Axiomatizing Flat Iteration

R.J. van Glabbeek*
Computer Science Department, Stanford University
Stanford, CA 94305-9045, USA.
rvg@cs.stanford.edu

Flat iteration is a variation on the original binary version of the Kleene star operation P^*Q, obtained by restricting the first argument to be a sum of atomic actions. It generalizes prefix iteration, in which the first argument is a single action. Complete finite equational axiomatizations are given for five notions of bisimulation congruence over basic CCS with flat iteration, viz. strong congruence, branching congruence, η-congruence, delay congruence and weak congruence. Such axiomatizations were already known for prefix iteration and are known not to exist for general iteration. The use of flat iteration has two main advantages over prefix iteration:

1. The current axiomatizations generalize to full CCS, whereas the prefix iteration approach does not allow an elimination theorem for an asynchronous parallel composition operator.
2. The greater expressiveness of flat iteration allows for much shorter completeness proofs.

In the setting of prefix iteration, the most convenient way to obtain the completeness theorems for η-, delay, and weak congruence was by reduction to the completeness theorem for branching congruence. In the case of weak congruence this turned out to be much simpler than the only direct proof found. In the setting of flat iteration on the other hand, the completeness theorems for delay and weak (but not η-) congruence can equally well be obtained by reduction to the one for strong congruence, without using branching congruence as an intermediate step. Moreover, the completeness results for prefix iteration can be retrieved from those for flat iteration, thus obtaining a second indirect approach for proving completeness for delay and weak congruence in the setting of prefix iteration.

1 Introduction

The research literature on process theory has recently witnessed a resurgence of interest in Kleene star-like operations [6, 11, 9, 19, 8, 3, 10, 1, 2]. In [8] tree-based models for theories involving Kleene's star operation * [15] are studied. [6] investigates the expressive power of variations on standard process description languages in which infinite behaviours are defined by means of * rather than by means of systems of recursion equations. The papers [11, 19, 9, 3, 10, 1, 2] study

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

the possibility of giving finite equational axiomatizations of bisimulation-like equivalences over fragments of such languages. This study is continued here.

In [11] a complete finite equational axiomatization of strong bisimulation equivalence was given for a process algebra featuring choice, sequential composition, and the original binary version of the Kleene star operation P^*Q [15]. [19] shows that such an axiomatization does not exists in the presence of the process 0 denoting inaction, or a process denoting successful termination. The same proof strategy can be adopted to conclude that there is no finite equational axiomatization for weak or branching bisimulation over an enrichment of this basic process algebra with an internal action.

For this reason restrictions of the Kleene star have been investigated. [9] presents a finite, complete equational axiomatization of strong bisimulation equivalence for Basic CCS (the fragment of Milner's CCS [16] containing the operations needed to express finite synchronization trees) with *prefix iteration*. Prefix iteration is a variation on the binary Kleene star operation P^*Q, obtained by restricting the first argument to be an atomic action. The same is done in [1] for *string iteration*. The work of [9] has been extended in [2] and its predecessors [3, 10, 12] to cope with *weak*, *delay*, *eta*- and *branching bisimulation congruence* in a setting with the unobservable action τ. Motivation and background material on these behavioural congruences can be found, e.g., in [16] and [13]. The strategy adopted in [2] in establishing the completeness results is based upon the use of branching equivalence in the analysis of weak, delay and η-equivalence, advocated in [12]. Following [12], complete axiomatizations for weak, delay and η-congruence were obtained from one for branching congruence by:

1. identifying a collection of process terms on which branching congruence coincides with the congruence one aims at axiomatizing, and
2. finding an axiom system that allows for the reduction of every process term to one of the required form.

Perhaps surprisingly, the proof for weak congruence so obtained is simpler than the one given in [3] which only uses properties of weak congruence. The direct proof method employed in [3] yields a long proof with many case distinctions, while the indirect proof via branching congruence in [2] is considerably shorter, and relies on a general relationship between the two congruences. Moreover, attempts to obtain a direct proof of the completeness theorem for weak congruence which is simpler than the one presented in [3] have been to no avail.

Results The present paper extends the results from [9] and [2] from prefix iteration to *flat iteration*. Flat iteration was first mentioned in the technical report version of [6]; it allows the first argument P of P^*Q to be a (possibly empty) sum of actions. For convenience, the CCS operator of action-prefixing is also generalized to prefixing with sums of actions.

My completeness proofs are considerably shorter than the ones in [2]. This is mostly a result of the presence of expressions of the form 0^*P in the language, which allows a collapse of several cases in the case distinction in [2]. In addition, the results for weak and delay congruence can be obtained without using branching congruence as an intermediate step. Thanks to the greater expressiveness of flat iteration, these results can be reduced to the one for strong congruence, using the same proof strategy as outlined above. However, the proposed reduction to strong congruence does not work for η- and branching congruence.

In addition I derive the existing axiomatizations for prefix iteration from the ones for flat iteration. In the case of weak congruence one finds therefore that although a direct proof is cumbersome, there is a choice between two attractive indirect proofs. One of them involves first establishing the result for branching congruence; the other involves first establishing the result for a richer language.

Finally, extending a result from [6], I derive an expansion theorem for the CCS parallel composition operator in the setting of flat iteration. This is the key to extending the complete axiomatizations of this paper to full CCS. I show that such a theorem does not exist in the setting of prefix iteration.

As in [2], my completeness proofs apply to open terms directly, and thus yield the ω-completeness of the axiomatizations as well as their completeness for closed terms. However, the generalization to full CCS applies to closed terms only.

Outline of the paper Section 2 introduces the language of basic CCS with flat iteration, BCCS^{f*}, and its operational semantics. It also recalls the definitions of strong, branching, η-, delay and weak congruence. The axiom systems that will be shown to completely characterize the aforementioned congruences over BCCS^{f*} are presented in Section 3, and Section 4 contains the proofs of their completeness. In Section 5 the existing axiomatizations for prefix iteration are derived from the ones for flat iteration. Finally, Section 6 indicates how the completeness results of this paper, unlike the ones for prefix iteration, can, at least for closed terms, be extended to full CCS.

2 Basic CCS with Flat Iteration

Assume a set A of observable *actions*. Let $\tau \notin A$ denote a special *invisible action* and write $A_\tau := A \cup \{\tau\}$. Also assume an infinite set Var of *variables*, disjoint with A_τ. Let $x, y, ...$ range over Var, $a, b, ...$ over A, $\alpha, \beta, \gamma, ...$ over A_τ and ξ over $A_\tau \cup \mathsf{Var}$.

The two-sorted language BCCS^{f*} of basic CCS with flat iteration is given by the BNF grammar:

$$S ::= 0 \mid \alpha \mid S + S$$

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

Terms of sort S are called *sumforms*, whereas terms of sort P are called *process expressions*. The set of sumforms is denoted by SF and the set of (open) process expressions by $\mathbb{T}$. Let s, t, u range over SF and P, Q, R, S, T over $\mathbb{T}$. In writing terms over the above syntax one may leave out redundant brackets, assuming that $+$ binds weaker than . and *. For $I = \{i_1, \ldots, i_n\}$ a finite index set, $\sum_{i \in I} P_i$ or $\sum \{P_i \mid i \in I\}$ denotes $P_{i_1} + \cdots + P_{i_n}$. By convention, $\sum_{i \in \emptyset} P_i$ stands for 0.

The transition relations $\xrightarrow{\xi}$ are the least subsets of $(\mathbb{T} \times \mathbb{T}) \cup \mathrm{SF}$ satisfying the rules in Fig. 1. These determine the operational semantics of BCCS^{f*}. A transition $P \xrightarrow{\alpha} Q$ $(\alpha \in A_\tau)$ indicates that the system represented by the term P can perform the action α, thereby evolving into Q, whereas $P \xrightarrow{x} P'$ means that the initial behaviour of P may depend on the term that is substituted for the process variable x. It is not hard to see that if $P \xrightarrow{x} P'$ then $P' = x$. A transition $s \xrightarrow{\alpha}$ just says that α is one of the actions in the sumform s.

$$\frac{}{\alpha \xrightarrow{\alpha}} \qquad \frac{s \xrightarrow{\alpha}}{s+t \xrightarrow{\alpha}} \qquad \frac{t \xrightarrow{\alpha}}{s+t \xrightarrow{\alpha}}$$

$$\frac{}{x \xrightarrow{x} x} \qquad \frac{s \xrightarrow{\alpha}}{s.P \xrightarrow{\alpha} P} \qquad \frac{P \xrightarrow{\xi} P'}{P+Q \xrightarrow{\xi} P'} \qquad \frac{Q \xrightarrow{\xi} Q'}{P+Q \xrightarrow{\xi} Q'}$$

$$\frac{s \xrightarrow{\alpha}}{s^*P \xrightarrow{\alpha} s^*P} \qquad \frac{P \xrightarrow{\xi} P'}{s^*P \xrightarrow{\xi} P'}$$

Fig. 1: Transition rules for BCCS^{f*}

The set $\mathsf{der}(P)$ of *derivatives* of P is the least set containing P that is closed under action-transitions. Formally, $\mathsf{pder}(P)$ is the least set satisfying:

if $Q \in \{P\} \cup \mathsf{pder}(P)$ and $Q \xrightarrow{\alpha} Q'$ for some $\alpha \in A_\tau$, then $Q' \in \mathsf{pder}(P)$,

and $\mathsf{der}(P) = \{P\} \cup \mathsf{pder}(P)$. Members of $\mathsf{pder}(P)$ are called *proper* derivatives.

Definition 2.1 Write $p \Rightarrow q$ 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 $\xi \in A_\tau \cup \mathsf{Var}$, write $p \xrightarrow{(\xi)} q$ for $p \xrightarrow{\xi} q \vee (\xi = \tau \wedge p = q)$. Thus $\xrightarrow{(\xi)}$ is the same as $\xrightarrow{\xi}$ for $\xi \in A \cup \mathsf{Var}$, and $\xrightarrow{(\tau)}$ denotes zero or one τ-steps.

A *weak bisimulation* is a symmetric binary relation $\mathcal{R}$ on $\mathbb{T}$, such that

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

A weak bisimulation is a *delay bisimulation* if in the conclusion of (1) one has $t_2 = t'$. It is an *η-bisimulation* if one has $s\mathcal{R}t_1$, and it is a *branching bisimulation* if one has both $t_2 = t'$ and $s\mathcal{R}t_1$. Finally, it is a *strong bisimulation* if one has

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

Let s, w, d, b be abbreviations for *strong*, *weak*, *delay* and *branching*, and let $\aleph$ range over $\{s, w, d, \eta, b\}$. Then two processes $P, Q \in \mathbb{T}$ are $\aleph$*(-bisimulation) equivalent*—notation $P \leftrightarroweq_\aleph Q$—if there is a $\aleph$-bisimulation $\mathcal{R}$ with $P\mathcal{R}Q$.

Following [17, 2], the above definitions depart from the standard approach followed in, e.g., [16] in that notions of bisimulation equivalence are defined that apply to open terms directly. Usually, bisimulation equivalences like those presented in Def. 2.1 are defined explicitly for closed process expressions only. Open process expressions are then regarded equivalent iff they are equivalent under any closed substitution of their (free) variables. In [2] it has been shown, for the language BCCS with prefix iteration, that both approaches yield the same equivalence relation over open terms. The same proof applies to BCCS^{f*}. For this result it is essential that the set A of observable actions is nonempty.

The following lemma will be of use in the completeness proof for branching congruence (cf. the proof of Propn. 4.2). It is a standard result for branching bisimulation equivalence.

Lemma 2.2 (Stuttering Lemma [13]) *If $P_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} P_n$ and $P_n \leftrightarrow_b P_0$, then $P_i \leftrightarrow_b P_0$ for $i = 1, ..., n-1$.*

The definition of $\leftrightarrow_b$ is equivalent to the one in [13], as follows immediately from the proof of the stuttering lemma in [13]. However, what is here introduced as a branching bisimulation was there called a *semi branching bisimulation*, whereas "branching bisimulation" was the name of a slightly more restrictive type of relation. The advantages of the current setup have been pointed out in [5].

Proposition 2.3 *Each of the relations $\leftrightarrow_\aleph$ $(\aleph \in \{s, b, \eta, d, w\})$ is an equivalence relation and the largest $\aleph$-bisimulation. Furthermore, for all P, Q,*

$$\begin{array}{ccccc} P \leftrightarrow_s Q & \Rightarrow & P \leftrightarrow_b Q & \Rightarrow & P \leftrightarrow_d Q \\ & & \Downarrow & & \Downarrow \\ & & P \leftrightarrow_\eta Q & \Rightarrow & P \leftrightarrow_w Q. \end{array}$$

Proof: For $\aleph \in \{s, b, \eta, d, w\}$, the identity relation, the converse of a $\aleph$-bisimulation and the symmetric closure of the composition of two $\aleph$-bisimulations are all $\aleph$-bisimulations. Hence $\leftrightarrow_\aleph$ is an equivalence relation. As pointed out in [5], for this argument to apply to branching bisimulations it is essential that the definition of a branching bisimulation is relaxed to that of a semi branching bisimulation.

That $\leftrightarrow_\aleph$ is the largest $\aleph$-bisimulation follows immediately from the observation that the set of $\aleph$-bisimulations is closed under arbitrary unions. The implications hold by definition. $\square$

For $s, t \in \mathrm{SF}$ write $s \leq t$ if $\forall \alpha (s \xrightarrow{\alpha} \;\; \Rightarrow t \xrightarrow{\alpha})$, and $s \leftrightarrow t$ if $s \leq t$ and $t \leq s$. It is easily checked that $\leftrightarrow$ is a congruence on sumforms in the sense that

$$\text{if } s \leftrightarrow t \text{ then } \; s + u \leftrightarrow t + u, \;\; u + s \leftrightarrow u + t, \;\; s.P \leftrightarrow_s t.P, \;\text{ and }\; s^*P \leftrightarrow_s t^*P.$$

Likewise, $\leftrightarrow_s$ turns out to be a congruence on $\mathbb{T}$ in the sense that

$$\text{if } P \leftrightarrow_s Q \text{ then } \; P{+}R \leftrightarrow_s Q{+}R, \;\; R{+}P \leftrightarrow_s R{+}Q, \;\; s.P \leftrightarrow_s s.Q \;\text{ and }\; s^*P \leftrightarrow_s s^*Q.$$

However, for the standard reasons explained in, e.g., [16], none of the equivalences $\leftrightarrow_w$, $\leftrightarrow_d$, $\leftrightarrow_\eta$ and $\leftrightarrow_b$ is a congruence with respect to $+$. In fact, also none of these equivalences is preserved by $*$ [2]. Following Milner [16], the solution to these congruence problems is by now standard; it is sufficient to consider, for each equivalence $\leftrightarrow_\aleph$, the largest congruence over $\mathbb{T}$ contained in it. These largest congruences can be explicitly characterized as follows.

Definition 2.4

- P and Q are *branching congruent*, written $P \leftrightarrow_b^c Q$, iff for all $\xi \in A_\tau \cup \mathsf{Var}$,
  1. if $P \xrightarrow{\xi} P'$, then $Q \xrightarrow{\xi} Q'$ for some Q' such that $P' \leftrightarrow_b Q'$;
  2. if $Q \xrightarrow{\xi} Q'$, then $P \xrightarrow{\xi} P'$ for some P' such that $P' \leftrightarrow_b Q'$.
- P and Q are *η-congruent*, written $P \leftrightarrow_\eta^c Q$, iff for all $\xi \in A_\tau \cup \mathsf{Var}$,
  1. if $P \xrightarrow{\xi} P'$, then $Q \xrightarrow{\xi} Q_1 \Rightarrow Q'$ for some Q_1, Q' such that $P' \leftrightarrow_\eta Q'$;
  2. if $Q \xrightarrow{\xi} Q'$, then $P \xrightarrow{\xi} P_1 \Rightarrow P'$ for some P_1, P' such that $P' \leftrightarrow_\eta Q'$.
- P and Q are *delay congruent*, written $P \leftrightarrow_d^c Q$, iff for all $\xi \in A_\tau \cup \mathsf{Var}$,
  1. if $P \xrightarrow{\xi} P'$, then $Q \Rightarrow Q_1 \xrightarrow{\xi} Q'$ for some Q_1, Q' such that $P' \leftrightarrow_d Q'$;
  2. if $Q \xrightarrow{\xi} Q'$, then $P \Rightarrow P_1 \xrightarrow{\xi} P'$ for some P_1, P' such that $P' \leftrightarrow_d Q'$.
- P and Q are *weakly congruent*, written $P \leftrightarrow_w^c Q$, iff for all $\xi \in A_\tau \cup \mathsf{Var}$,
  1. if $P \xrightarrow{\xi} P'$, then $Q \Rightarrow\xrightarrow{\xi}\Rightarrow Q'$ for some Q' such that $P' \leftrightarrow_w Q'$;
  2. if $Q \xrightarrow{\xi} Q'$, then $P \Rightarrow\xrightarrow{\xi}\Rightarrow P'$ for some P' such that $P' \leftrightarrow_w Q'$.
- Finally, *strong congruence*, denoted $\leftrightarrow_s^c$, is the same as $\leftrightarrow_s$.

Proposition 2.5 *For every $\aleph \in \{s, b, \eta, d, w\}$, the relation $\leftrightarrow_\aleph^c$ is the largest congruence over $\mathbb{T}$ contained in $\leftrightarrow_\aleph$.*

Proof: Exactly as in [2]. □

3 Axiom Systems

Table 1 presents the axiom system $\mathcal{E}_s$, which will be shown to completely characterize strong congruence over BCCS^{f*}. The entries in this table are axiom schemes in the sense that there is one axiom for every choice of the sumforms s, t, u. For an axiom system $\mathcal{T}$, one writes $\mathcal{T} \vdash P = Q$ iff the equation $P = Q$ is provable from the axiom system $\mathcal{T}$ using the rules of equational logic. For a collection of equations X over the signature of BCCS^{f*}, $P \stackrel{\mathrm{X}}{=} Q$ is used as a short-hand for A1–A4,$X \vdash P = Q$. The axioms A1–4 are known to completely characterize the operator $+$ of CCS. As this operator occurs both in sumforms and in process expressions, these axioms appear for each of the two sorts. It is easily checked that they are sound and complete for $\leftrightarrow$ on sumforms:

Proposition 3.1 $s \leftrightarrow t \Leftrightarrow \mathrm{A1–4} \vdash s = t$. *Moreover,* $s \leq t \Leftrightarrow \mathrm{A1–4} \vdash t = t + s$.

The axioms A5 and A6 are inspired by the ACP axioms for sequential composition [7], and the axiom FA1 stems from [10], where a form of iteration P^*Q was used in which P had to be either an action, or a process (like 0) that cannot perform any actions. In [6] three axioms for general iteration in a process

A1	$x+y$	$=$	$y+x$	$s+t$	$=$	$t+s$
A2	$(x+y)+z$	$=$	$x+(y+z)$	$(s+t)+u$	$=$	$s+(t+u)$
A3	$x+x$	$=$	x	$s+s$	$=$	s
A4	$x+0$	$=$	x	$s+0$	$=$	s
A5			$(s+t).x$	$=$	$s.x+t.x$	
A6			$0.x$	$=$	0	
FA1			0^*x	$=$	x	
FA2			$s^*(t.(s+t)^*x+x)$	$=$	$(s+t)^*x$	

Table 1: The axiom system $\mathcal{E}_s$

algebra without 0 where proposed, called BKS1–3. These axioms where shown to be complete in [11]. The axiom BKS2 deals with the interaction between iteration and general sequential composition, and therefore has no counterpart in BCCS^{f*}. My axiom FA2 is obtained from BKS3 by requiring the first argument in an expression P^*Q to be a sumform. In the same spirit, the axiom BKS1 could be modified to $t.(t^*x)+x=t^*x$. This law is derivable from $\mathcal{E}_s$ by setting $s=0$ in FA2. The remaining axiom $a^*(a^*x)=a^*x$ of [9] is derivable as well: take $s=t$ in FA2 and apply BKS1 to the left-hand side.

In addition to the axioms in $\mathcal{E}_s$, the axiom systems $\mathcal{E}_\aleph$ ($\aleph \in \{b,\eta,d,w\}$) include equations describing the various ways in which the congruences $\leftrightarrow^c_\aleph$ abstract away from internal actions τ. These equations are presented in Table 2. The axiom system $\mathcal{E}_b$ is obtained by adding the axioms FT1–2 to $\mathcal{E}_s$, and $\mathcal{E}_\eta$ extends $\mathcal{E}_b$ with the equations T3 and FT3. The set of axioms $\mathcal{E}_d$ consists of the axioms of $\mathcal{E}_s$ together with T1 and FFIR. Finally, $\mathcal{E}_w$ extends $\mathcal{E}_d$ with T3 and FT3.

The equations T1 and T3 are standard laws for the silent action τ in weak congruence. Together with T2: $\tau.x=\tau.x+x$ and the laws for strong congruence,

$\mathcal{E}_\eta$, $\mathcal{E}_b$	FT1	$(s+\tau)^*x$	$=$	$\tau.(s^*x)+(s^*x)$		
$\mathcal{E}_\eta$, $\mathcal{E}_b$	FT2	$\alpha.s^*(\tau.s^*(x+y)+x)$	$=$	$\alpha.s^*(x+y)$		
$\mathcal{E}_\eta$	T3	$\alpha.(x+\tau.y)$	$=$	$\alpha.(x+\tau.y)+\alpha.y$		$\mathcal{E}_w$
$\mathcal{E}_\eta$	FT3	$s^*(x+\tau.y)$	$=$	$s^*(x+\tau.y+s.y)$		$\mathcal{E}_w$
	T1	$\alpha.\tau.x$	$=$	$\alpha.x$	$\mathcal{E}_d$	$\mathcal{E}_w$
	FFIR	$(s+\tau)^*x$	$=$	$\tau.(s^*x)$	$\mathcal{E}_d$	$\mathcal{E}_w$

Table 2: Extra axioms for $\mathcal{E}_\eta$, $\mathcal{E}_b$, $\mathcal{E}_d$ and $\mathcal{E}_w$

they are known to completely characterize weak congruence in the absence of iteration. Here T2 is derivable from $\mathcal{E}_s$ and FFIR (set $s = 0$ in FFIR and apply BKS1 on τ^*x). Also the law $\alpha.(\tau.(x+y)+x) = \alpha.(x+y)$, which together with the laws for strong congruence characterizes branching bisimulation for BCCS without iteration, is derivable: just take $s = 0$ in FT2.

The four remaining axioms, which describe the interplay between τ and prefix iteration, are new here. The law FFIR is a generalization of the *Fair Iteration Rule* $\tau^*x = \tau.x$ (FIR$_1$) of [6], which is an equational formulation of *Koomen's Fair Abstraction Rule* [4]. Like FIR, FFIR expresses that modulo weak (or delay) congruence a process remains the same if τ-loops are added (or deleted) in (or from) its proper derivatives. The law FT1 has the same function in branching (or η-)bisimulation semantics, but has to be formulated more carefully because T2 is not valid there. Note that FT1 can be reformulated as $\alpha.(s+\tau)^*x = \alpha.s^*x$. The laws FT2 and FT3 are straightforward generalizations of the laws PB2 and PT3 of [2]. The remaining law PT2 of [2] is (by the forthcoming completeness theorem for $\leftrightarrows^c_d$) derivable from the ones given here.

Note that even over a finite alphabet A there exist infinitely many sumforms. Hence the axiomatizations as given here are infinite. However, for each axiom scheme only the instantiations are needed in which the sumforms have the form $\sum_{i=1}^n \alpha_i$ in which all the α_i's are different. With this modification each of the axiom systems $\mathcal{E}_\aleph$ ($\aleph \in \{s, b, \eta, d, w\}$) is finite if so is the set of actions A. If A is not finite, the axiomatizations can still be interpreted as finite ones, namely by replacing the actions α in FT2 and T2,3 by sumforms t, introducing variables that range over sumforms, and interpreting each entry in the resulting Tables 1–4 as a single axiom in which s, t and u are such variables.

The following states the soundness of the axiom systems.

Proposition 3.2 *Let* $\aleph \in \{s, b, \eta, d, w\}$. *If* $\mathcal{E}_\aleph \vdash P = Q$, *then* $P \leftrightarrows^c_\aleph Q$.

Proof: As $\leftrightarrows^c_\aleph$ is a congruence, it is sufficient to show that each equation in $\mathcal{E}_\aleph$ is sound with respect to it. This is rather straightforward and left to the reader. □

As in [2], it can be shown that $\mathcal{E}_w \vdash \mathcal{E}_d \vdash \mathcal{E}_b \vdash \mathcal{E}_s$ and $\mathcal{E}_w \vdash \mathcal{E}_\eta \vdash \mathcal{E}_b$, where $\mathcal{T} \vdash \mathcal{T}'$ denotes that $\mathcal{T} \vdash P = Q$ for every equation $(P = Q) \in \mathcal{T}'$.

4 Completeness

This section is entirely devoted to detailed proofs of the completeness of the axiom systems $\mathcal{E}_\aleph$ ($\aleph \in \{s, b, \eta, d, w\}$) with respect to $\leftrightarrows^c_\aleph$ over the language of open terms $\mathbb{T}$. The first subsection contains the completeness proof for branching congruence. Its contents also apply to strong congruence if you read $\mathcal{E}_s$ for $\mathcal{E}_b$, $\leftrightarrows_s$ for $\leftrightarrows_b$, α for a, and α for (α) and skip the underlined and sidelined parts.

4.1 Completeness for strong <u>and branching</u> congruence

First I identify a subset of process expressions of a special form, which will be convenient in the proof of the completeness result. Following a long-established tradition in the literature on process theory, these terms are referred to as *normal forms*. The set of normal forms is the smallest set of process expressions of the form

$$s^*(\sum_{i\in I} \alpha_i.P_i + \sum_{j\in J} x_j),$$

where $s \stackrel{\tau}{\not\to}$, the terms P_i are themselves normal forms, and I, J are finite index sets. (Recall that the empty sum represents 0.)

Lemma 4.1 *Each term in* $\mathbb{T}$ *can be proven equal to a normal form using equations* A1–6*,* FA1,2 *and* FT1*.*

Proof: A straightforward induction on the structure of process expressions. The expressions x and 0 can be brought in the required form by a single application of FA1. Now suppose P and Q have the required form. Then $s.P$ can be brought in normal form using A5 or A6 (possibly after applying A4 on s), followed by FA1. $P+Q$ can be brought in normal form by first applying the derivable law $t^*x = t.(t^*x) + x$ (BKS1) on each of P and Q, then A4–6 to rewrite the subterms $t.(t^*x)$, and concluding with FA1. Finally s^*P is dealt with by applying BKS1 on P, again followed by A4–6. In case $s \xrightarrow{\tau}$, apply FT1, followed by another round of BKS1, A4–6 and FA1. □

Note that this is the only place in the completeness proof where the axioms FA1 and FT1 are used. The following result is the key to the completeness theorem.

Proposition 4.2 *For all* $P, Q \in \mathbb{T}$*, if* $P \leftrightarrow_b Q$*, then,* $\forall \gamma \in A_\tau : \mathcal{E}_b \vdash \underline{\gamma}.P = \underline{\gamma}.Q$*.*

Proof: First of all, note that, as the equations in $\mathcal{E}_b$ are sound with respect to $\leftrightarrow_b^c$, and, *a fortiori*, with respect to $\leftrightarrow_b$, by Lem. 4.1 it is sufficient to prove that the statement of the proposition holds for branching equivalent normal forms P and Q. I do so by complete induction on the sum of the sizes of P and Q.

Let $P = s^*(\sum_i \alpha_i.P_i + \sum_k x_k)$ and $Q = t^*(\sum_j \beta_j.Q_j + \sum_l y_l)$. Write P' for $\sum_i \alpha_i.P_i + \sum_k x_k$ and Q' for $\sum_j \beta_j.Q_j + \sum_l y_l$. Consider the following two conditions:

A. $P_i \leftrightarrow_b Q$ for some i;

B. $Q_j \leftrightarrow_b P$ for some j.

I distinguish two cases in the proof, depending on which of these conditions hold.

I Suppose that both of A and B hold. In this case, there exist i and j such that $P_i \leftrightarrow_b Q \leftrightarrow_b P \leftrightarrow_b Q_j$. Applying the inductive hypothesis to the equivalences $P \leftrightarrow_b Q_j$, $Q_j \leftrightarrow_b P_i$ and $P_i \leftrightarrow_b Q$, one infers that, for all $\gamma \in A_\tau$,

$$\mathcal{E}_b \vdash \underline{\gamma}.P = \underline{\gamma}.Q_j = \underline{\gamma}.P_i = \underline{\gamma}.Q$$

II Suppose that at most one of A and B holds. Assume, without loss of generality, that B does not hold.

Suppose $s \xrightarrow{a}$. As $P \leftrightarrow_b Q$, the transition $P \xrightarrow{a} P$ must be matched by a sequence of transitions $Q = Q_0 \xrightarrow{\tau} Q_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q_n \xrightarrow{a} Q''$ with $P \leftrightarrow_b Q_n$ and $P \leftrightarrow_b Q''$. As condition B does *not* hold, using Lem. 2.2 it follows that $n = 0$, $Q'' = Q$ and $t \xrightarrow{a}$. Hence $\mathcal{E}_b \vdash t = t + s$ by Prop. 3.1.

Let $u = \sum\left\{\alpha_i \mid P_i \leftrightarrow_b Q \wedge (t \stackrel{\alpha_i}{\rightarrow} \underline{\vee \alpha_i = \tau})\right\}$ and $v = \sum\left\{\alpha_i \mid P_i \leftrightarrow_b Q \wedge t \stackrel{\alpha_i}{\rightarrow}\right\}$. Then $\mathcal{E}_b \vdash t = t + v = t + s + v$.

For every summand $\alpha_i.P_i$ of P' with $P_i \leftrightarrow_b Q$, induction yields $\mathcal{E}_b \vdash \alpha_i.P_i = \alpha_i.Q$. Hence, using axiom A5 to assemble all such summands with $u \stackrel{\alpha_i}{\rightarrow}$, and possibly using A4 and/or A6 if there are no or only such summands, one infers that

$$\mathcal{E}_b \vdash P = s^*(u.Q + S)$$

where $S = \sum\left\{\alpha_i.P_i \mid P_i \not\leftrightarrow_b Q \vee (t \stackrel{\alpha_i}{\not\rightarrow} \underline{\wedge \alpha_i \neq \tau})\right\} + \sum_k x_k$.

Consider now a summand $\alpha_i.P_i$ of S. As $P \leftrightarrow_b Q$, the transition $P \stackrel{\alpha_i}{\rightarrow} P_i$ must be matched by a sequence $Q \underline{= Q_0 \stackrel{\tau}{\rightarrow} Q_1 \stackrel{\tau}{\rightarrow} \cdots \stackrel{\tau}{\rightarrow} Q_n} \stackrel{(\alpha_i)}{\rightarrow} Q''$ with $\underline{P \leftrightarrow_b Q_n \text{ and}}$ $P_i \leftrightarrow_b Q''$. $\underline{\text{As condition B does not hold, using Lem. 2.2 it follows that } n = 0.}$ Furthermore, the possibility $Q \stackrel{(\alpha_i)}{\rightarrow} Q \leftrightarrow_b P_i$ is ruled out by the construction of S. Hence, each summand $\alpha_i.P_i$ of S matches with a summand $\beta_j.Q_j$ of Q', in the sense that $\alpha_i = \beta_j$ and $P_i \leftrightarrow_b Q_j$. For each such pair of related summands, induction yields

$$\mathcal{E}_b \vdash \alpha_i.P_i = \alpha_i.Q_j = \beta_j.Q_j \ .$$

Moreover, each summand x_k of S must be a summand of Q'. Hence, possibly using axiom A3, it follows that $\mathcal{E}_b \vdash Q' = Q' + S$. <u>Now I distinguish two sub-cases.</u>

IIa <u>Suppose that A does not hold for an index i with $\alpha_i = \tau$.</u> Again <u>using Lem. 2.2,</u> it follows that every summand $\beta_j.Q_j$ of Q' matches with a summand $\alpha_i.P_i$ of S (since also B does not hold, the cases $Q_j \leftrightarrow_b P$ and $Q_j \leftrightarrow_b Q \leftrightarrow_b P$ do not apply), and every y_l is equal to an x_k. Possibly using axiom A3, it follows that $\mathcal{E}_b \vdash S = Q' + S = Q'$. Moreover, whenever $t \stackrel{a}{\rightarrow}$ then $Q \stackrel{a}{\rightarrow} Q$, so $P \stackrel{a}{\rightarrow} P'' \leftrightarrow_b Q$ and hence either $s \stackrel{a}{\rightarrow}$ or $v \stackrel{a}{\rightarrow}$. It follows that $\mathcal{E}_b \vdash t = s + v$. Finally $u = v$, so

$$\underline{\gamma}.P = \underline{\gamma}.s^*(v.Q + S) = \underline{\gamma}.s^*(v.(s+v)^*S + S) \stackrel{\text{FA2}}{=} \underline{\gamma}.(s+v)^*S = \underline{\gamma}.Q.$$

IIb Suppose that A holds for an index i with $\alpha_i = \tau$. Then $\mathcal{E}_b \vdash u = \tau + v$, so

$$\begin{array}{rcl}
\gamma.P & \stackrel{\text{A5}}{=} & \gamma.s^*(\tau.Q + v.Q + S) \\
& = & \gamma.s^*\Big(\tau.t^*\big(Q' + S\big) + v.t^*\big(Q' + S\big) + S\Big) \\
& \stackrel{\text{FA2}}{=} & \gamma.s^*\Bigg(\tau.s^*\Big(t.(s+t)^*\big(Q' + S\big) + Q' + S\Big) + v.t^*\big(Q' + S\big) + S\Bigg) \\
& = & \gamma.s^*\Bigg(\tau.s^*\Big(Q' + (t+v).t^*\big(Q' + S\big) + S\Big) + v.t^*\big(Q' + S\big) + S\Bigg) \\
& \stackrel{\text{FT2, A5}}{=} & \gamma.s^*\Big(Q' + (t+v).t^*\big(Q' + S\big) + S\Big) \\
& \stackrel{\text{FA2}}{=} & \gamma.t^*(Q' + S) \ = \ \gamma.Q.
\end{array}$$

The proof of the inductive step is now complete. □

Theorem 4.3 *Let $P, Q \in \mathbb{T}$. If $P \leftrightarrow_b^c Q$, then $\mathcal{E}_b \vdash P = Q$.*

Proof: Consider two process expressions P and Q that are branching congruent. Using the same technique as in the proof of Lem. 4.1, one may derive that

$$\begin{array}{l}
\mathcal{E}_b \vdash P = \sum\{\alpha_i.P_i \mid i \in I\} + \sum\{x_j \mid j \in J\} \quad \text{and} \\
\mathcal{E}_b \vdash Q = \sum\{\beta_k.Q_k \mid k \in K\} + \sum\{y_l \mid l \in L\}
\end{array}$$

for some finite index sets I, J, K, L. As $P \leftrightarrow_b^c Q$, it follows that

1. for every $i \in I$ there exists an index $k_i \in K$ such that $\alpha_i = \beta_{k_i}$ and $P_i \leftrightarrow_b Q_{k_i}$,
2. and for every $j \in J$ there exists an index $l_j \in L$ such that $x_j = y_{l_j}$.

By Propn. 4.2, for every $i \in I$ one may infer that

$$\mathcal{E}_b \vdash \alpha_i.P_i = \alpha_i.Q_{k_i} = \beta_{k_i}.Q_{k_i} \quad .$$

Using A3 it follows immediately that $\mathcal{E}_b \vdash Q = P + Q$. By symmetry one obtains $\mathcal{E}_b \vdash P = P + Q = Q$. □

4.2 Completeness for η-, delay, and weak congruence

I now proceed to derive completeness results for η-, delay, and weak congruence from the ones for strong and branching congruence. The key to this derivation is the observation that, for certain classes of process expressions, these congruence relations coincide with $\leftrightarrow_s$ or $\leftrightarrow_b^c$. These classes of process expressions are defined below.

Definition 4.4 A term P is:

- *η-saturated* iff for each of its derivatives Q, R and S and $\xi \in A_\tau \cup \mathsf{Var}$ one has that:
$$Q \xrightarrow{\xi} R \xrightarrow{\tau} S \text{ implies } Q \xrightarrow{\xi} S.$$
- *d-saturated* iff for each of its derivatives Q, R and S and $\xi \in A_\tau \cup \mathsf{Var}$ one has that:
$$Q \xrightarrow{\tau} R \xrightarrow{\xi} S \text{ implies } Q \xrightarrow{\xi} S.$$
- *w-saturated* iff it is both η- and d-saturated.
- *strongly $\aleph$-saturated* (for $\aleph \in \{\eta, d, w\}$) if it is $\aleph$-saturated and for each of its proper derivatives $Q \in \mathsf{pder}(P)$ there is a *τ-loop* $Q \xrightarrow{\tau} Q$.

The following was first observed in [13] for process graphs.

Theorem 4.5

1. *If P and Q are $\aleph$-saturated, $\aleph \in \{\eta, d, w\}$, and $P \leftrightarrow_\aleph^c Q$, then $P \leftrightarrow_b^c Q$.*
2. *If P and Q are strongly $\aleph$-saturated, $\aleph \in \{d, w\}$, and $P \leftrightarrow_\aleph^c Q$, then $P \leftrightarrow_s Q$.*

Proof: In case 1, the relation

$$\mathcal{B} \stackrel{\text{def}}{=} \{(S,T) \mid S \leftrightarrow_\aleph T, \quad S, T \ \aleph\text{-saturated}\}$$

is a branching bisimulation. From this it follows easily (as shown in [2]) that $P \leftrightarrow_\aleph^c Q$ implies $P \leftrightarrow_b^c Q$. In case 2, $\mathcal{B}$ is a strong bisimulation. □

Note that the second statement does not apply to $\leftrightarrow_\eta^c$. A counterexample concerns the terms $P = a.\tau^*\tau.\tau^*b.\tau^*0 + a.\tau^*b.\tau^*0$ and $Q = a.\tau^*b.\tau^*0$. These terms are strongly η-saturated and $P \leftrightarrow_\eta^c Q$, but $P \not\leftrightarrow_s Q$.

Theorem 4.6 *Let $\aleph \in \{\eta, d, w\}$.*

1. *For each term P, $\mathcal{E}_\aleph \vdash P = P'$ for some $\aleph$-saturated term P'.*
2. *For each term P, $\mathcal{E}_\aleph \vdash P = P''$ for some strongly $\aleph$-saturated term P''.*

Proof: The first statement has been shown in [2] for the language BCCS^{p*}. The resulting term P' has the form $P' = \sum_{i \in I} \alpha_i.P_i + \sum_{j \in J} x_j$. The same proof applies here.

For the second result, first prove P equal to a term P' as above, and bring the subterms P_i for $i \in I$ in normal form, using Lem. 4.1. Now each proper derivative of the resulting term has the form s^*Q, and appears in a subterm of the form $\alpha.s^*Q$. In combination with T1, the axiom FFIR derives $\alpha.s^*x = \alpha.(s+\tau)^*x$. As mentioned before, this law is also derivable from $\mathcal{E}_b$. Applying this law to all subterms of the form $\alpha.s^*Q$ results in a term P'' that is still $\aleph$-saturated, and for which each proper derivative Q has a τ-loop $Q \xrightarrow{\tau} Q$. □

The results in Thms. 4.5.1 and 4.6.1 effectively reduce the completeness problem for η-, delay, and weak congruence over $\mathbb{T}$ to that for branching congruence.

Corollary 4.7 *Let $\aleph \in \{\eta, d, w\}$. If $P \leftrightarrow^c_\aleph Q$, then $\mathcal{E}_\aleph \vdash P = Q$.*

Proof (for the case $\aleph = \eta$): Suppose that $P \leftrightarrow^c_\aleph Q$. Prove P and Q equal to $\aleph$-saturated processes P' and Q', respectively (Thm. 4.6.1). By the soundness of the axiom system $\mathcal{E}_\aleph$ (Propn. 3.2), P' and Q' are $\aleph$-congruent. It follows that P' and Q' are branching congruent (Thm. 4.5.1). Hence, by Thm. 4.3, $\mathcal{E}_b \vdash P' = Q'$. The claim now follows because $\mathcal{E}_b \subset \mathcal{E}_\eta$. □

The cases $\aleph = d$ and $\aleph = w$ can be proved in the same way, using in the last step that $\mathcal{E}_w \vdash \mathcal{E}_d \vdash \mathcal{E}_b \vdash P = Q$ (cf. the last sentence of Section 3). However, Thms. 4.5.2 and 4.6.2 allow a simpler proof that doesn't need the completeness result for branching bisimulation as an intermediate step, but instead reduces the problem to the completeness for strong congruence.

Proof of Corollary 4.7 (for the cases $\aleph \in \{d, w\}$): Suppose that $P \leftrightarrow^c_\aleph Q$. Prove P and Q equal to strongly $\aleph$-saturated processes P' and Q', respectively (Thm. 4.6.2). By the soundness of the axiom system $\mathcal{E}_\aleph$ (Propn. 3.2), P' and Q' are $\aleph$-congruent. It follows that P' and Q' are strong congruent (Thm. 4.5.2). Hence, by Prop. 4.2 for strong congruence, $\mathcal{E}_s \vdash P' = Q'$. The claim now follows because $\mathcal{E}_s \subset \mathcal{E}_\aleph$. □

5 Prefix Iteration

In this section I derive complete axiomatizations for prefix iteration from the ones for flat iteration. A BCCS^{f*} process expression is a BCCS^{p*} expression iff in each subexpression $s.P$ or s^*P, the sumform s consists of a single action $\alpha \in A_\tau$. The following result about the expressiveness of BCCS^{p*} stems from [3].

Lemma 5.1 *If P_0 is a* BCCS^{p*} *expression and* $P_n \Rightarrow\xrightarrow{a_n} P_{n+1}$ *for* $n = 0, 1, 2, \ldots$, *then there is an N such that* $a_n = a_N$ *for* $n > N$.

Definition 5.2 A BCCS^{f*} expression P_0 is a *potential* BCCS^{p*} *expression* if every sequence $P_n \Rightarrow\xrightarrow{a_n} P_{n+1}$ ($n = 0, 1, 2, \ldots$) has the property of Lem. 5.1.

It is easy to see that a potential BCCS^{p*} expression can not be weakly equivalent to an expression that is not so. Hence, using Propn. 3.2 (soundness):

Lemma 5.3 *Let* $\aleph \in \{s, b, \eta, d, w\}$. *If* $\mathcal{E}_\aleph \vdash P = Q$ *then either both P and Q are potential* BCCS^{p*} *expressions, or neither of them is.*

Using structural induction, the following Lemma is straightforward:

Lemma 5.4 *If s^*P is a subterm of a potential* BCCS^{p*} *expression, then either* A1–4 $\vdash s = 0$ *or* A1–4 $\vdash s = \alpha \in A_\tau$ *or* A1–4 $\vdash s = a + \tau$ *with* $a \in A$. *Moreover, these alternatives are mutually exclusive.*

Let R be the rewrite system consisting of the axioms A5, A6, FA1 and FT1, read from left to right. As these rewrite rules have no overlapping redexes, R is confluent, and it is equally straightforward to see that it is terminating. Now let φ be the operator on potential BCCS^{p*} expressions P that first converts any sumform s in a subterm s^*Q of P into one of the forms 0, α or $a + \tau$ (using A1–4 and Lem. 5.4), and subsequently brings the resulting term in normal form w.r.t. R. Note that the resulting term $\varphi(P)$ is a BCCS^{p*} expression.

Theorem 5.5 *Let* $\aleph \in \{b, \eta, d, w\}$. *The theory*

$$\varphi(\mathcal{E}_\aleph) = \{\varphi(P) = \varphi(Q) \mid (P = Q) \in \mathcal{E}_\aleph\}$$

is a complete axiomatization of $\leftrightarrow_\aleph^c$ *over the language* BCCS^{p*}.

Proof: An equation $P = Q$ is provable in equational logic iff there exists a sequence $T_0, \ldots, T_n$ with $P = T_0$, $Q = T_n$, and the equation $T_{i-1} = T_i$ is obtained from one axiom by means of substitution, placement in a context and (possibly) symmetry ($i = 1, \ldots, n$). Suppose that $P \leftrightarrow_\aleph^c Q$ for certain BCCS^{p*} expressions P and Q. As P and Q are also BCCS^{f*} expressions, this implies $\mathcal{E}_\aleph \vdash P = Q$. Thus, by Lem. 5.3, there exists a proof-sequence as mentioned above in which all the T_i are potential BCCS^{p*} expressions. Now, for $i = 1, \ldots, n$, the equation $\varphi(T_{i-1}) = \varphi(T_i)$ can be obtained from an axiom in $\varphi(\mathcal{E}_\aleph)$ by means of substitution, placement in a context and symmetry. This yields a proof-sequence for the equation $\varphi(P) = \varphi(Q)$. However, since P and Q are BCCS^{p*} expressions, $\varphi(P) = P$ and $\varphi(Q) = Q$. Hence $\varphi(\mathcal{E}_\aleph) \vdash P = Q$. $\square$

In the axiom systems $\varphi(\mathcal{E}_\aleph)$, the axioms A5, A6 and FA1 evaluate to identities, whereas the axioms A1–4, T1 and T3 remain unchanged. Furthermore, there are three axioms corresponding to each of FT1–3 and FFIR, depending on whether s evaluates to 0, α, or $a + \tau$, and nine corresponding to FA2, depending on how s and t evaluate. All resulting axiomatizations turn out to be derivable from the corresponding axiomatizations in [2] and vice versa. Hence the above constitutes an alternative proof of the completeness results in [2].

A similar result can be obtained for $\aleph = s$, but in that case τ should be treated as a normal action, and FT1 should be omitted from the rewrite system.

6 Parallelism

Complete axiomatizations of strong and weak bisimulation congruence over full CCS without recursion or iteration were given in [14]. The strategy, in both cases, was to prove every such CCS expression strongly equivalent to a BCCS expression, using the well known *expansion theorem*, and then apply the relevant completeness theorem for BCCS expressions. This method does not work in the setting of prefix iteration, as the parallel composition of two BCCS^{p*} expressions need not be (weakly) equivalent to a BCCS^{p*} expression. A simple counterexample concerns the expression $a^*0 \mid b^*0$, which is not a potential BCCS^{p*} expression in the sense of Def. 5.2. However, an expansion theorem for CCS^{f*} poses no problem: let $P = s^* \sum_{i\in I} \alpha_i.P_i$ and $Q = t^* \sum_{j\in J} \beta_j.Q_j$, then

$$P \mid Q \leftrightarrow_s (s+t+\gamma)^* \left(\sum_{i\in I} \alpha_i.(P_i \mid Q) + \sum_{j\in J} \beta_j.(P \mid Q_j) + C \right)$$

with

$$C = \sum_{\alpha_i=\overline{\beta_j}} \tau.(P_i \mid Q_j) + \sum_{i\in I,\ t\overset{\overline{\alpha_i}}{\to}} \tau.(P_i \mid Q) + \sum_{j\in J,\ s\overset{\overline{\beta_j}}{\to}} \tau.(P \mid Q_j)$$

and $\gamma = \begin{cases} \tau & \text{if there is an } a \in A \text{ with } s \overset{a}{\to} \text{ and } t \overset{\bar{a}}{\to} \\ 0 & \text{otherwise.} \end{cases}$

For a parallel composition without communication just leave out γ and C; in this shape the theorem was first found in [6].

In the presence of a CSP-style parallel composition in which processes are forced to synchronize over a shared alphabet [18], closed expressions with flat iteration can be expressed in terms of prefix iteration. An expression

$$(a+b)^*(c.P + d.Q)$$

for instance, in which c and d do not occur in P and Q, is strongly equivalent to

$$a^*(c.0 + d.0)\|_{\{c,d\}} b^*(c.P + d.Q)$$

where synchronization over c and d is enforced. In the general case renaming operators are needed as well.

Acknowledgment The inspiration to write this paper originated from the fruitful collaboration with Luca Aceto, Wan Fokkink and Anna Ingólfsdóttir in [2]. The referees are thanked for careful proofreading, and correcting the expansion theorem above.

References

[1] L. Aceto and J.F. Groote (1995), *A complete equational axiomatization for MPA with string iteration*, BRICS Research Report RS-95-28, Department of Mathematics and Computer Science, Aalborg University. Available by anonymous ftp from `ftp.daimi.aau.dk` in the directory `pub/BRICS/RS/95/28`.

[2] L. Aceto, W. Fokkink, R. van Glabbeek and A. Ingólfsdóttir (1996), *Axiomatizing prefix iteration with silent steps*, I&C 127(1), pp. 26–40.

[3] L. Aceto and A. Ingólfsdóttir (1996), *An equational axiomatization of observation congruence for prefix iteration*, in Proc. AMAST '96, Munich, Germany, M. Wirsing and M. Nivat, eds., LNCS 1101, Springer-Verlag, pp. 195–209.

[4] J. Baeten, J. Bergstra and J. Klop (1987), *On the consistency of Koomen's fair abstraction rule*, TCS 51, pp. 129–176.

[5] T. Basten (1996), *Branching bisimilarity is an equivalence indeed!*, IPL 58(3), pp. 141–147.

[6] J. Bergstra, I. Bethke and A. Ponse (1994), *Process algebra with iteration and nesting*, Computer Journal 37, pp. 243–258. Originally appeared as report P9314, Programming Research Group, University of Amsterdam, 1993.

[7] J. Bergstra and J. Klop (1984), *The algebra of recursively defined processes and the algebra of regular processes*, in Proceedings 11^{th} ICALP, Antwerpen, J. Paredaens, ed., LNCS 172, Springer-Verlag, pp. 82–95.

[8] F. Corradini, R. De Nicola and A. Labella (1995), *Fully abstract models for nondeterministic Kleene algebras (extended abstract)*, in Proc. CONCUR 95, Philadelphia, I. Lee and S. Smolka, eds., LNCS 962, Springer-Verlag, pp. 130–144.

[9] W. Fokkink (1994), *A complete equational axiomatization for prefix iteration*, IPL 52, pp. 333–337.

[10] W. Fokkink (1996), *A complete axiomatization for prefix iteration in branching bisimulation*, Fundamenta Informaticae 26, pp. 103–113.

[11] W. Fokkink and H. Zantema (1994), *Basic process algebra with iteration: Completeness of its equational axioms*, Computer Journal 37, pp. 259–267.

[12] R. v. Glabbeek (1995), *Branching bisimulation as a tool in the analysis of weak bisimulation.* Available at `ftp://boole.stanford.edu/pub/DVI/tool.dvi.gz`.

[13] R. v. Glabbeek and W. Weijland (1996), *Branching time and abstraction in bisimulation semantics*, JACM 43(3), pp. 555–600.

[14] M. Hennessy and R. Milner (1985), *Algebraic laws for nondeterminism and concurrency*, JACM 32, pp. 137–161.

[15] S. Kleene (1956), *Representation of events in nerve nets and finite automata*, in Automata Studies, C. Shannon and J. McCarthy, eds., Princeton University Press, pp. 3–41.

[16] R. Milner (1989), *Communication and Concurrency*, Prentice-Hall.

[17] R. Milner (1989), *A complete axiomatisation for observational congruence of finite-state behaviours*, I&C 81, pp. 227–247.

[18] E.-R. Olderog and C.A.R. Hoare (1986), *Specification-oriented semantics for communicating processes*, Acta Informatica 23, pp. 9–66.

[19] P. Sewell (1994), *Bisimulation is not finitely (first order) equationally axiomatisable*, in Proc. 9^{th} LICS, Paris, IEEE Computer Society Press, pp. 62–70.