS with prefix iteration.

In Aceto & Ingólfsdóttir [2], where the axiomatization is proposed, a completeness proof in terms of weak bisimulation is given, counting 12 pages—written out in great detail—and containing several levels of nested case-distinctions. I have examined the proof carefully and found no way to significantly shorten it, otherwise than by removal of details or using branching bisimulation.

Fokkink [10] also reports on an unsuccessful attempt to shorten this proof, which however yielded a completeness proof of an axiomatization for branching bisimulation, counting less than two pages. For the sake of fairness is should be taken into account that that proof uses a completeness result for strong bisimulation, proven in Fokkink [9] in four pages—also written out in detail. However, Fokkink [11] has recently shown how that result can be obtained in only half a page.

In this paper I show within two pages how the completeness of the axiomatization of [2] for weak congruence follows from the completeness of the axiomatization of [10] for branching congruence. Altogether the route via branching bisimulation is much simpler than the 'direct' way.

I also provide a complete axiomatization for the notion of *delay congruence*, a version of observational equivalence proposed in Milner [20]. This is even more an easy corollary of the result for branching bisimulation.

2 Basic CCS with prefix iteration

Assume a set V of *variables* and A of *actions*. Let $\tau \notin A$ denote a special *invisible action* and write $A_\tau := A \cup \{\tau\}$. Let $x, y, ...$ range over V, $a, b, ...$ over A and $\alpha, \beta, ...$ over A_τ. The language BCCS* (called MPA$^*_\delta(A_\tau)$ in [9, 2]) is given by the BNF grammar

$$P := x \mid 0 \mid \alpha.P \mid P + P \mid \alpha^* P$$

One may leave out redundant brackets, assuming that $+$ binds weaker than $\alpha.$ and α^*.

The *action relations* $\xrightarrow{\alpha}$ between terms over BCCS* are generated by the following *action rules*:

$$\alpha.x \xrightarrow{\alpha} x \qquad \frac{x \xrightarrow{\alpha} x'}{x + y \xrightarrow{\alpha} x'} \qquad \frac{y \xrightarrow{\alpha} y'}{x + y \xrightarrow{\alpha} y'} \qquad \alpha^* x \xrightarrow{\alpha} \alpha^* x \qquad \frac{x \xrightarrow{\alpha} x'}{\alpha^* x \xrightarrow{\alpha} x'}$$

Let p, q, r, s, t (possibly primed or subscripted) range over $T(\text{BCCS}^*)$, the set of *closed* BCCS* expressions (not containing variables). More on prefix iteration can be found in [9, 10, 2].

3 Bisimulations

This section recalls the definitions and congruence theorems for several kinds of bisimulation equivalences. For motivation and further information see [19, 20, 17].

Definition 1 Two processes $p, q \in T(\mathrm{BCCS}^*)$ are *strong (bisimulation) equivalent* if there exists a symmetric binary relation $\mathcal{R}$ on $T(\mathrm{BCCS}^*)$ with $p\mathcal{R}q$, such that

$$s\mathcal{R}t \wedge s \xrightarrow{\alpha} t' \text{ implies } \exists t' : t \xrightarrow{\alpha} t' \wedge s'\mathcal{R}t'.$$

As observed in [9], strong equivalence is a congruence for BCCS^*.

Write $p \Longrightarrow q$ for $\exists n \geq 0 : \exists p_0, ..., p_n : p = p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} ... \xrightarrow{\tau} p_n = q$, i.e. a (possibly empty) path of τ-steps from p to q. Furthermore, for $\alpha \in A_\tau$, write $p \xrightarrow{(\alpha)} q$ for $p \xrightarrow{\alpha} q \vee (\alpha = \tau \wedge p = q)$. Thus $\xrightarrow{(a)}$ is the same as $\xrightarrow{a}$ for $a \in A$, and $\xrightarrow{(\tau)}$ denotes zero or one τ-steps.

Definition 2 Two processes $p, q \in T(\mathrm{BCCS}^*)$ are *branching (bisimulation) equivalent*—notation $p \leftrightarroweq_b q$—if there exists a symmetric binary relation $\mathcal{R}$ on $T(\mathrm{BCCS}^*)$ with $p\mathcal{R}q$, such that

$$s\mathcal{R}t \wedge s \xrightarrow{\alpha} s' \text{ implies } \exists t_1, t_2, t' : t \Longrightarrow t_1 \xrightarrow{(\alpha)} t_2 \Longrightarrow t' \wedge s\mathcal{R}t_1 \wedge s'\mathcal{R}t_2 \wedge s'\mathcal{R}t'. \tag{1}$$

They are *branching congruent*—$p \leftrightarroweq_{rb} q$—if $\begin{array}{l} p \xrightarrow{\alpha} p' \text{ implies } \exists q' : q \xrightarrow{\alpha} q' \wedge p' \leftrightarroweq_b q' \text{ and} \\ q \xrightarrow{\alpha} q' \text{ implies } \exists p' : p \xrightarrow{\alpha} p' \wedge p' \leftrightarroweq_b q'. \end{array}$

This definition is equivalent to the one in [15, 16, 17], as follows immediately from the proof of Lemma 1.1 in [16, 17]. There a relation satisfying (1) is called a *semi branching bisimulation.*

Definition 3 *Weak (bisimulation) equivalence* [19]—notation $\leftrightarroweq_w$—is defined as $\leftrightarroweq_b$, but without the requirements $s\mathcal{R}t_1$ and $s'\mathcal{R}t_2$. Two processes p and q are *weak congruent*—notation $p \leftrightarroweq_{rw} q$—if

$$\begin{array}{l} p \xrightarrow{\alpha} p' \text{ implies } \exists q_1, q_2, q' : q \Longrightarrow q_1 \xrightarrow{\alpha} q_2 \Longrightarrow q' \wedge p' \leftrightarroweq_w q' \text{ and} \\ q \xrightarrow{\alpha} q' \text{ implies } \exists p_1, p_2, p' : p \Longrightarrow p_1 \xrightarrow{\alpha} p_2 \Longrightarrow p' \wedge p' \leftrightarroweq_w q'. \end{array}$$

Delay bisimulation equivalence [20]—$\leftrightarroweq_d$—is defined as $\leftrightarroweq_b$, but without the requirement $s\mathcal{R}t_1$.

p and q are *delay congruent*—$p \leftrightarroweq_{rd} q$—if $\begin{array}{l} p \xrightarrow{\alpha} p' \text{ implies } \exists q_1, q' : q \Longrightarrow q_1 \xrightarrow{\alpha} q' \wedge p' \leftrightarroweq_w q' \text{ and} \\ q \xrightarrow{\alpha} q' \text{ implies } \exists p_1, p' : p \Longrightarrow p_1 \xrightarrow{\alpha} p' \wedge p' \leftrightarroweq_w q'. \end{array}$

A remarked in [16], the existence requirement of a process t with $t_2 \Longrightarrow t'$ and $s'\mathcal{R}t'$ is redundant in the definitions of $\leftrightarroweq_b$ and $\leftrightarroweq_d$.

Proposition 1 (**Congruence**) For $x \in \{b, w, d\}$, $\leftrightarroweq_{rx}$ is an equivalence relation such that

$$\text{if } p \leftrightarroweq_{rx} q \text{ then } \alpha.p \leftrightarroweq_{rx} \alpha.q, \quad p + r \leftrightarroweq_{rx} q + r, \quad r + p \leftrightarroweq_{rx} r + q \text{ and } \alpha^* p \leftrightarroweq_{rx} \alpha^* q.$$

Moreover $\leftrightarroweq_{rx}$ is the coarsest relation with these properties contained in $\leftrightarroweq_x$.

Proof: Equivalence is straightforward; see [5] or [17]. However, as observed in Basten [5], the transitivity proof for $\leftrightarroweq_{rb}$ in [16] is wrong. He also shows that stating Definition 2 in terms of semi branching bisimulations prevents the error.

That $\leftrightarroweq_{rx}$ is a congruence for BCCS^* follows easily from the definitions, using that the relation

$$\{(\alpha^* P, \alpha^* Q) \mid \alpha \in A_\tau, P \leftrightarroweq_{rx} Q\} \cup \leftrightarroweq_x$$

satisfies (1) (depending on x possibly modified as in Definition 3).

That $\leftrightarroweq_{rx}$ is the coarsest congruence contained in $\leftrightarroweq_x$ is shown in [16, 17]. □

Obviously, $\leftrightarroweq_{rb}$ is a finer congruence than $\leftrightarroweq_{rw}$ and $\leftrightarroweq_{rd}$ sits in between.

4 Complete axiomatizations

Aceto & Ingólfsdóttir [2] proposed the following axiomatization for weak congruence on BCCS* expressions.

$$\begin{array}{llcl@{\qquad}llcl}
\mathrm{A1} & x+y & = & y+x & & & & \\
\mathrm{A2} & (x+y)+z & = & x+(y+z) & \mathrm{T1} & \alpha.\tau.x & = & \alpha.x \\
\mathrm{A3} & x+x & = & x & \mathrm{T3} & a.(x+\tau.y) & = & a.(x+\tau.y)+a.y \\
\mathrm{A0} & x+0 & = & x & \mathrm{MT1} & a^*(x+\tau.y) & = & a^*(x+\tau.y+a.y) \\
\mathrm{MI1} & \alpha.\alpha^*x+x & = & \alpha^*x & \mathrm{MT2} & a^*\tau.a^*x & = & \tau.a^*x \\
\mathrm{MI2} & a^*a^*x & = & a^*x & \mathrm{FIR} & \tau^*x & = & \tau.x
\end{array}$$

Figure 1: A complete axiomatization for weak bisimulation

They also derived the following equations from these axioms:

$$\begin{array}{llcl@{\qquad}llcl}
\mathrm{T2} & \tau.x & = & \tau.x+x & \mathrm{DT2} & \tau.(x+y)+x & = & \tau.(x+y) \\
\mathrm{DMI1} & a^*x & = & a^*x+x & \mathrm{MT3} & a^*(x+\tau.y) & = & a^*(x+\tau.y)+a.y
\end{array}$$

It is well known that A0–3 form a complete axiomatization for the sublanguage without * and τ, and A0–3 together with T1–3 form a complete axiomatization for weak congruence on the sublanguage without * [19]. In Fokkink [9] the axioms A0–3 and MI1–2 are shown to be complete for the sublanguage without τ (on which all bisimulation equivalences of the previous section coincide).

In [2] the entire axiomatization of Figure 1 is proven sound and complete for weak congruence on BCCS*. The new axioms MT1–2 and FIR were also shown to be independent. All completeness results mentioned here refer in the first place to closed terms. In [2], the axioms of Figure 1 were in addition shown to be *ω-complete*, which together with completeness for closed terms yields completeness for open terms.

Fokkink [10] established that the axioms A0–3 and MI1–2 together with

$$\begin{array}{lrcl}
\mathrm{B} & \alpha.(\tau.(x+y)+x) & = & \alpha.(x+y) \\
\mathrm{MI6} & \tau^*x & = & \tau.x+x \\
\mathrm{MI7} & a.a^*(\tau.a^*(x+y)+x) & = & a.a^*(x+y)
\end{array}$$

form a complete axiomatization of branching congruence on BCCS*. As remarked before, this result is much easier to obtain than the completeness result for weak congruence of [2]. Therefore I will now significantly shorten the proof of the latter result by deriving it from the former.

Proposition 2 The axioms B and MI6–7 are derivable from the axioms of Figure 1.

Proof: Let $p \stackrel{X}{=} q$ denote that the equation $p = q$ is derivable from the laws A1, A2 and X.

- $\alpha.(\tau.(x+y)+x) \stackrel{\mathrm{DT2}}{=} \alpha.\tau.(x+y) \stackrel{\mathrm{T1}}{=} \alpha.(x+y)$.
- $\tau^*x \stackrel{\mathrm{FIR}}{=} \tau.x \stackrel{\mathrm{T2}}{=} \tau.x+x$.
- $a.a^*(\tau.a^*(x+y)+x) \stackrel{\mathrm{DMI1}}{=}$
  $a.a^*(\tau.(a^*(x+y)+x+y)+x) \stackrel{\mathrm{DT2}}{=}$
  $a.a^*\tau.(a^*(x+y)+x+y) \stackrel{\mathrm{DMI1}}{=}$
  $a.a^*\tau.a^*(x+y) \stackrel{\mathrm{MT2}}{=}$
  $a.\tau.a^*(x+y) \stackrel{\mathrm{T1}}{=}$
  $a.a^*(x+y)$. $\square$

Definition 4 The *derivatives* of a closed BCCS* expression p are defined recursively by

- p is a derivative of p
- if s is a derivative of p, and $s \xrightarrow{\alpha} t$, then so is t.

A process p is *saturated* if for each of its derivatives r, s and t and $\alpha \in A_\tau$ we have:

$$r \xrightarrow{\alpha} s \xrightarrow{\tau} t \Rightarrow r \xrightarrow{\alpha} t \quad \text{and} \quad r \xrightarrow{\tau} s \xrightarrow{\alpha} t \Rightarrow r \xrightarrow{\alpha} t.$$

Theorem 1 [16] If p and q are saturated and weakly congruent, then they are branching congruent.

Proof: If $t \Longrightarrow t_1 \xrightarrow{(\alpha)} t_2 \Longrightarrow t'$ for a derivative t of a saturated process q, then $t \xrightarrow{(\alpha)} t'$. This also holds with α instead of (α). Thus if p and q are saturated and weakly equivalent, then $p\mathcal{R}q$ for a symmetric relation $\mathcal{R}$ such that $s\mathcal{R}t \wedge s \xrightarrow{\alpha} t'$ implies $\exists t' : t \xrightarrow{(\alpha)} t' \wedge s'\mathcal{R}t'$. □

Proposition 3 Any BCCS* term s is provably equal to a saturated one.

Proof: A BCCS* expression p is *in head normal form* if $p \overset{\text{A0}}{=} \Sigma_{i \in I} \alpha_i.p_i$ (with I finite). By induction on the structure of a process s, I prove s equal to a process that is both saturated and in head normal form.

The case $s = 0$ is trivial.

Let $s = p + q$. By induction p can be transformed into the required form p', and similarly q can be brought in saturated head normal form q'. Now $p' + q'$ is saturated and in head normal form.

Let $s = a.p$ with $a \neq \tau$. Prove p equal to a saturated process $p' = \Sigma_{i \in I} b_i.p_i + \Sigma_{j \in J} \tau.q_j$. By T3, s can be proven equal to $s' = a.p' + \Sigma_{j \in J} a.q_j$. This process is in head normal form and saturated. Namely the derivatives p' and q_j are saturated by assumption. So let $s' \xrightarrow{\alpha} q \xrightarrow{\tau} r$. Then $\alpha = a$ and q must be p' or q_j with $j \in J$. In case $q = p'$ it must be that $r = q_k$ with $k \in J$. It follows that $s' \xrightarrow{a} r$. In case $q = q_j$ one has $p' \xrightarrow{\tau} q \xrightarrow{\tau} r$ so it must be that $p' \xrightarrow{\tau} r$ since p' is saturated, and hence $r = q_k$ for certain $k \in J$. Again $s' \xrightarrow{a} r$. The case $s' \xrightarrow{\tau} q \xrightarrow{\alpha} r$ does not apply.

Let $s = \tau.p$. Prove p equal to a saturated process $p' = \Sigma_{i \in I} \alpha_i.p_i$. By T2, s can be proven equal to $\tau.p' + \Sigma_{i \in I} \alpha_i.p_i$. Just as above, this process is saturated and in head normal form.

Let $s = a^*p$ with $a \neq \tau$. Prove p equal to a saturated process $p' = \Sigma_{i \in I} b_i.p_i + \Sigma_{j \in J} \tau.q_j$. By MT3, s can be proven equal to $a^*p' + \Sigma_{j \in J} a.q_j$ which by MI1 equals $a.a^*p' + p' + \Sigma_{j \in J} a.q_j$. Applying MT1 gives $\vdash s = a.a^*(p' + \Sigma_{j \in J} a.q_j) + p' + \Sigma_{j \in J} a.q_j$. I show that this process s' is saturated.

- All derivatives other than s' itself and $t = a^*(p' + \Sigma_{j \in J} a.p_i)$ are saturated by assumption.
- Let $s' \xrightarrow{\alpha} q \xrightarrow{\tau} r$. There are three possibilities:
  - $\alpha = a$ and $q = t$. Then $r = q_k$ with $k \in J$ and $s' \xrightarrow{a} r$.
  - $p' \xrightarrow{\alpha} q \xrightarrow{\tau} r$. In that case $s' \xrightarrow{\alpha} r$ since p' is saturated.
  - $\alpha = a$ and $q = q_j$ for certain $j \in J$. In that case $p' \xrightarrow{\tau} q_j \xrightarrow{\tau} r$ so it must be that $p' \xrightarrow{\tau} r$ since p' is saturated, and hence $r = q_k$ for certain $k \in J$. Again $s' \xrightarrow{a} r$.
- Let $s' \xrightarrow{\tau} q \xrightarrow{\alpha} r$. Then $p' \xrightarrow{\tau} q \xrightarrow{\alpha} r$ and since p' is saturated $s' \xrightarrow{\alpha} r$.
- Let $t \xrightarrow{\alpha} q \xrightarrow{\tau} r$. There are three possibilities:
  - $\alpha = a$ and $q = t$. Then $r = q_k$ with $k \in J$ and $t \xrightarrow{a} r$.
  - $p' \xrightarrow{\alpha} q \xrightarrow{\tau} r$. In that case $t \xrightarrow{\alpha} r$ since p' is saturated.
  - $\alpha = a$ and $q = q_j$ for certain $j \in J$. In that case $p' \xrightarrow{\tau} q_j \xrightarrow{\tau} r$ so it must be that $p' \xrightarrow{\tau} r$ since p' is saturated, and hence $r = q_k$ for certain $k \in J$. Again $t \xrightarrow{a} r$.
- Let $t \xrightarrow{\tau} q \xrightarrow{\alpha} r$. Then $p' \xrightarrow{\tau} q \xrightarrow{\alpha} r$ and since p' is saturated $t \xrightarrow{\alpha} r$.

Let $s = \tau^*p$. Applying FIR reduces this to the case $s = \tau.p$. □

Corollary 1 The axiom system of Figure 1 is complete for weak congruence.

Proof: Suppose s is weakly congruent to t. Prove s and t equal to saturated processes s' and t'. These must be branching congruent. Hence $\vdash s' = t'$ using the axioms for branching congruence.

5 A complete axiomatization for delay bisimulation

The technique from the previous section can easily be simplified to find a complete axiomatization of delay bisimulation. Call a process *d-saturated* if for each of its derivatives r, s and t we have

$$r \xrightarrow{\tau} s \xrightarrow{\alpha} t \Rightarrow r \xrightarrow{\alpha} t.$$

As before it follows that if two processes are d-saturated and delay congruent, they must be branching congruent.

Proposition 4 Any BCCS* term s can be proven equal to a saturated one, using axioms A1-2, MI1, T1 and FIR only.

Proof: Exactly as before, except for the cases $s = a.p$ and $s = a^*p$. Note that the axiom T2 (used in the case $s = \tau.p$) is derivable from FIR, MI1 and T1.

Let $s = a.p$ with $a \neq \tau$. By induction I may assume that p is d-saturated. But then s is also d-saturated (and in head normal form).

Let $s = a^*p$ with $a \neq \tau$. By induction I assume that p is d-saturated and in head normal form. Now MI1 $\vdash s = a.a^*p + p$, which is d-saturated and in head normal form. □

Corollary 2 The axioms A0–3, MI1–2, T1, MT2 and FIR are complete for delay congruence.

Proof: As before. The derivations used in Proposition 2 only employ the axioms listed above. □

This axiomatization is also sound, as follows from inspection of the axioms.

Using the other saturation property of Definition 4, one can easily find a complete axiomatization for the notion of *η-bisimulation* [4, 16].

References

[1] L. Aceto, W.J. Fokkink, R.J. van Glabbeek & A. Ingólfsdóttir (1995): *Equational axiomatizations for prefix iteration with silent steps.* In preparation.

[2] L. Aceto & A. Ingólfsdóttir (1995): *A complete equational axiomatization for prefix iteration with silent steps.* Research Report RS–95–5, BRICS (Basic Research in Computer Science), Department of Mathematics and Computer Science, Aalborg University, Denmark. Available at `ftp://ftp.daimi.aau.dk/pub/BRICS/RS/95/5/BRICS-RS-95-5.dvi.gz`.

[3] G.J. Akkerman & J.C.M. Baeten (1990): *Term rewriting analysis in process algebra.* Report P9006, Programming Research Group, University of Amsterdam.

[4] J.C.M. Baeten & R.J. van Glabbeek (1987): *Another look at abstraction in process algebra.* In Th. Ottmann, editor: *Proceedings 14th ICALP, Karlsruhe*, LNCS 267, Springer-Verlag, pp. 84–94.

[5] T. Basten (1995): *Branching bisimulation is an equivalence indeed!* Unpublished manuscript.

[6] R. De Nicola, P. Inverardi & M. Nesi (1990): *Using axiomatic presentation of behavioural equivalences for manipulating CCS specifications.* In J. Sifakis, editor: *Automatic Verification Methods for Finite State Systems*, LNCS 407, Springer-Verlag, pp. 54–67.

[7] R. De Nicola, U. Montanari & F.W. Vaandrager (1990): *Back and forth bisimulations.* In J.C.M. Baeten & J.W. Klop, editors: *Proceedings CONCUR 90,* Amsterdam, LNCS 458, Springer-Verlag, pp. 152–165.

[8] R. De Nicola & F.W. Vaandrager (1995): *Three logics for branching bisimulation.* *Journal of the ACM* 42(2), pp. 458–487.

[9] W.J. Fokkink (1994): *A complete equational axiomatization for prefix iteration. Information Processing Letters* 52, pp. 333–337.

[10] W.J. Fokkink (1995): *A complete axiomatization for prefix iteration in branching bisimulation.* Logic Group Preprint Series 126, Utrecht University. To appear in *Fundamenta Informaticae*. Available at `ftp://ftp.phil.ruu.nl/pub/logic/PREPRINTS/preprint126.ps`.

[11] W.J. Fokkink (1995): Personal communication.

[12] R.J. van Glabbeek (1993): *A complete axiomatization for branching bisimulation congruence of finite-state behaviours.* In A.M. Borzyszkowski & S. Sokołowski, editors: Proceedings *Mathematical Foundations of Computer Science 1993 (MFCS),* Gdansk, Poland, August/September 1993, LNCS 711, Springer-Verlag, pp. 473–484. Available by anonymous ftp at `ftp://Boole.stanford.edu/pub/DVI/complete.dvi.gz`.

[13] R.J. van Glabbeek (1994): *What is branching time semantics and why to use it?* In M. Nielsen, editor: *The Concurrency Column*, *Bulletin of the EATCS* 53, pp. 190–198. Also available as Report STAN-CS-93-1486, Stanford University, 1993, and by ftp at `ftp://Boole.stanford.edu/pub/DVI/branching.dvi.gz`.

[14] R.J. van Glabbeek & W.P. Weijland (1989): *Branching time and abstraction in bisimulation semantics (extended abstract).* In G.X. Ritter, editor: *Information Processing 89,* Proceedings of the IFIP 11th World Computer Congress, San Fransisco 1989, North-Holland, pp. 613–618. Full version appeared as [16].

[15] R.J. van Glabbeek & W.P. Weijland (1989): *Refinement in branching time semantics.* Report CS-R8922, CWI, Amsterdam. Also appeared in: Proceedings AMAST Conference, May 1989, Iowa, USA, pp. 197–201.

[16] R.J. van Glabbeek & W.P. Weijland (1990): *Branching time and abstraction in bisimulation semantics.* Technical Report TUM-I9052, SFB-Bericht Nr. 342/29/90 A, Institut für Informatik, Technische Universität München, Germany. Extended abstract appeared as [14].

[17] R.J. van Glabbeek & W.P. Weijland (1995): *Branching time and abstraction in bisimulation semantics.* Revision of [16]. To appear in the *Journal of the ACM.*

[18] J.F. Groote & F.W. Vaandrager (1990): *An efficient algorithm for branching bisimulation and stuttering equivalence.* In M. Paterson, editor: *Proceedings* 17^{th} *ICALP,* Warwick, LNCS 443, Springer-Verlag, pp. 626–638.

[19] R. Milner (1980): *A Calculus of Communicating Systems*, LNCS 92. Springer-Verlag.

[20] R. Milner (1981): *A modal characterisation of observable machine-behaviour.* In G. Astesiano & C. Bohm, editors: *Proceedings CAAP 81*, LNCS 112, Springer-Verlag, pp. 25–34.