On Cool Congruence Formats for Weak Bisimulations[*]

R.J. van Glabbeek
National ICT Australia
and School of Computer Science and Engineering
The University of New South Wales
rvg@cs.stanford.edu

In TCS 146, Bard Bloom presented rule formats for four main notions of bisimulation with silent moves. He proved that weak bisimulation equivalence is a congruence for any process algebra defined by *WB cool rules*, and established similar results for rooted weak bisimulation (Milner's "observational congruence"), branching bisimulation and rooted branching bisimulation. This study reformulates Bloom's results in a more accessible form and contributes analogues for (rooted) η-bisimulation and (rooted) delay bisimulation. Moreover, finite equational axiomatisations of rooted weak bisimulation equivalence are provided that are sound and complete for finite processes in any RWB cool process algebra. These require the introduction of auxiliary operators with lookahead, and an extension of Bloom's formats for this type of operator with lookahead. Finally, a challenge is presented for which Bloom's formats fall short and further improvement is called for.

1 Introduction

Structural Operational Semantics [13, 16] is one of the main methods for defining the meaning of operators in system description languages like CCS [13]. A system behaviour, or *process*, is represented by a closed term built from a collection of operators, and the behaviour of a process is given by its collection of (outgoing) transitions, each specifying the action the process performs by taking this transition, and the process that results after doing so. For each n-ary operator f in the language, a number of *transition rules* are specified that generate the transitions of a term $f(p_1,\ldots,p_n)$ from the transitions (or the absence thereof) of its arguments $p_1,\ldots,p_n$.

For purposes of representation and verification, several behavioural equivalence relations have been defined on processes, of which the most well-known is *strong bisimulation equivalence* [13], and its variants *weak* and *branching* bisimulation equivalence [13, 12], that feature abstraction from internal actions. In order to allow compositional system verification, such equivalence relations need to be *congruences* for the operators under consideration, meaning that the equivalence class of an n-ary operator f applied to arguments $p_1,\ldots,p_n$ is completely determined by the equivalence classes of these arguments. Although strong bisimulation equivalence is a congruence for the operators of CCS and many other languages found in the literature, weak bisimulation equivalence fails to be a congruence for the *choice* or *alternative composition* operator + of CCS. To bypass this problem one uses the coarsest congruence relation for + that is finer than weak bisimulation equivalence, characterised as *rooted weak bisimulation equivalence* [13, 4, 11], which turns out to be a minor variation of weak bisimulation equivalence, and a congruence for all of CCS and many

[*]An extended abstract of this paper appeared in: D.V. Hung & M. Wirsing, editors: Proceedings International Colloquium on Theoretical Aspects of Computing, ICTAC'05, Hanoi, LNCS 3722, Springer, 2005, pp. 318-333.

other languages. Analogously, *rooted branching bisimulation* is the coarsest congruence for CCS and many other languages that is finer than branching bisimulation equivalence [12].

In order to streamline the process of proving that a certain equivalence is a congruence for certain operators, and to guide sensible language definitions, syntactic criteria (*rule formats*) for the transition rules in structural operational semantics have been developed, ensuring that the equivalence is a congruence for any operator specified by rules that meet these criteria. One of these is the *GSOS format* of Bloom, Istrail & Meyer [6], generalising an earlier format by De Simone [17]. When adhering to this format, all processes are computably finitely branching, and strong bisimulation equivalence is a congruence [6]. Bloom [5] defines congruence formats for (rooted) weak and branching bisimulation equivalence by imposing additional restrictions on the GSOS format. As is customary in this field, finer equivalences have wider formats, so Bloom's *BB cool* GSOS format, which guarantees that branching bisimulation equivalence is a congruence, is more general than his *WB cool* GSOS format, which suits weak bisimulation equivalence; also his *RWB cool* GSOS format, suiting rooted weak bisimulation, is more general than the WB cool GSOS format, and his *RBB cool* GSOS format, guaranteeing that rooted branching bisimulation equivalence is a congruence, is the most general of all. The prime motivating example for these formats is the structural operational semantics of CCS [13]. All CCS operators are RWB cool, and the CCS operators other than the $+$ are even WB cool.

Bloom's formats involve a fast bookkeeping effort of names of variables, used to precisely formulate the *bifurcation rules* that his formats require. To make his work more accessible, Bloom also presents simpler but less general versions of his formats, obtained by imposing an additional syntactic restriction. This restriction makes it possible to simplify the bifurcation rules to *patience rules*, which do not require such an extensive bookkeeping. Fokkink [8] generalises Bloom's *simply RBB cool* format to a format he calls *RBB safe*, and writes "The definition of bifurcation rules is deplorably complicated, and we do not know of any examples from the literature that are RBB cool but not simply RBB cool. Therefore, we refrain from this generalisation here." Ulidowski [18, 19, 20] studies congruence formats for variations of the semantic equivalences mentioned above with a different treatment of divergence. Ulidowski's formats form the counterparts of Bloom's simply cool formats only.

The main aim of the present study is to simplify and further clarify Bloom's work, so as to make it more accessible for the development of applications, variations and extensions. In passing, analogous results are obtained for two equivalences, and their rooted variants, that bridge the gap between weak and branching bisimulation. Moreover, the method of Aceto, Bloom & Vaandrager [1] to extract from any GSOS language a finite equational axiomatisation that is sound and complete for strong bisimulation equivalence on finite processes, is adapted to rooted weak bisimulation equivalence. In the construction fresh function symbols may need to be added whose transition rules have *lookahead* and thereby fall outside the GSOS format. To this end, I extend the RWB cool format with a form of lookahead.

One of the simplifications of Bloom's formats presented here stems from the observation that the operators in any of the cool formats can be partitioned in *principal operators* and *abbreviations*, such that the abbreviations can be regarded as syntactic sugar, adding nothing that could not be expressed with principal operators. For any abbreviation f there exists a principal operator $f^\star$ that typically takes more arguments. For instance, $f(x_1, x_2)$ could be an abbreviation of $f^\star(x_1, x_1, x_2)$. The rules for the abbreviations are completely determined by the rules for the principal operators, and for principal operators patience rules suffice, i.e. one does not need the full generality of bifurcation rules. Moreover, the simply cool formats can be characterised by the requirement that all operators are principal. These observations make it possible to define the cool formats of Bloom without mentioning bifurcation rules altogether. It also enables a drastic simplification of the congruence proofs, namely by establishing the congruence results for the simply cool formats first, and reducing the general case to the simple case by means of some general insights in abbreviation expansion.

In fact, I will present a general algorithm that given any rule format, guaranteeing that a certain equivalence, coarser than or equal to strong bisimulation equivalence, is a congruence, extends this format with a mechanism for abbreviation expansion. Bloom's general formats can then be obtained by applying this algorithm to his simple formats.

Even though any operation that fits the cool formats can also be defined using merely the simply cool formats, in practice it may be handy to work with the full generality of the cool formats. The unary copying operator cp of [6] (page 257) for instance does not fit the cool formats directly, but can be made to fit by adding an auxiliary binary copying operator to the language, of which the unary one is an abbreviation. Dumping the abbreviation from the language would appear unnatural here, as the unary operator motivates the rules for both itself and its binary expansion, the latter being needed merely to make it work.

Another simplification contributed here is in the description of the RWB cool format. Bloom requires for every operational rule with target t the existence of two terms t_1 and t_2, and seven types of derived operational rules. I show that without limitation of generality it is always possible to choose $t_2 = t$, thereby making four of those seven types of rules redundant. Thus, the same format is obtained by requiring only t_1 and two types of derived rules (the third being a bifurcation rule, that was already required for its own sake).

After defining the basic concepts in Section 2, I present the congruence proofs for the simply cool languages in Section 3. To that end, I first prove the result from [6] that strong bisimulation equivalence is a congruence for all GSOS operators. Then I change this proof in minimal ways so as to obtain the congruence results for the simply cool languages. In doing so, for each simply cool format I state a lemma that tells exactly what is needed to make the congruence proof work. Section 4 defines the simply cool formats for the unrooted case, in such a way that the lemmas hold. As an example I show a version of the language CSP [7, 15] to be simply WB cool, implying that weak and branching bisimulation equivalence are congruences on this language. Section 5 presents the theory of abbreviations that lifts the results from the simple to the general congruence formats, and Section 6 deals with the rooted congruence formats. Section 7 compares my definitions of the cool formats with the ones of Bloom. In Section 8 I generalise the RWB cool format to what I call *GSOS languages with lookahead*. Section 9 recapitulates the method of [1] to provide finite equational axiomatisations of strong bisimulation equivalence that are sound and complete for finite processes on an augmentation of any given GSOS language, and Section 10 extends this work to the rooted weak equivalences. Section 11 discusses extensions of Bloom's formats that occur in the literature. Finally, Section 12 presents a fairly intuitive GSOS language for which the existing congruence formats fall short and further improvement is called for.

Acknowledgements. My thanks to Simone Tini for inspiration, and to the referees for their proposals of improvements in presentation.

2 Preliminaries

In this paper $V = \{x_1, x_2, \ldots\}$ and *Act* are two sets of *variables* and *actions*.

Definition 1 A *signature* is a collection Σ of *function symbols* $f \notin V$ equipped with a function $ar : \Sigma \rightarrow \mathbb{N}$. The set $\mathbb{T}(\Sigma)$ of *terms* over a signature Σ is defined recursively by:

- $V \subseteq \mathbb{T}(\Sigma)$,
- if $f \in \Sigma$ and $t_1, \ldots, t_{ar(f)} \in \mathbb{T}(\Sigma)$ then $f(t_1, \ldots, t_{ar(f)}) \in \mathbb{T}(\Sigma)$.

A term $c()$ is abbreviated as c. For $t \in \mathbb{T}(\Sigma)$, $var(t)$ denotes the set of variables that occur in t. $T(\Sigma)$ is the set of closed terms over Σ, i.e. the terms $p \in \mathbb{T}(\Sigma)$ with $var(p) = \emptyset$. A Σ-*substitution* σ is a partial function from V to $\mathbb{T}(\Sigma)$. If σ is a substitution and S is any syntactic object, then $\sigma(S)$ denotes the object obtained from S by replacing, for x in the domain of σ, every occurrence of x in S by $\sigma(x)$. In that case $\sigma(S)$ is called a *substitution instance* of S. A Σ-substitution is *closed* if it is a total function from V to $T(\Sigma)$.

Definition 2 Let Σ be a signature. A *positive* Σ-*literal* is an expression $t \xrightarrow{a} t'$ and a *negative* Σ-*literal* an expression $t \not\xrightarrow{a}$ with $t, t' \in \mathbb{T}(\Sigma)$ and $a \in Act$. A *transition rule* over Σ is an expression of the form $\frac{H}{\alpha}$ with H a set of Σ-literals (the *premises* of the rule) and α a positive Σ-literal (the *conclusion*). The left- and right-hand side of α are called the *source* and the *target* of the rule, respectively. A rule $\frac{H}{\alpha}$ with $H = \emptyset$ is also written α. A *transition system specification* (*TSS*), written (Σ, R), consists of a signature Σ and a collection R of transition rules over Σ. A TSS is *positive* if the premises of its rules are positive.

Definition 3 [6] A *GSOS* rule is a transition rule such that

- its source has the form $f(x_1, \ldots, x_{ar(f)})$ with $f \in \Sigma$ and $x_i \in V$,
- the left-hand sides of its premises are variables x_i with $1 \leq i \leq ar(f)$,
- the right-hand sides of its positive premises are variables that that are all distinct, and that do not occur in its source,
- its target only contains variables that also occur in its source or premises.

A *GSOS language*, or TSS in GSOS format, is a TSS whose rules are GSOS rules.

Definition 4 A *transition* over a signature Σ is a closed positive Σ-literal. With structural recursion on p one defines when a GSOS language $\mathcal{L}$ generates a transition $p \xrightarrow{a} p'$ (notation $p \xrightarrow{a}_{\mathcal{L}} p'$):

$f(p_1, \ldots, p_n) \xrightarrow{a}_{\mathcal{L}} q$ iff $\mathcal{L}$ has a transition rule $\frac{H}{f(x_1,\ldots,x_n) \xrightarrow{a} t}$ and there is a closed substitution σ with $\sigma(x_i) = p_i$ for $i = 1, ..., n$ and $\sigma(t) = q$, such that $p_i \xrightarrow{c}_{\mathcal{L}} \sigma(y)$ for $(x_i \xrightarrow{c} y) \in H$ and $\neg\exists r(p_i \xrightarrow{c}_{\mathcal{L}} r)$ for $(x_i \not\xrightarrow{c}) \in H$.

Henceforth a GSOS language $\mathcal{L}$ over a signature Σ is assumed, and closed Σ-terms will be called *processes*. The subscript $\mathcal{L}$ will often be suppressed. Moreover, $Act = A \,\dot{\cup}\, \{\tau\}$ with τ the *silent move* or *hidden action*.

Definition 5 Two processes t and u are *weak bisimulation equivalent* or *weakly bisimilar* ($t \leftrightarroweq_w u$) if $t\mathcal{R}u$ for a symmetric binary relation $\mathcal{R}$ on processes (a *weak bisimulation*) satisfying, for $a \in Act$,

$$\text{if } p\mathcal{R}q \text{ and } p \xrightarrow{a} p' \text{ then there are } q_1, q_2, q' \text{ such that } q \Longrightarrow q_1 \xrightarrow{(a)} q_2 \Longrightarrow q' \text{ and } p'\mathcal{R}q'. \qquad (*)$$

Here $p \Longrightarrow p'$ abbreviates $p = p_0 \xrightarrow{\tau} p_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_n = p'$ for some $n \geq 0$, whereas $p \xrightarrow{(a)} p'$ abbreviates $(p \xrightarrow{a} p') \vee (a = \tau \wedge p = p')$.

t and u are *η-bisimilar* ($t \leftrightarrows_\eta u$) if in (*) one additionally requires $p\mathcal{R}q_1$;
t and u are *delay bisimilar* ($t \leftrightarrows_d u$) if in (*) one additionally requires $q_2 = q'$;
t and u are *branching bisimilar* ($t \leftrightarrows_b u$) if in (*) one requires both $p\mathcal{R}q_1$ and $q_2 = q'$;
t and u are *strongly bisimilar* ($t \leftrightarrows u$) if in (*) one simply requires $q \xrightarrow{a} q'$.
Two processes t and u are *rooted weak bisimulation equivalent* ($t \leftrightarrows_{rw} u$), if they satisfy

if $t \xrightarrow{a} t'$ then there are u_1, u_2, u such that $u \Longrightarrow u_1 \xrightarrow{a} u_2 \Longrightarrow u'$ and $t' \leftrightarrows_w u'$, and
if $u \xrightarrow{a} u'$ then there are t_1, t_2, t such that $t \Longrightarrow t_1 \xrightarrow{a} t_2 \Longrightarrow t'$ and $t' \leftrightarrows_w u'$.

They are *rooted η-bisimilar* ($t \leftrightarrows_{r\eta} u$) if above one additionally requires $u_1 = u$, $t_1 = t$, and $t' \leftrightarrows_\eta u'$, they are *rooted delay bisimilar* ($t \leftrightarrows_{rd} u$) if one requires $u_2 = u'$, $t_2 = t'$ and $t' \leftrightarrows_d u'$, and they are *rooted branching bisimilar* ($t \leftrightarrows_{rb} u$) if one requires $u_1 = u$, $u_2 = u'$, $t_1 = t$, $t_2 = t'$ and $t' \leftrightarrows_b u'$.

It is well known and easy to check that the nine relations on processes defined above are equivalence relations indeed [2, 12], and that, for $x \in$ {weak, η, delay, branching, strong}, x-bisimulation equivalence is the largest x-bisimulation relation on processes [13, 4, 11, 12]. Moreover, $p \leftrightarrows_{rx} q$ implies $p \leftrightarrows_x q$.

Definition 6 An equivalence relation $\sim$ on processes is a *congruence* if for all $f \in \Sigma$

$$p_i \sim q_i \text{ for } i = 1, \ldots, ar(f) \quad \Rightarrow \quad f(p_1, \ldots, p_{ar(f)}) \sim f(q_1, \ldots, q_{ar(f)}).$$

This is equivalent to the requirement that for all $t \in \mathbb{T}(\Sigma)$ and closed substitutions $\sigma, \nu : V \rightarrow T(\Sigma)$

$$\sigma(x) = \nu(x) \text{ for } x \in var(t) \quad \Rightarrow \quad \sigma(t) = \nu(t).$$

This note, and Bloom [5], deal with syntactic conditions on GSOS languages that guarantee that the equivalence notions of Definition 5 are congruences.

3 The congruence proofs for the simply cool rule formats

In this section I first prove the result from [6] that strong bisimulation equivalence is a congruence for all GSOS operators. Then I change this proof in minimal ways so as to obtain the congruence results for the simply cool GSOS languages. In doing so, for each simply cool GSOS format I state a lemma that tells exactly what is needed to make the congruence proof work. Later on, the simply cool congruence formats will be defined in such a way that these lemmas hold.

Theorem 1 [6] On any GSOS language, strong bisimulation equivalence is a congruence.

Proof: Let $\mathcal{R}$ be the smallest relation on processes satisfying

- if $p \leftrightarrows q$ then $p\mathcal{R}q$, and
- if $p_i\mathcal{R}q_i$ for $i = 1, \ldots, ar(f)$ then $f(p_1, \ldots, p_{ar(f)})\mathcal{R}f(q_1, \ldots, q_{ar(f)})$.

It suffices to show that $\mathcal{R}$ is a strong bisimulation, because this implies that $\mathcal{R}$ equals $\leftrightarrows$, and by construction $\mathcal{R}$ is a congruence. Because $\leftrightarrows$ is symmetric, so is $\mathcal{R}$. So it remains to show that

$$\text{if } p\mathcal{R}q \text{ and } p \xrightarrow{a} p', \text{ with } a \in Act, \text{ then there is a } q' \text{ such that } q \xrightarrow{a} q' \text{ and } p'\mathcal{R}q'.$$

This I will do with induction on the number of applications of the second clause in the definition of $\mathcal{R}$ above in establishing that $p\mathcal{R}q$. Note that this number is the same for $q\mathcal{R}p$.

Base case: Let $p \leftrightarrows q$ and $p \xrightarrow{a} p'$. Using that $\leftrightarrows$ is a strong bisimulation, there must be a process q' such that $q \xrightarrow{a} q'$ and $p' \leftrightarrows q'$, hence $p'\mathcal{R}q'$.

Induction step: Let $p = f(p_1,\ldots,p_n)$ and $q = f(q_1,\ldots,q_n)$ where $p_i\mathcal{R}q_i$ for $i = 1,\ldots,n$, and $p_i\mathcal{R}q_i$ is established in less applications of the second step than $p\mathcal{R}q$. By induction, one may assume

$$\text{if } p_i \xrightarrow{c} p' \text{ then there is a } q' \text{ such that } q_i \xrightarrow{c} q' \text{ and } p'\mathcal{R}q' \tag{1}$$
$$\text{if } q_i \xrightarrow{c} q' \text{ then there is a } p' \text{ such that } p_i \xrightarrow{c} p' \text{ and } p'\mathcal{R}q' \tag{2}$$

for $i = 1,...,n$ and $c \in \mathit{Act}$. Let $p \xrightarrow{a} p'$. By Definition 4, there must be a rule $\frac{H}{f(x_1,...,x_n)\xrightarrow{a}t}$ in $\mathcal{L}$ and a closed substitution σ with $\sigma(x_i) = p_i$ for $i = 1,...,n$ and $\sigma(t) = p'$, such that $p_i \xrightarrow{c} \sigma(y)$ for $(x_i \xrightarrow{c} y) \in H$ and $\neg\exists r(p_i \xrightarrow{c} r)$ for $(x_i \overset{c}{\not\longrightarrow}) \in H$.

For $(x_i \xrightarrow{c} y) \in H$, using that $p_i \xrightarrow{c} \sigma(y)$, by (1) there is a q_y such that $q_i \xrightarrow{c} q_y$ and $\sigma(y)\mathcal{R}q_y$.

For $(x_i \overset{c}{\not\longrightarrow}) \in H$, using that $\neg\exists r(p_i \xrightarrow{c} r)$, by (2) there can not be a $s \in T(\Sigma)$ with $q_i \xrightarrow{c} s$. Let ν be a substitution with $\nu(x_i) = q_i$ for $i = 1,\ldots,n$ and $\nu(y) = q_y$ if y is the right-hand side of a premise in H, taking $\nu(z) = \sigma(z)$ for all other variables z; by the third clause of Definition 3 such a substitution ν does indeed exist. I now have $\sigma(x)\mathcal{R}\nu(x)$ for all $x \in V$, and hence $\sigma(t)\mathcal{R}\nu(t)$ by the definition of $\mathcal{R}$. Take $q' = \nu(t)$. So $p'\mathcal{R}q'$. Moreover, $q_i \xrightarrow{c} \nu(y)$ for $(x_i \xrightarrow{c} y) \in H$ and $\neg\exists r(q_i \xrightarrow{c} r)$ for $(x_i \overset{c}{\not\longrightarrow}) \in H$. Thus, by Definition 4, $q = f(q_1,\ldots,q_n) \xrightarrow{a} \nu(t) = q'$. □

In Sections 4 and 6 I define 8 simply cool formats in such a way that the following theorem holds.

Theorem 2 On any simply WB cool GSOS language, $\leftrightarrows_w$ is a congruence.
On any simply RWB cool GSOS language, $\leftrightarrows_{rw}$ is a congruence.
On any simply DB cool GSOS language, $\leftrightarrows_d$ is a congruence.
On any simply RDB cool GSOS language, $\leftrightarrows_{rd}$ is a congruence.
On any simply HB cool GSOS language, $\leftrightarrows_\eta$ is a congruence.
On any simply RHB cool GSOS language, $\leftrightarrows_{r\eta}$ is a congruence.
On any simply BB cool GSOS language, $\leftrightarrows_b$ is a congruence.
On any simply RBB cool GSOS language, $\leftrightarrows_{rb}$ is a congruence.

The proofs are simple variations of the proof of Theorem 1 that merely require the following lemmas. These lemmas could have been taken as definitions of the simply cool rule formats, albeit not very syntactic ones. The real definitions will consist of the simplest syntactic requirements that guarantee these lemmas to hold. For now, all one needs to know about the cool formats is that

- (simply) WB, RWB, DB, RDB, HB and BB cool GSOS languages are required to be positive—hence one doesn't need a counterpart of equation (2) above,
- the targets of all rules in simply RXB languages (X∈ {W,D,H,B}) belong to a simply XB cool sublanguage,
- the simply XB cool languages only have rules of the form $\frac{\{x_i \xrightarrow{c_i} y_i | i\in I\}}{f(x_1,...,x_n)\xrightarrow{a}t}$ for $I \subseteq \{1,\ldots,n\}$.

I write $p \Longrightarrow\xrightarrow{a} p'$ for $\exists p^{pre}(p \Longrightarrow p^{pre} \xrightarrow{a} p')$, and similarly for other relation compositions.

Lemma WB Let $\mathcal{L}$ be simply WB cool, let $\frac{H}{s\xrightarrow{a}t}$ be a rule in $\mathcal{L}$, and let ν be a closed substitution. If for each premise $x \xrightarrow{c} y$ in H one has $\nu(x) \Longrightarrow\xrightarrow{(c)}\Longrightarrow \nu(y)$, then $\nu(s) \Longrightarrow\xrightarrow{(a)}\Longrightarrow \nu(t)$.

Lemma RWB Let $\mathcal{L}$ be simply RWB cool, let $\frac{H}{s \xrightarrow{a} t}$ be a rule in $\mathcal{L}$, and let ν be a closed substitution. If for each premise $x \xrightarrow{c} y$ in H one has $\nu(x) \Longrightarrow\xrightarrow{c}\Longrightarrow \nu(y)$, then $\nu(s) \Longrightarrow\xrightarrow{a}\Longrightarrow \nu(t)$.

Lemma DB Let $\mathcal{L}$ be simply DB cool, let $\frac{H}{s \xrightarrow{a} t}$ be a rule in $\mathcal{L}$, and let ν be a closed substitution. If for each premise $x \xrightarrow{c} y$ in H one has $\nu(x) \Longrightarrow\xrightarrow{(c)} \nu(y)$, then $\nu(s) \Longrightarrow\xrightarrow{(a)} \nu(t)$.

Lemma RDB Let $\mathcal{L}$ be simply RDB cool, let $\frac{H}{s \xrightarrow{a} t}$ be a rule in $\mathcal{L}$, and let ν be a closed substitution. If for each premise $x \xrightarrow{c} y$ in H one has $\nu(x) \Longrightarrow\xrightarrow{c} \nu(y)$, then $\nu(s) \Longrightarrow\xrightarrow{a} \nu(t)$.

Lemma HB Let $\mathcal{L}$ be simply HB cool, let $\frac{\{x_i \xrightarrow{c_i} y_i | i \in I\}}{f(x_1,...,x_n) \xrightarrow{a} t}$ be a rule in $\mathcal{L}$, and let ρ, ν be closed substitutions satisfying $\rho(x_i) \Longrightarrow \nu(x_i) \xrightarrow{(c_i)}\Longrightarrow \nu(y_i)$ for $i \in I$ and $\rho(x_i) = \nu(x_i)$ for $i \notin I$. Then $\rho(f(x_1,...,x_n)) \Longrightarrow \nu(f(x_1,...,x_n)) \xrightarrow{(a)}\Longrightarrow \nu(t)$.

Lemma RHB Let $\mathcal{L}$ be simply RHB cool, let $\frac{H}{s \xrightarrow{a} t}$ be a rule in $\mathcal{L}$, and let ν be a closed substitution such that $\nu(x) \xrightarrow{c}\Longrightarrow \nu(y)$ for each positive premise $x \xrightarrow{c} y$ in H and $\neg\exists r\,(\nu(x) \overset{c}{\not\longrightarrow} r)$ for each negative premise $x \overset{c}{\not\longrightarrow}$ in H. Then $\nu(s) \xrightarrow{a}\Longrightarrow \nu(t)$.

Lemma BB Let $\mathcal{L}$ be simply BB cool, let $\frac{\{x_i \xrightarrow{c_i} y_i | i \in I\}}{f(x_1,...,x_n) \xrightarrow{a} t}$ be a rule in $\mathcal{L}$, and let ρ, ν be closed substitutions satisfying $\rho(x_i) \Longrightarrow \nu(x_i) \xrightarrow{(c_i)} \nu(y_i)$ for $i \in I$ and $\rho(x_i) = \nu(x_i)$ for $i \notin I$. Then $\rho(f(x_1,...,x_n)) \Longrightarrow \nu(f(x_1,...,x_n)) \xrightarrow{(a)} \nu(t)$.

Lemma RBB Let $\mathcal{L}$ be simply RBB cool, let $\frac{H}{s \xrightarrow{a} t}$ be a rule in $\mathcal{L}$, and let ν be a closed substitution such that $\nu(x) \xrightarrow{c} \nu(y)$ for each positive premise $x \xrightarrow{c} y$ in H and $\neg\exists r\,(\nu(x) \overset{c}{\not\longrightarrow} r)$ for each negative premise $x \overset{c}{\not\longrightarrow}$ in H. Then $\nu(s) \xrightarrow{a} \nu(t)$.

By Definition 4, the last lemma holds trivially. The others will be obtained in Sections 4 and 6. With these lemmas the proof of Theorem 2 is easy. I will only present the representative cases of (rooted) weak and branching bisimulation equivalence.

Theorem 2WB On any simply WB cool GSOS language, $\leftrightarroweq_w$ is a congruence.

Proof: Let $\mathcal{R}$ be the smallest relation on processes satisfying

- if $p \leftrightarroweq_w q$ then $p\mathcal{R}q$, and
- if $p_i\mathcal{R}q_i$ for $i = 1, \ldots, ar(f)$ then $f(p_1, \ldots, p_{ar(f)})\mathcal{R}f(q_1, \ldots, q_{ar(f)})$.

It suffices to show that $\mathcal{R}$ is a weak bisimulation, because this implies that $\mathcal{R}$ equals $\leftrightarroweq_w$, and by construction $\mathcal{R}$ is a congruence. Because $\leftrightarroweq_w$ is symmetric, so is $\mathcal{R}$. So it remains to show that

$$\text{if } p\mathcal{R}q \text{ and } p \xrightarrow{a} p', \text{ then there is a } q' \text{ such that } q \Longrightarrow\xrightarrow{(a)}\Longrightarrow q' \text{ and } p'\mathcal{R}q'.$$

This I will do with induction on the number of applications of the second clause in the definition of $\mathcal{R}$ above in establishing $p\mathcal{R}q$.

Base case: Let $p \leftrightarroweq_w q$ and $p \xrightarrow{a} p'$. Using that $\leftrightarroweq_w$ is a weak bisimulation, there must be a process q' such that $q \Longrightarrow\xrightarrow{(a)}\Longrightarrow q'$ and $p' \leftrightarroweq_w q'$, hence $p'\mathcal{R}q'$.

Induction step: Let $p = f(p_1, \ldots, p_n)$ and $q = f(q_1, \ldots, q_n)$ where $p_i \mathcal{R} q_i$ for $i = 1, \ldots, n$, and $p_i \mathcal{R} q_i$ is established in less applications of the second step than $p\mathcal{R}q$. By induction, one may assume

$$\text{if } p_i \xrightarrow{c_i} p'_i \text{ then there is a } q'_i \text{ such that } q_i \Longrightarrow\xrightarrow{(c_i)}\Longrightarrow q'_i \text{ and } p'_i \mathcal{R} q'_i \tag{3}$$

for $i = 1, ..., n$ and $c_i \in \mathit{Act}$. Let $p \xrightarrow{a} p'$. By Definition 4, there must be a rule $\frac{\{x_i \xrightarrow{c_i} y_i \mid i \in I\}}{f(x_1,...,x_n) \xrightarrow{a} t}$ in $\mathcal{L}$ and a closed substitution σ with $\sigma(x_i) = p_i$ for $i = 1, \ldots, n$ and $\sigma(t) = p'$, such that $p_i \xrightarrow{c_i} \sigma(y_i)$ for $i \in I$. So by (3), for $i \in I$ there is a q'_i such that $q_i \Longrightarrow\xrightarrow{(c_i)}\Longrightarrow q'_i$ and $\sigma(y_i)\mathcal{R}q'_i$. Let ν be a substitution with $\nu(x_i) = q_i$ for $i = 1, \ldots, n$ and $\nu(y_i) = q'_i$ for $i \in I$, taking $\nu(z) = \sigma(z)$ for all other variables z. I now have $\sigma(x)\mathcal{R}\nu(x)$ for all $x \in V$, and hence $\sigma(t)\mathcal{R}\nu(t)$ by the definition of $\mathcal{R}$. Take $q' = \nu(t)$. Then $p'\mathcal{R}q'$. Moreover, $\nu(x_i) \Longrightarrow\xrightarrow{(c_i)}\Longrightarrow \nu(y_i)$ for each $i \in I$. Thus, by Lemma WB, $q = \nu(f(x_1, \ldots, x_n)) \Longrightarrow\xrightarrow{(a)}\Longrightarrow \nu(t) = q'$. □

Theorem 2RWB On any simply RWB cool GSOS language, $\leftrightarrows_{rw}$ is a congruence.

Proof: Let f be an operator of arity n, and let $p_i \leftrightarrows_{rw} q_i$ for $i = 1, \ldots, n$. I have to show that $f(p_1, \ldots, p_n) \leftrightarrows_{rw} f(q_1, \ldots, q_n)$. Let $f(p_1, \ldots, p_n) \xrightarrow{a} p'$. By Definition 4, there must be a rule $\frac{H}{f(x_1,...,x_n) \xrightarrow{a} t}$ in $\mathcal{L}$ and a closed substitution σ with $\sigma(x_i) = p_i$ for $i = 1, ..., n$ and $\sigma(t) = p'$, such that $p_i \xrightarrow{c} \sigma(y)$ for $(x_i \xrightarrow{c} y) \in H$. For any $(x_i \xrightarrow{c} y) \in H$, using that $p_i \leftrightarrows_{rw} q_i$, there is a q_y such that $q_i \Longrightarrow\xrightarrow{c}\Longrightarrow q_y$ and $\sigma(y) \leftrightarrows_w q_y$. Let ν be a substitution with $\nu(x_i) = q_i$ for $i = 1, \ldots, n$ and $\nu(y) = q_y$ if y is the right-hand side of a premise in H, taking $\nu(z) = \sigma(z)$ for all other variables z; by the last clause of Definition 3 such a substitution ν does indeed exist. I now have $\sigma(x) \leftrightarrows_w \nu(x)$ for all $x \in V$, and hence $\sigma(t) \leftrightarrows_w \nu(t)$ by Theorem 2WB. Take $q' = \nu(t)$. So $p' \leftrightarrows_w q'$. Moreover, $\nu(x_i) \Longrightarrow\xrightarrow{c}\Longrightarrow \nu(y)$ for each premise $(x_i \xrightarrow{c} y) \in H$. Thus, by Lemma RWB, $f(q_1, \ldots, q_n) = \nu(f(x_1, \ldots, x_n)) \Longrightarrow\xrightarrow{a}\Longrightarrow \nu(t) = q'$.

The case assuming $f(q_1, \ldots, q_n) \xrightarrow{a} q'$ follows by symmetry. □

Theorem 2BB On any simply BB cool GSOS language, $\leftrightarrows_b$ is a congruence.

Proof: Let $\mathcal{R}$ be the smallest relation on processes satisfying

- if $p \leftrightarrows_b q$ then $p\mathcal{R}q$, and
- if $p_i \mathcal{R} q_i$ for $i = 1, \ldots, ar(f)$ then $f(p_1, \ldots, p_{ar(f)}) \mathcal{R} f(q_1, \ldots, q_{ar(f)})$.

It suffices to show that $\mathcal{R}$ is a branching bisimulation, because this implies that $\mathcal{R}$ equals $\leftrightarrows_b$, and by construction $\mathcal{R}$ is a congruence. Because $\leftrightarrows_b$ is symmetric, so is $\mathcal{R}$. So it remains to show that

$$\text{if } p\mathcal{R}q \text{ and } p \xrightarrow{a} p' \text{ then there are } q^{pre}, q' \text{ such that } q \Longrightarrow q^{pre} \xrightarrow{(a)} q', \; p\mathcal{R}q^{pre} \text{ and } p'\mathcal{R}q'.$$

This I will do with induction on the number of applications of the second clause in the definition of $\mathcal{R}$ above in establishing $p\mathcal{R}q$.

Base case: Let $p \leftrightarrows_b q$ and $p \xrightarrow{a} p'$. Using that $\leftrightarrows_b$ is a branching bisimulation, there must be processes q^{pre}, q' such that $q \Longrightarrow q^{pre} \xrightarrow{(a)} q'$, $p \leftrightarrows_b q^{pre}$ and $p' \leftrightarrows_b q'$, hence $p\mathcal{R}q^{pre}$ and $p'\mathcal{R}q'$.

Induction step: Let $p = f(p_1, \ldots, p_n)$ and $q = f(q_1, \ldots, q_n)$ where $p_i \mathcal{R} q_i$ for $i = 1, \ldots, n$, and $p_i \mathcal{R} q_i$ is established in less applications of the second step than $p\mathcal{R}q$. By induction, one may assume

$$\text{if } p_i \xrightarrow{c_i} p'_i \text{ then there are } q_i^{pre}, q'_i \text{ such that } q_i \Longrightarrow q_i^{pre} \xrightarrow{(c_i)} q'_i, \; p_i \mathcal{R} q_i^{pre} \text{ and } p'_i \mathcal{R} q'_i \tag{4}$$

for $i=1,...,n$ and $c_i \in Act$. Let $p \xrightarrow{a} p'$. By Definition 4, there must be a rule $\frac{\{x_i \xrightarrow{c_i} y_i | i \in I\}}{f(x_1,...,x_n) \xrightarrow{a} t}$ in $\mathcal{L}$ and a closed substitution σ with $\sigma(x_i)=p_i$ for $i=1,\ldots,n$ and $\sigma(t)=p'$, such that $p_i \xrightarrow{c_i} \sigma(y_i)$ for $i \in I$. So by (4), for $i \in I$ there are q_i^{pre}, q_i' such that $q_i \Longrightarrow q_i^{pre} \xrightarrow{(c_i)} q_i'$, $p_i \mathcal{R} q_i^{pre}$ and $\sigma(y_i)\mathcal{R} q_i'$. Let ν be a substitution with $\nu(x_i)=q_i^{pre}$ and $\nu(y_i)=q_i'$ for $i \in I$ and $\nu(x_i)=q_i$ for $i \notin I$, taking $\nu(z)=\sigma(z)$ for all other variables z. I now have $\sigma(x)\mathcal{R}\nu(x)$ for all $x \in V$, and hence $\sigma(t)\mathcal{R}\nu(t)$ and $\sigma(f(x_1,\ldots,x_n)\mathcal{R}\nu(f(x_1,\ldots,x_n)$ by the definition of $\mathcal{R}$. Take $q^{pre}=\nu(f(x_1,\ldots,x_n))$ and $q'=\nu(t)$. So $p\mathcal{R}q^{pre}$ and $p'\mathcal{R}q'$. Moreover, $q_i \Longrightarrow \nu(x_i) \xrightarrow{(c_i)} \nu(y)$ for each $i \in I$. Thus Lemma BB, taking ρ to be a closed substitution with $\rho(x_i)=q_i$, yields $q \Longrightarrow q^{pre} \xrightarrow{(a)} q'$. □

Theorem 2RBB On any simply RBB cool GSOS language, $\leftrightarroweq_{rb}$ is a congruence.

Proof: Let f be an operator of arity n, and let $p_i \leftrightarroweq_{rb} q_i$ for $i=1,\ldots,n$. I have to show that $f(p_1,\ldots,p_n) \leftrightarroweq_{rb} f(q_1,\ldots,q_n)$. Let $f(p_1,\ldots,p_n) \xrightarrow{a} p'$. By Definition 4, there must be a rule $\frac{H}{f(x_1,...,x_n) \xrightarrow{a} t}$ in $\mathcal{L}$ and a closed substitution σ with $\sigma(x_i)=p_i$ for $i=1,...,n$ and $\sigma(t)=p'$, such that $p_i \xrightarrow{c} \sigma(y)$ for $(x_i \xrightarrow{c} y) \in H$ and $\neg\exists r(p_i \xrightarrow{c} r)$ for $(x_i \not\xrightarrow{c}) \in H$.

For any $(x_i \xrightarrow{c} y) \in H$, using that $p_i \leftrightarroweq_{rb} q_i$, there is a q_y such that $q_i \xrightarrow{c} q_y$ and $\sigma(y) \leftrightarroweq_b q_y$.

For any $(x_i \not\xrightarrow{c}) \in H$, using that $p_i \leftrightarroweq_{rb} q_i$, there can not be a $s \in T(\Sigma)$ with $q_i \xrightarrow{c} s$.

Let ν be a substitution with $\nu(x_i)=q_i$ for $i=1,\ldots,n$ and $\nu(y)=q_y$ if y is the right-hand side of a premise in H, taking $\nu(z)=\sigma(z)$ for all other variables z; by the last clause of Definition 3 such a substitution ν does indeed exist. I now have $\sigma(x) \leftrightarroweq_b \nu(x)$ for all $x \in V$, and hence $\sigma(t) \leftrightarroweq_b \nu(t)$ by Theorem 2BB. Take $q'=\nu(t)$. So $p' \leftrightarroweq_b q'$. Moreover, $\nu(x_i) \xrightarrow{c} \nu(y)$ for each premise $x_i \xrightarrow{c} y$ in H and $\neg\exists r(\nu(x_i) \xrightarrow{c} r)$ for each premise $x_i \not\xrightarrow{c}$ in H. Thus, by Lemma RBB, or Definition 4, $f(q_1,\ldots,q_n)=\nu(f(x_1,\ldots,x_n)) \xrightarrow{a} \nu(t)=q'$.

The case assuming $f(q_1,\ldots,q_n) \xrightarrow{a} q'$ follows by symmetry. □

4 Simply cool GSOS languages

In this section I will define the simply XB cool rule formats (X∈ {W,D,H,B}) and show that they satisfy Lemma XB. Let $\mathcal{L}$ be a positive GSOS language.

Definition 7 For an operator f in $\mathcal{L}$, the *rules of* f are the rules in $\mathcal{L}$ with source $f(x_1,...,x_{ar(f)})$.

- An operator in $\mathcal{L}$ is *straight* if it has no rules in which a variable occurs multiple times in the left-hand side of its premises. A operator is *smooth* if moreover it has no rules in which a variable occurs both in the target and in the left-hand side of a premise.
- An argument $i \in \mathbb{N}$ of an operator f is *active* if f has a rule in which x_i appears as left-hand side of a premise.
- A variable x occurring in a term t is *receiving* in t if t is the target of a rule in $\mathcal{L}$ in which x is the right-hand side of a premise. An argument $i \in \mathbb{N}$ of an operator f is *receiving* if a variable x is receiving in a term t that has a subterm $f(v_1,\ldots,v_n)$ with x occurring in v_i.
- A rule of the form $\dfrac{x_i \xrightarrow{\tau} y}{f(x_1,\ldots,x_n) \xrightarrow{\tau} f(x_1,\ldots,x_n)[y/x_i]}$ with $1 \le i \le n$ is called a *patience rule* for the ith argument of f. Here $t[y/x]$ denotes term t with all occurrences of x replaced by y.

Non-straight operators in positive GSOS languages rarely occur in the literature.

Definition 8 A GSOS language $\mathcal{L}$ is *simply WB cool* if it is positive and

1. all operators in $\mathcal{L}$ are straight,
2. patience rules are the only rules in $\mathcal{L}$ with τ-premises,
3. every active argument of an operator has a patience rule,
4. every receiving argument of an operator has a patience rule,
5. all operators in $\mathcal{L}$ are smooth.

The formats *simply DB cool*, *simply HB cool* and *simply BB cool* are defined likewise, but skipping Clause 4 for DB and BB, and Clause 5 for HB and BB.

The simply WB and BB cool formats above coincide with the ones of [5], whereas the simply DB cool format coincides with the `eb` format of [19].

Example 1 Consider the following fragment of the language *Communicating Sequential Processes* (CSP), for a given set A of visible actions. The set $\mathcal{P}$ of CSP expressions is defined inductively by

0	$\in \mathcal{P}$		(inaction)
$a.P$	$\in \mathcal{P}$	for $a \in A$ and $P \in \mathcal{P}$	(action prefix)
$P \sqcap Q$	$\in \mathcal{P}$	for $P, Q \in \mathcal{P}$	(internal choice)
$P \mathrel{\Box} Q$	$\in \mathcal{P}$	for $P, Q \in \mathcal{P}$	(external choice)
$P \|_S Q$	$\in \mathcal{P}$	for $P, Q \in \mathcal{P}$ and $S \subseteq A$	(parallel composition)
$P \backslash a$	$\in \mathcal{P}$	for $P \in \mathcal{P}$ and $a \in A$	(abstraction)

Roughly, the meaning of these process constructions is as follows. The process 0 never performs any actions. $a.P$ denotes a process which first performs the action a and then behaves as P. $P \sqcap Q$ is a process that first makes a choice between P and Q and subsequently behaves like the chosen process. $P \mathrel{\Box} Q$ is a process that behaves either like P or like Q, the choice being made by the occurrence of any visible action of either P or Q, which in turn may be influenced by the environment. $P\|_S Q$ denotes the parallel composition of processes P and Q. Actions $a \in S$ enforce synchronisation between P and Q; they can only happen when both arguments can partake in performing them. Actions $a \notin S$ of either P and Q will be interleaved. $P \backslash a$ behaves like P, but with the action a made invisible or internal.

The following operational semantics of CSP stems from Olderog and Hoare [15]; it is shown to be consistent with the original denotational semantics of Hoare, Brookes & Roscoe [7]. It models the making of an internal choice through an internal action $\tau \notin A$ and captures the interplay between internal and external choice by assuming that in $P \mathrel{\Box} Q$, as long as no visible actions occurs, the internal actions of P and Q happen in parallel. Below, a and b range over $Act = A \mathbin{\dot\cup} \{\tau\}$.

$$a.x_1 \xrightarrow{a} x_1 \qquad x_1 \sqcap x_2 \xrightarrow{\tau} x_1 \qquad x_1 \sqcap x_2 \xrightarrow{\tau} x_2$$

$$\frac{x_1 \xrightarrow{a} y_1}{x_1 \mathrel{\Box} x_2 \xrightarrow{a} y_1}\ (a \neq \tau) \qquad \frac{x_2 \xrightarrow{a} y_2}{x_1 \mathrel{\Box} x_2 \xrightarrow{a} y_2}\ (a \neq \tau) \qquad \frac{x_1 \xrightarrow{\tau} y_1}{x_1 \mathrel{\Box} x_2 \xrightarrow{\tau} y_1 \mathrel{\Box} x_2} \qquad \frac{x_2 \xrightarrow{\tau} y_2}{x_1 \mathrel{\Box} x_2 \xrightarrow{\tau} x_1 \mathrel{\Box} y_2}$$

$$\frac{x_1 \xrightarrow{a} y_1}{x_1\|_S x_2 \xrightarrow{a} y_1\|_S x_2}\ (a \notin S) \qquad \frac{x_2 \xrightarrow{a} y_2}{x_1\|_S x_2 \xrightarrow{a} x_1\|_S y_2}\ (a \notin S) \qquad \frac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{a} y_2}{x_1\|_S x_2 \xrightarrow{a} y_1\|_S y_2}\ (a \in S)$$

$$\frac{x_1 \xrightarrow{a} y_1}{x_1 \backslash a \xrightarrow{\tau} y_1 \backslash a} \qquad \frac{x_1 \xrightarrow{b} y_1}{x_1 \backslash a \xrightarrow{b} y_1 \backslash a}\ (b \neq a)$$

This makes CSP into a WB cool GSOS language. Hence, $\leftrightarrows_w$, $\leftrightarrows_d$, $\leftrightarrows_\eta$ and $\leftrightarrows_b$ are congruences on this language.

Example 2 The GSOS language with 0, action prefix operators $a.P$ for $a \in Act$ and the unary operator n with rules $\frac{x_1 \xrightarrow{\tau} y_1}{n(x_1) \xrightarrow{\tau} n(y_1)}$ and $\frac{x_1 \stackrel{a}{\not\longrightarrow}}{n(x_1) \xrightarrow{c} 0}$ (for a specific action a) is not positive, but otherwise meets the requirements of Definition 8. Yet, $\leftrightarrows_w$, $\leftrightarrows_d$, $\leftrightarrows_\eta$ and $\leftrightarrows_b$ fail to be congruences on this language. Namely $a.0 \leftrightarrows_b \tau.a.0$ yet $n(a.0) \not\leftrightarrows_w n(\tau.a.0)$, as only the latter process can perform a c-transition. (Here I use that $\leftrightarrows_b$ is the finest equivalence of $\leftrightarrows_w$, $\leftrightarrows_d$, $\leftrightarrows_\eta$ and $\leftrightarrows_b$, and $\leftrightarrows_w$ the coarsest. Thus we have $a.0 \leftrightarrows_x \tau.a.0$ yet $n(a.0) \not\leftrightarrows_x n(\tau.a.0)$ for any $x \in \{w, d, \eta, b\}$.) This shows that the requirement that $\mathcal{L}$ be positive cannot simply be skipped.

Example 3 An operator q with a rules $\frac{x_1 \xrightarrow{\tau} y_1}{q(x_1) \xrightarrow{\tau} q(y_1)}$ and $\frac{x_1 \xrightarrow{a} y,\ x_1 \xrightarrow{b} z}{q(x_1) \xrightarrow{c} q(y)}$ (for three specific actions $a, b, c \in A$) would not be straight, although it satisfies the other requirements of Definition 8. A process p with $p \xrightarrow{a} q$ and $p \xrightarrow{b} r$ (and no other outgoing transitions) is branching bisimilar to processes p' as well as p'' with $p' \xrightarrow{\tau} p''$, $p'' \xrightarrow{\tau} p'$, $p' \xrightarrow{a} q$ and $p'' \xrightarrow{b} r$ (and p' and p'' having no other outgoing transitions). Yet $q(p)$ can do a c-action, whereas $q(p')$ cannot. This shows that in general $\leftrightarrows_w$, $\leftrightarrows_d$, $\leftrightarrows_\eta$ and $\leftrightarrows_b$ fail to be congruences on languages incorporating this operator. Hence requirement 1 of Definition 8 cannot be skipped.

Example 4 The GSOS language with 0, action prefix operators $a.P$ for $a \in Act$ and the unary operator m with rules $\frac{x_1 \xrightarrow{\tau} y_1}{m(x_1) \xrightarrow{\tau} m(y_1)}$ and $\frac{x_1 \xrightarrow{\tau} y_1}{m(x) \xrightarrow{c} m(y_1)}$ fails requirement 2 of Definition 8 only. As $0 \leftrightarrows_b \tau.0$ but $m(0) \not\leftrightarrows_w m(\tau.0)$, given that only the latter process can do a c-transition, $\leftrightarrows_w$, $\leftrightarrows_d$, $\leftrightarrows_\eta$ and $\leftrightarrows_b$ fail to be congruences on this language. Thus also requirement 2 cannot be skipped.

Example 5 The GSOS language with 0, action prefix operators $a.P$ for $a \in Act$ and the unary operator i with rule $\frac{x_1 \xrightarrow{a} y_1}{i(x) \xrightarrow{c} 0}$, and no patience rule, fails requirement 3 of Definition 8 only, considering that the argument of i is active but not receiving. As $a.0 \leftrightarrows_b \tau.a.0$ but $i(a.0) \not\leftrightarrows_w i(\tau.a.0)$, given that only the former process can ever do a c-transition, $\leftrightarrows_w$, $\leftrightarrows_d$, $\leftrightarrows_\eta$ and $\leftrightarrows_b$ fail to be congruences on this language. Thus also requirement 3 cannot be skipped.

Example 6 The GSOS language with 0, $\square$, $\sqcap$, action prefix operators $a.P$ for $a \in Act$ and the unary operators f, g with rules $\frac{x_1 \xrightarrow{\tau} y_1}{f(x_1) \xrightarrow{\tau} f(y_1)}$, $\frac{x_1 \xrightarrow{a} y_1}{f(x_1) \xrightarrow{a} g(y_1)}$ and $g(x) \xrightarrow{c} x$, fails requirement 4 of Definition 8 only, considering that the argument of g is receiving but not active. Hence $\leftrightarrows_d$ and $\leftrightarrows_b$ are congruences on this language. One has $a.(b.0 \sqcap d.0) \leftrightarrows_\eta a.(b.0 \sqcap d.0) \,\square\, a.b.0$, yet

$$f(a.(b.0 \sqcap d.0)) \leftrightarrows a.c.(b.0 \sqcap d.0) \not\leftrightarrows_w a.c.(b.0 \sqcap d.0) \,\square\, a.c.b.0 \leftrightarrows f(a.(b.0 \sqcap d.0) \,\square\, a.b.0)$$

given that only the right-hand process can do an a-transition (possibly followed or preceded by internal actions) and reach a state in which the possibility to ever do a d-transition is ruled out. Hence $\leftrightarrows_w$ and $\leftrightarrows_\eta$ fail to be congruences on this language. Thus also requirement 4 cannot be skipped for the simply WB and HB cool formats.

Example 7 An example of a straight but not smooth operator is the operator s of [12, Section 10(4)] that allows a process (its argument) to proceed normally, but in addition can report that the process is ready to perform a visible action, without actually doing it. It supposes an alphabet $Act = \mathcal{L} \dot{\cup} \{\text{Can do `}a\text{'} \mid a \in \mathcal{L}\} \dot{\cup} \{\tau\}$, and its rules are

$$\frac{x \xrightarrow{a} y}{s(x) \xrightarrow{a} s(y)}\ (a \in Act) \qquad\qquad \frac{x \xrightarrow{a} y}{s(x) \xrightarrow{\text{Can do `}a\text{'}} s(x)}\ (a \in \mathcal{L})$$

It is not smooth because in the latter rule x occurs in the target as well as in the left-hand side of its premise. The GSOS language that combines this operator with 0, action prefix operators $a.P$ for $a \in Act$, and constants p and q with rules $p \xrightarrow{a} 0$, $p \xrightarrow{\tau} b.0$, $q \xrightarrow{a} 0$, $q \xrightarrow{\tau} b.0$ and $q \xrightarrow{b} 0$, satisfies requirements 1–4 of Definition 8 but not requirement 5. Hence $\leftrightarrows_\eta$ and $\leftrightarrows_b$ are congruences on this language. Yet $\leftrightarrows_d$ and $\leftrightarrows_w$ are not, for $p \leftrightarrows_d q$ whereas $s(p) \not\leftrightarrows_w s(q)$. Namely only $s(q)$ can report "can do 'b'" and then do a (see Figure 7 in [12]). Thus also requirement 5 cannot be skipped for the simply WB and DB cool formats.

Lemma 1 Suppose $\mathcal{L}$ satisfies Clause 4 in the definition of simply WB cool above, and let μ, ν be closed substitutions. If $\mu(y) \Longrightarrow \nu(y)$ for every $y \in var(t)$ that is receiving in t, and $\mu(x) = \nu(x)$ for every $x \in var(t)$ that is not receiving in t, then $\mu(t) \Longrightarrow \nu(t)$.

Proof: By structural induction on t. If t is a variable, the statement follows by assumption. Otherwise, $t = f(t_1, \dots, t_n)$. It suffices to show that for $i = 1, \dots, n$ one has

$$f(\nu(t_1), ..., \nu(t_{i-1}), \mu(t_i), \mu(t_{i+1}), ..., \mu(t_n)) \Longrightarrow f(\nu(t_1), ...\nu(t_{i-1}), \nu(t_i), \mu(t_{i+1}), ..., \mu(t_n)). \qquad (5)$$

If t_i contains no variable that is receiving in t, then $\mu(t_i) = \nu(t_i)$ by assumption, which yields (5). If t_i does contain such a variable, then, by definition, i is a receiving argument of f. By induction, $\mu(t_i) \Longrightarrow \nu(t_i)$, and the patience rule for the i^{th} argument of f yields (5). $\square$

Proof of Lemma WB: Let $\mathcal{L}$ be simply WB cool, let $r = \frac{\{x_i \xrightarrow{c_i} y_i \mid i \in I\}}{f(x_1,\dots,x_n) \xrightarrow{a} t}$ be a rule in $\mathcal{L}$ and let μ, ν be closed substitutions such that

- $\nu(x_i) \Longrightarrow \mu(x_i) \xrightarrow{(c_i)} \mu(y_i) \Longrightarrow \nu(y_i)$ for $i \in I$,
- $\nu(x) = \mu(x)$ for all variables x that do not occur in the premises of r.

I need to show that $\nu(f(x_1, \dots, x_n)) \Longrightarrow\xrightarrow{(a)}\Longrightarrow \nu(t)$.

In case r is a patience rule—so $I = \{k\}$, $c_k = \tau$ and $t = f(x_1, \dots, x_n)[y_k/x_k]$ with $1 \leq k \leq n$—one has $\nu(f(x_1, ..., x_n)) \Longrightarrow \mu(f(x_1, ..., x_n)) \xrightarrow{(\tau)} \mu(f(x_1, ..., x_n)[y_k/x_k]) \Longrightarrow \nu(f(x_1, ..., x_n)[y_k/x_k])$ by repeated application of r.

Otherwise, $\xrightarrow{(c)} = \xrightarrow{c}$ by Clause 2 of Definition 8. Now $\nu(f(x_1, \dots, x_n)) \Longrightarrow \mu(f(x_1, \dots, x_n))$ by Clause 3, and $\mu(f(x_1, \dots, x_n)) \xrightarrow{a} \mu(t)$ by application of r. Clause 5 yields that $\mu(x) = \nu(x)$ for all variables $x \in var(t)$ that are not receiving in t, so $\mu(t) \Longrightarrow \nu(t)$ by Lemma 1. $\square$

The **Proof of Lemma DB** proceeds likewise, but taking $\mu(x) = \nu(x)$ for all variables x that do not occur in the left-hand side of premises. Clause 5 now implies that $\mu(t) = \nu(t)$. $\square$

Proof of Lemma HB: Let $\mathcal{L}$ be simply HB cool, let $r = \frac{\{x_i \xrightarrow{c_i} y_i \mid i \in I\}}{f(x_1,\dots,x_n) \xrightarrow{a} t}$ be a rule in $\mathcal{L}$ and let ρ, μ, ν be closed substitutions such that

- $\rho(x_i) \Longrightarrow \nu(x_i) \xrightarrow{(c_i)} \mu(y_i) \Longrightarrow \nu(y_i)$ for $i \in I$,

- $\rho(x_i) = \nu(x_i) = \mu(x_i)$ for $i \not\in I$, and
- $\nu(x) = \mu(x)$ for all variables x that do not occur as right-hand sides of premises in H.

I need to show that $\rho(f(x_1, \dots, x_n)) \Longrightarrow \nu(f(x_1, \dots, x_n)) \xrightarrow{(a)}\Longrightarrow \nu(t)$.
Note that $\nu(f(x_1, \dots, x_n)) = \mu(f(x_1, \dots, x_n))$.

In case r is a patience rule—so $I = \{k\}$, $c_k = \tau$ and $t = f(x_1, \dots, x_n)[y_k/x_k]$ with $1 \leq k \leq n$—one has $\rho(f(x_1, ..., x_n)) \Longrightarrow \nu(f(x_1, ..., x_n)) \xrightarrow{(\tau)} \mu(f(x_1, ..., x_n)[y_k/x_k]) \Longrightarrow \nu(f(x_1, ..., x_n)[y_k/x_k])$ by repeated application of r.

Otherwise, $\xrightarrow{(c)} = \xrightarrow{c}$ by Clause 2 of Definition 8. Now $\rho(f(x_1, \dots, x_n)) \Longrightarrow \nu(f(x_1, \dots, x_n))$ by Clause 3, $\nu(f(x_1, \dots, x_n)) \xrightarrow{a} \mu(t)$ by application of r; and $\mu(t) \Longrightarrow \nu(t)$ by Lemma 1. $\Box$

The **Proof of Lemma BB** proceeds likewise, but omitting μ. $\Box$

5 Cool GSOS languages

In this section I will extend the simply XB cool rule formats to XB cool rule formats and establish the associated congruence theorems (X$\in$ {W,D,H,B}).

Definition 9 A GSOS language is *two-tiered* if its operators are partitioned into *abbreviations* and *principal operators*, and for every abbreviation f a principal operator $f^\star$ is specified, together with a substitution $\sigma_f : \{x_1, \dots, x_{ar(f^\star)}\} \rightarrow \{x_1, \dots, x_{ar(f)}\}$, such that the rules of f are

$$\left\{ \frac{\sigma_f(H)}{f(x_1, \dots, x_{ar(f)}) \xrightarrow{a} \sigma_f(t)} \;\middle|\; \frac{H}{f^\star(x_1, \dots, x_{ar(f^\star)}) \xrightarrow{a} t} \text{ is a rule of } f^\star \right\}.$$

Write $f(i)$ for the j such that $\sigma_f(x_i) = x_j$; take $f^\star = f$ and $f(i) = i$ in case f is a principal operator.

Trivially, any positive GSOS language can be extended (*straightened*) to a two-tiered GSOS language whose principal operators are (straight and) smooth [1].

Example 8 Let $\mathcal{L}$ have a binary operator f with as only rule $\frac{x_1 \xrightarrow{a} y, \quad x_1 \xrightarrow{b} z}{f(x_1,x_2) \xrightarrow{a} f(x_1,(f(y,x_2))}$.

Then $\mathcal{L}$ can be straightened by adding a operator $f^\star$ with rule $\frac{x_1 \xrightarrow{a} y, \quad x_2 \xrightarrow{b} z}{f^\star(x_1,x_2,x_3,x_4) \xrightarrow{a} f(x_3,f(y,x_4))}$.
In this case $\sigma_f(x_4) = x_2$ and $\sigma_f(x_1) = \sigma_f(x_2) = \sigma_f(x_3) = x_1$.

Equally trivial, $f(p_1, ..., p_{ar(f)}) \xrightarrow{a} t$ iff $f^\star(p_{f(1)}, ..., p_{f(ar(f^\star))}) \xrightarrow{a} t$;
so $f(p_1, ..., p_{ar(f)}) \leftrightarrows f^\star(p_{f(1)}, ..., p_{f(ar(f^\star))})$.

Definition 10 A two-tiered GSOS language $\mathcal{L}$ is *WB cool* if it is positive and

1. all principal operators in $\mathcal{L}$ are straight,
2. patience rules are the only rules of principal operators with τ-premises,
3. every active argument of a principal operator has a patience rule,
4. if argument $f(i)$ of operator f is receiving, then argument i of $f^\star$ has a patience rule,
5. all principal operators in $\mathcal{L}$ are smooth.

The formats *DB cool*, *HB cool* and *BB cool* are defined likewise, but skipping Clause 4 for DB and BB, and Clause 5 for HB and BB. Clause 4 may be weakened slightly; see Section 5.2.

Note that the simply cool formats defined before are exactly the cool formats with the extra restriction that all operators are principal.

Theorem 3 On any WB cool GSOS language, weak bisimulation equivalence is a congruence.
On any DB cool GSOS language, delay bisimulation equivalence is a congruence.
On any HB cool GSOS language, η-bisimulation equivalence is a congruence.
On any BB cool GSOS language, branching bisimulation equivalence is a congruence.

Given that the cool GSOS languages differ from the simply cool GSOS languages only by the addition of operators that can be regarded as syntactic sugar, the theorems above are a simple consequence of the corresponding theorems for simply cool GSOS languages. Below I go through the details.

Definition 11 Let $\mathcal{L}$ be a two-tiered GSOS language, with signature Σ. Let $\Sigma^\star$ be the subcollection of principal operators in Σ, and $\Sigma^* = \{f^* \mid f \in \Sigma^\star\}$ be a collection of fresh names for the latter. For $f \in \Sigma$ an abbreviation, write f^* for $(f^\star)^*$. Define the translation $^* : T(\Sigma) \rightarrow T(\Sigma^*)$ recursively by $x^* = x$ for $x \in V$ and $(f(t_1, \dots, t_{ar(f)}))^* = f^*(t^*_{f(1)}, \dots, t^*_{f(ar(f^*))})$. Let $\mathcal{L}^*$ be the GSOS language with signature Σ^* and rules $\frac{H}{f^*(x_1,\dots,x_{ar(f)}) \xrightarrow{a} t^*}$ for $f \in \Sigma^\star$ and $\frac{H}{f(x_1,\dots,x_{ar(f)}) \xrightarrow{a} t}$ a rule of $\mathcal{L}$.

Observation 1 Let $\mathcal{L}$ be XB cool, with X$\in$ {W,D,H,B}. Then $\mathcal{L}^*$ is simply XB cool.

Any equivalence relation $\sim$ on processes defined in terms of the transitions between them, naturally extends to an equivalence relation on the disjoint union of $T(\Sigma)$ and $T(\Sigma^*)$, with $\mathcal{L}$ generating the transitions between processes from $T(\Sigma)$ and $\mathcal{L}^*$ generating the transitions between processes from $T(\Sigma)^*$ (see Definition 4).

Lemma 2 Let $\mathcal{L}$ be a two-tiered GSOS language, with signature Σ. Then $p^* \xrightarrow{a}_{\mathcal{L}^*} p'_{\text{next}}$ iff $\exists p_{\text{next}}(p'_{\text{next}} = p^*_{\text{next}} \wedge p \xrightarrow{a}_{\mathcal{L}} p_{\text{next}})$.

Proof: Note that any term in $\mathcal{L}^*$ has the form t^*, with t a term in $\mathcal{L}$, uniquely determined by t^*. Using this, the statement of the lemma can be simplified to $p^* \xrightarrow{a}_{\mathcal{L}^*} p^*_{\text{next}}$ iff $p \xrightarrow{a}_{\mathcal{L}} p_{\text{next}}$.

"If": Suppose $p^* \xrightarrow{a}_{\mathcal{L}^*} p^*_{\text{next}}$. Let $p^* := f^*(p^*_1, ..., p^*_n)$. By Definition 4, $\mathcal{L}^*$ has a transition rule $\frac{H}{f^*(x_1,\dots,x_n) \xrightarrow{a} t^*}$ and there is a closed substitution $\sigma^* : V \rightarrow \mathbb{T}(\Sigma^*)$ with $\sigma^*(x_i) = p^*_i$ for $i = 1, ..., n$ and $\sigma^*(t^*) = p^*_{\text{next}}$, such that $p^*_i \xrightarrow{c}_{\mathcal{L}} \sigma^*(y)$ for $(x_i \xrightarrow{c} y) \in H$ and $\neg\exists r^*(p^*_i \xrightarrow{c}_{\mathcal{L}} r^*)$ for $(x_i \overset{c}{\not\longrightarrow}) \in H$. Let $\sigma : V \rightarrow \mathbb{T}(\Sigma)$ be the closed substitution with $\sigma^*(x) = \sigma(x)^*$ for all $x \in V$. Then $\sigma(t) = p_{\text{next}}$. By induction, $p_i \xrightarrow{c}_{\mathcal{L}} \sigma(y)$ for $(x_i \xrightarrow{c} y) \in H$ and $\neg\exists r(p_i \xrightarrow{c}_{\mathcal{L}} r)$ for $(x_i \overset{c}{\not\longrightarrow}) \in H$. Using Definition 11, $\mathcal{L}$ must have a transition rule $\frac{H}{f(x_1,\dots,x_n) \xrightarrow{a} t}$. So by Definition 4, $p = f(p_1, ..., p_n) \xrightarrow{a}_{\mathcal{L}} p_{\text{next}}$.

"Only if": Suppose $p \xrightarrow{a}_{\mathcal{L}} p_{\text{next}}$. Let $p := f(p_1, ..., p_n)$. I first deal with the case that $f \in \Sigma^\star$. By Definition 4, $\mathcal{L}$ has a transition rule $\frac{H}{f(x_1,\dots,x_n) \xrightarrow{a} t}$ and there is a closed substitution $\sigma : V \rightarrow \mathbb{T}(\Sigma)$ with $\sigma(x_i) = p_i$ for $i = 1, ..., n$ and $\sigma(t) = p_{\text{next}}$, such that $p_i \xrightarrow{c}_{\mathcal{L}} \sigma(y)$ for $(x_i \xrightarrow{c} y) \in H$ and $\neg\exists r(p_i \xrightarrow{c}_{\mathcal{L}} r)$ for $(x_i \overset{c}{\not\longrightarrow}) \in H$. Let $\sigma^* : V \rightarrow \mathbb{T}(\Sigma^*)$ be the closed substitution with $\sigma^*(x) = \sigma(x)^*$ for all $x \in V$. Then $\sigma^*(t^*) = p^*_{\text{next}}$. By induction, $p^*_i \xrightarrow{c}_{\mathcal{L}} \sigma^*(y)$ for $(x_i \xrightarrow{c} y) \in H$ and $\neg\exists r^*(p^*_i \xrightarrow{c}_{\mathcal{L}} r^*)$ for $(x_i \overset{c}{\not\longrightarrow}) \in H$. Using Definition 11, $\mathcal{L}^*$ must have a transition rule $\frac{H}{f^*(x_1,\dots,x_n) \xrightarrow{a} t^*}$. So by Definition 4, $p^* = f^*(p^*_1, ..., p^*_n) \xrightarrow{a}_{\mathcal{L}^*} p^*_{\text{next}}$.

Next, consider the case that $f \notin \Sigma^\star$. Then $q \xrightarrow{a}_{\mathcal{L}} p_{\text{next}}$ with $q := f^\star(p_{f(1)}, ..., p_{f(ar(f^\star))})$. As $f^\star \in \Sigma^\star$, the previous case of this proof yields $q^* \xrightarrow{a}_{\mathcal{L}^*} p^*_{\text{next}}$. But $p^* = f^*(t^*_{f(1)}, \ldots, t^*_{f(ar(f^*))}) = q^*$, by Definition 11. $\Box$

Corollary 1 Let $\mathcal{L}$ be a two-tired GSOS language, and $\sim$ be any equivalence relation on processes satisfying $p \leftrightarrows q \Rightarrow p \sim q$. Then $p^* \sim q^*$ iff $p \sim q$.

Proof: We have $p \leftrightarrows p^*$ for all $p \in T(\Sigma)$, because the relation $\{(p, p^*), (p^*, p) \mid p \in T(\Sigma)\}$ is a strong bisimulation by Lemma 2. Hence if $p \sim q$ then $p^* \leftrightarrows p \sim q \leftrightarrows q*$, implying $p^* \sim q^*$, and if $p^* \sim q^*$ then $p \leftrightarrows p^* \sim q^* \leftrightarrows q$, implying $p \sim q$. $\Box$

Corollary 2 Let $\mathcal{L}$ be a two-tiered GSOS language such that $\sim$ is a congruence on $\mathcal{L}^*$, for $\sim$ an equivalence relation satisfying $p \leftrightarrows q \Rightarrow p \sim q$. Then $\sim$ is a congruence on $\mathcal{L}$.

Proof: Let $ar(f) = n$. Suppose $p_i \sim q_i$ for $i = 1, ..., n$. By Corollary 1, $p_i^* \sim q_i^*$ for $i = 1, ..., n$. By assumption, $(f(p_1, ..., p_n))^* \sim (f(q_1, ..., q_n))^*$. Thus, by Corollary 1, $f(p_1, ..., p_n) \sim f(q_1, ..., q_n)$. $\Box$

Proof of Theorem 3: Let $\mathcal{L}$ be XB cool, with X$\in$ {W,D,H,B}. By Observation 1, $\mathcal{L}^*$ is simply XB cool. So by Theorem 2, $\leftrightarrows_X$ is a congruence on $\mathcal{L}^*$. Apply Corollary 2. $\Box$

In fact, Theorem 3 can be obtained as an instance of a more general theorem.

Theorem 4 Let $\sim$ be any equivalence relation on processes defined in terms of the transitions between them, satisfying $p \leftrightarrows q \Rightarrow p \sim q$, and let F be a format on transition system specifications, so that on any language in F-format, $\sim$ is a congruence. Define a two-tiered language $\mathcal{L}$ to be in the *two-tiered* F-format iff $\mathcal{L}^*$ is in F-format. Now $\sim$ is a congruence on any language in the two-tiered F-format.

This follows by the very same proof as of Theorem 3. Note that the XB cool format is exactly the two-tiered simply XB cool format, for $X \in \{W, D, B, H\}$.

5.1 Bifurcation rules

Let $\mathcal{L}$ be BB cool. If argument $f(i)$ of operator f is active, then argument i of $f^\star$ must also be active, so Clause 3 of Definition 10 says that $f^\star$ has a patience rule $\frac{x_i \xrightarrow{\tau} y}{f^\star(x_1,...,x_n) \xrightarrow{\tau} f^\star(x_1,...,x_n)[y/x_i]}$.

By Definition 9 this implies that f must have a rule $\frac{x_{f(i)} \xrightarrow{\tau} y}{f(x_1,...,x_{ar(f)}) \xrightarrow{\tau} \sigma_f(f^\star(x_1,...,x_n)[y/x_i])}$.
This rule is called a *bifurcation rule* of f [5].
By Clause 2 of Definition 10, bifurcation rules are the only rules of f with τ-premises.

5.2 A small extension

Say that an argument i of an operator f is *ignored* if $f^\star$ has no argument k with $f(k) = i$. In that case there can be no rule with source $f(x_1, \ldots, x_{ar(f)})$ with x_i in its premises or in its target. A subterm u of a term t is *irrelevant* if occurs within an ignored argument v_i of a subterm $f(v_1, \ldots, v_{ar(f)})$ of t. Now Definition 7 of an argument of an operator being receiving may be strengthened by replacing "a subterm $f(v_1, \ldots, v_n)$ with x occurring in v_i" by "a relevant subterm $f(v_1, \ldots, v_n)$ with x a relevant subterm of v_i". This yields a slight weakening of Clause 4 in Definition 10. The weakened clause is still sufficient to obtain Observation 1 and hence Theorem 3.

Example 9 Let $\mathcal{L}$ have a rule $\dfrac{x_1 \xrightarrow{a} y}{g(x_1) \xrightarrow{a} f(h(f(x_1,y)),k(y))}$. By Definition 7 both the arguments of h and k are receiving, so Clause 4 in Definition 10 demands patience rules for both $h^\star$ and $k^\star$. Now suppose that $h^\star = h$, $k^\star = k$, $ar(f^\star) = 1$ and $\sigma_f(x_1) = x_1$. This means that $f(x_1, x_2)$ is an abbreviation for $f^\star(x_1)$ and the second argument of f is ignored. In such a case one has $f(p,q) \leftrightarrows f(p,r)$ for all closed terms p, q and r. Now the weakened Clause 4 does not demand a patience rule for either $h^\star$ or $k^\star$, since the arguments of h and k are no longer receiving. Namely, y is an irrelevant subterm of $f(x_1,y)$ and $k(y)$ is an irrelevant subterm of $f(h(f(x_1,y)),k(y))$.

The remainder of this section establishes a result that helps to relate my formulation of the cool formats to Bloom's [5]; for all other purposes it can be skipped. Proposition 1 below uses the concept of a *ruloid*; please consult Definition 14 and Example 10 in the next section first.

Above a definition of an argument of an operator being receiving was proposed that is more restrictive than Definition 7. This definition can be rephrased as follows.

Definition 12 Let $\mathcal{L}$ be a two-tiered GSOS language. Define the translation $^\star : T(\Sigma) \to T(\Sigma)$ recursively by $x^\star = x$ for $x \in V$ and $(f(t_1,\ldots,t_n))^\star = f^\star(t^\star_{f(1)},\ldots,t^\star_{f(n)})$.

An argument $i \in \mathbb{N}$ of an operator f is *truly receiving* if a variable x is receiving in a term t that has a subterm $v = f(v_1,\ldots,v_n)$ with $v^\star$ a subterm of $t^\star$ and $x \in var(v^\star_i)$.

Here the requirement "$v^\star$ is a subterm of $t^\star$" is equivalent to "v is a relevant subterm of t" and "$x \in var(v^\star_i)$" to "x is a relevant subterm of v_i". Note that for any term u one has $var(u^\star) \subseteq var(u)$, but not always $var(u^\star) = var(u)$. In Example 9, $y \notin var(f(x_1,y)^\star) = var(f^\star(x_1)) = \{x_1\}$ and $k(y)^\star = k(y)$ is not a subterm of $f(h(f(x_1,y)),k(y))^\star = f^\star(h(f(x_1,y))^\star)$.

The weakened version of Clause 4 of Definition 10 that was proposed above reads:

4^{G}. if argument $f(i)$ of operator f is truly receiving, then argument i of $f^\star$ has a patience rule.

Here I reformulate this clause in such a way that it resembles the corresponding requirement in Bloom [5].

Definition 13 A term $u \in \mathbb{T}(\Sigma)$ is *univariate* if no variable occurs more than once in u.

Trivially, any term can be written as $\sigma(u)$ with u a univariate term and $\sigma : V \to V$.

Proposition 1 In the definition of WB and HB cool GSOS languages, Clause 4^{G} is equivalent to

4^{B}. if $x \in var(t)$ is receiving in t, u is a univariate term, $y \in var(u)$, and $\sigma : V \to V$ is a substitution such that $\sigma(u) = t^\star$ and $\sigma(y) = x$, then there is an $\mathcal{L}$-ruloid $\frac{y \xrightarrow{\tau} z}{u \xrightarrow{\tau} u[z/y]}$.

Proof: $4^{\mathrm{G}} \Rightarrow 4^{\mathrm{B}}$: Suppose $\mathcal{L}$ satisfies Clause 4^{G}, $x \in var(t)$ is receiving in t, u is a univariate term, $y \in var(u)$, and $\sigma : V \to V$ is a substitution such that $\sigma(u) = t^\star$ and $\sigma(y) = x$. With structural induction on subterms w of u that contain y, I show that there is an $\mathcal{L}$-ruloid $\frac{y \xrightarrow{\tau} z}{w \xrightarrow{\tau} w[z/y]}$. For every such w there is a subterm v of t with $\sigma(w) = v^\star$ a subterm of $t^\star$ and $x \in var(t^\star)$.

Base case: $w = y$. By definition, there is a ruloid $\frac{y \xrightarrow{\tau} z}{y \xrightarrow{\tau} z}$.

Induction step: Let $v = f(v_1,\ldots,v_{ar(f)})$ and $w = f^\star(w_1,\ldots,w_n)$, and let $i \in \{1,\ldots,n\}$ be the unique argument of $f^\star$ with $y \in var(w_i)$. Then $\sigma(w_i) = v^\star_{f(i)}$, so $x \in var(v^\star_{f(i)}) \subseteq var(v_{f(i)})$. Thus

$f(i)$ is a truly receiving argument of f, and $\mathcal{L}$ has a patience rule $\frac{x_i \xrightarrow{\tau} z'}{f^\star(x_1,\dots,x_n) \xrightarrow{\tau} f^\star(x_1,\dots,x_n)[z'/x_i]}$. By induction, there is an $\mathcal{L}$-ruloid $\frac{y \xrightarrow{\tau} z}{w_i \xrightarrow{\tau} w_i[z/y]}$, so by Definition 14 one obtains the required $\mathcal{L}$-ruloid $\frac{y \xrightarrow{\tau} z}{f^\star(w_1,\dots,w_n) \xrightarrow{\tau} f^\star(w_1,\dots,w_n)[z/y]}$.

$4^{\rm B} \Rightarrow 4^{\rm G}$: Suppose $\mathcal{L}$ satisfies Clause $4^{\rm B}$, and argument $f(i)$ of operator f is truly receiving. Then there must be a term t with a subterm $v = f(v_1,\dots,v_{ar(f)})$ such that $v^\star$ is a subterm of $t^\star$ and variable $x \in var(v^\star_{f(i)})$ is receiving in t. Let u be a univariate term, and $\sigma : V \rightarrow V$ a substitution such that $\sigma(u) = t^\star$. Let $w = f^\star(w_1,\dots,w_n)$ be the subterm of u with $\sigma(w) = v^\star$. Then $\sigma(w_i) = v^\star_{f(i)}$, and w_i (and hence u) must contain a variable y with $\sigma(y) = x$. By Clause $4^{\rm B}$ there is an $\mathcal{L}$-ruloid $\frac{y \xrightarrow{\tau} z}{u \xrightarrow{\tau} u[z/y]}$. By Clause 2 of Definition 10, patience rules are the only rules for the operators of u with τ-premises. Hence, by Definition 14, this ruloid can only be obtained by stacking patience rules. As u is univariate, its subterm w_i contains the only occurrence of y in u, so one of the patience rules applied must be the one for argument i of $f^\star$. □

6 Rooted cool GSOS languages

In this section I will define the (simply) RWB, RDB, RHB and RBB cool rule formats and establish the associated congruence theorems. In order to formulate the requirements for the RWB and RDB cool GSOS languages I need the concept of a *ruloid*, this being a kind of derived GSOS rule.

Definition 14 For r transition rule, let RHS(r) denote the set of right-hand sides of its premises. Let $\mathcal{L}$ be a positive GSOS language. The class of *$\mathcal{L}$-ruloids* is the smallest set of rules such that

- $\frac{x \xrightarrow{a} y}{x \xrightarrow{a} y}$ is an $\mathcal{L}$-ruloid, for every $x, y \in V$ and $a \in Act$;
- if σ is a substitution, $\mathcal{L}$ has a rule $\frac{H}{s \xrightarrow{a} t}$, and for every premise $x \xrightarrow{c} y$ in H there is an $\mathcal{L}$-ruloid $r_y = \frac{H_y}{\sigma(x) \xrightarrow{c} \sigma(y)}$ such that the sets RHS(r_y) are pairwise disjoint and each RHS(r_y) is disjoint with $var(\sigma(s))$, then the transition rule $\frac{\bigcup_{y \in H} H_y}{\sigma(s) \xrightarrow{a} \sigma(t)}$ is an $\mathcal{L}$-ruloid.

Note that a transition α, seen as a rule $\frac{\emptyset}{\alpha}$, is an $\mathcal{L}$-ruloid iff it is generated by $\mathcal{L}$ in the sense of Definition 4. The left-hand sides of premises of a ruloid are variables that occur in its source, and the right-hand sides are variables that are all distinct and do not occur in its source. Its target only contains variables that also occur elsewhere in the rule.

Example 10 Let $\mathcal{L}$ contain the rule $\dfrac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{b} y_2}{f(x_1, x_2) \xrightarrow{a} g(x_1, y_1)}$.

Then $\mathcal{L}$ has ruloids $\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y'}{f(x,y) \xrightarrow{a} g(x,x')}$ and $\dfrac{x \xrightarrow{a} x' \quad y \xrightarrow{b} y' \quad z \xrightarrow{b} z'}{f(f(x,y),z) \xrightarrow{a} g(f(x,y), g(x,x'))}$.

Lemma 3 Suppose an $\mathcal{L}$-ruloid has a premise with right-hand side x and a target t containing a subterm $f(t_1,\dots,t_n)$ with $x \in var(t_i)$. Then i is a receiving argument of f.

Proof: By Definition 14 there must be terms $w_1,\dots,w_n$, variables $y_1,\dots,y_n$, and substitutions $\sigma_1,\dots,\sigma_n$, such that $t = \sigma_1(w_1)$, y_i is receiving in w_i $(i=1,\dots,n)$, $\sigma_i(y_i) = \sigma_{i+1}(w_{i+1})$ $(i=1,\dots,n-1)$, and $\sigma_n(y_n) = x$; and moreover one of the w_k has a subterm $f(w_k^1,\dots,w_k^m)$ with $y_k \in var(w_k^i)$. It follows that i is an receiving argument of f. □

Definition 15 A GSOS language $\mathcal{L}$ is *RWB cool* if the operators can be partitioned in *tame* and *wild* ones, such that

1. the target of every rule contains only tame operations;
2. the sublanguage $\mathcal{L}^{tame}$ of tame operators in $\mathcal{L}$ is WB cool;
3. $\mathcal{L}$ is positive, and for each rule $\frac{H}{s \xrightarrow{a} t}$ there is a term u and a substitution $\sigma : var(u) \rightarrow var(s)$ such that
   - there is an $\mathcal{L}$-ruloid $\frac{K}{u \xrightarrow{a} v}$ with $\sigma(K) = H$ and $\sigma(v) = t$,
   - and for every premise $x \xrightarrow{c} y$ in K, $\mathcal{L}$ has a rule $\frac{\sigma(x) \xrightarrow{\tau} y}{s \xrightarrow{\tau} \sigma(u[y/x])}$;

(4. if argument $f(i)$ of operator f is receiving in $\mathcal{L}$, then argument i of $f^\star$ has a patience rule.)

The formats *RDB cool*, *RHB cool* and *RBB cool* are defined likewise, adapting "WB cool" in the second clause appropriately, but skipping the third clause for RHB and RBB, and the last one for RDB and RBB. The last clause cannot be skipped for RHB. The *simply RXB cool* rule formats (X$\in${W,D,H,B}) are obtained by requiring the sublanguage of tame operators to be simply XB cool.

Note that in the third clause, u, σ and the ruloid can always be chosen in such a way that $v = t$. The instance of this clause with $s = f(x_1, \ldots, x_{ar(f)})$ for a tame operator f is redundant, as it vacuously holds when taking $u^\star = f^\star(x_1, \ldots, x_{ar(f^\star)})$ and $\sigma = \sigma_f$; the rule required in the second subclause is then the bifurcation rule derived in Section 5.1.

The last clause above appeared before as Clause 4 in Definition 10 of the WB and HB cool formats. Given that a term with a receiving variable cannot contain wild operators, this clause is almost implied by Clause 2 above. All it adds, is that the requirement of Clause 4 for the sublanguage of tame operators applies to "receiving in $\mathcal{L}$" instead of merely "receiving in $\mathcal{L}^{tame}$". Thus, the rules for the wild operators help determine which variables in a term t count as receiving.

Proposition 2 In the definition of the RWB cool format above, the last clause is redundant.

Proof: Let argument i of operator f be $\mathcal{L}$-receiving. I will show that it is already $\mathcal{L}^{tame}$-receiving. Let t be a term with a subterm $f(t_1, \ldots, t_n)$ such that $y \in var(t_i)$ and y is $\mathcal{L}$-receiving in t. So there is a rule $r = \frac{H}{f(x_1,\ldots,x_n) \xrightarrow{a} t}$ in $\mathcal{L}$ with y occurring as the right-hand side of a premise in H. Let u and σ be the term and substitution that must exists for r by the third clause of Definition 15. Definition 15 implies that the operators in u are tame, and there is an $\mathcal{L}$-ruloid, hence an $\mathcal{L}^{tame}$-ruloid, $\frac{K}{u \xrightarrow{a} v}$ with $\sigma(K) = H$ and $\sigma(v) = t$. Let $f(v_1, \ldots, v_n)$ be the subterm of v with $\sigma(v_i) = t_i$ for $i = 1, \ldots, n$. By the third clause of Definition 3, $y \notin range(\sigma)$. Hence, as $y \in var(\sigma(v_i))$, $y \in var(v_i)$. Given that $dom(\sigma) = var(u)$, σ does not effect the right-hand sides of K, so y is the right-hand side of a premise in K. By Lemma 3, i is an $\mathcal{L}^{tame}$-receiving argument of f. □

When working with the slightly stronger definition of receiving contemplated in Section 5.2, the proof above remains valid with trivial adaptations.

Now I will prove the remaining lemmas of Section 3, thereby completing the proof of Theorem 2.

Proof of Lemma RHB: Let $\mathcal{L}$ be simply RHB cool, let $r = \frac{H}{s \xrightarrow{a} t}$ be a rule in $\mathcal{L}$, and let ν be a closed substitution such that $\nu(x) \xrightarrow{c}\Longrightarrow \nu(y)$ for each positive premise $x \xrightarrow{c} y$ in H and $\neg\exists r\,(\nu(x) \overset{c}{\not\longrightarrow} r)$ for each negative premise $x \overset{c}{\not\longrightarrow}$ in H. I need to show that $\nu(s) \xrightarrow{a}\Longrightarrow \nu(t)$. By assumption, there is a closed substitution μ such that

- $\nu(x) \xrightarrow{c} \mu(y) \Longrightarrow \nu(y)$ for each premise $x \xrightarrow{c} y$ in H, and
- $\nu(x) = \mu(x)$ for all variables x that do not occur as right-hand sides of premises in H.

Thus $\nu(s) = \mu(s) \xrightarrow{a} \mu(t)$ by application of r, and $\mu(t) \Longrightarrow \nu(t)$ by Lemma 1. $\Box$

Lemma 4 Let $\mathcal{L}$ be simply WB cool, let $r = \frac{H}{s \xrightarrow{a} t}$ be an $\mathcal{L}$-ruloid, and let ν be a closed substitution. If for each premise $x \xrightarrow{c} y$ in H one has $\nu(x) \Longrightarrow\xrightarrow{c}\Longrightarrow \nu(y)$, then $\nu(s) \Longrightarrow\xrightarrow{a}\Longrightarrow \nu(t)$.

Proof: The case that r is a rule in $\mathcal{L}$ is proven exactly as in Lemma WB, just writing a for (a), etc. The general case now follows by a straightforward structural induction on s. $\Box$

Proof of Lemma RWB: Let $\mathcal{L}$ be simply RWB cool, let $r = \frac{H}{s \xrightarrow{a} t}$ be a rule in $\mathcal{L}$, and let ν be a closed substitution such that $\nu(x) \Longrightarrow\xrightarrow{c}\Longrightarrow \nu(y)$ for each premise $x \xrightarrow{c} y$ in H. I need to show that $\nu(s) \Longrightarrow\xrightarrow{a}\Longrightarrow \nu(t)$.
Case 1: Suppose that there is a closed substitution μ such that

- $\nu(x) \xrightarrow{c} \mu(y) \Longrightarrow \nu(y)$ for each premise $x \xrightarrow{c} y$ in H, and
- $\nu(x) = \mu(x)$ for all variables x that do not occur as right-hand sides of premises in H.

Then $\nu(s) = \mu(s) \xrightarrow{a} \mu(t)$ by application of r, and $\mu(t) \Longrightarrow \nu(t)$ by Lemma 1.
Case 2: Suppose that there is a premise $x^0 \xrightarrow{c} y^0$ in H such that $\nu(x^0) \xrightarrow{\tau} p \Longrightarrow\xrightarrow{c}\Longrightarrow \nu(y^0)$ for a closed term p. Let u, σ and $r' = \frac{K}{u \xrightarrow{a} v}$ be the term, substitution and ruloid that exists for r by the third clause of Definition 15, and let x^1 be the unique variable in u such that K has a premise $x^1 \xrightarrow{c} y^0$ (using that $\sigma(y^0) = y^0$). Hence $\sigma(x^1) = x^0$. Let μ be the closed substitution with $\mu(y^0) = p$ and $\mu(z) = \nu(z)$ for all variables $z \neq y^0$. Now $\mu(\sigma(x^1)) = \mu(x^0) = \nu(x^0) \xrightarrow{\tau} \mu(y^0)$. By Clause 3 of Definition 15, $\mathcal{L}$ has a rule $\frac{\sigma(x^1) \xrightarrow{\tau} y^0}{s \xrightarrow{\tau} \sigma(u[y^0/x^1])}$; hence $\nu(s) = \mu(s) \xrightarrow{\tau} \mu(\sigma(u[y^0/x^1]))$. Let ρ be the closed substitution with $\rho(x^1) = p$ and $\rho(z) = \nu(\sigma(z))$ for all variables $z \neq x^1$. Then $\mu(\sigma(u[y^0/x^1]) = \rho(u)$, the operators in u are tame, and $\rho(x) \Longrightarrow\xrightarrow{c}\Longrightarrow \rho(y)$ for each premise $x \xrightarrow{c} y$ in K. Lemma 4 yields $\rho(u) \Longrightarrow\xrightarrow{a}\Longrightarrow \rho(v)$. By Clause 5 of Definition 8, $x^1 \notin \mathit{var}(v)$, so $\rho(v) = \nu(\sigma(v)) = \nu(t)$. Thus $\nu(s) = \mu(s) \xrightarrow{\tau} \mu(\sigma(u[y^0/x^1])) = \rho(u) \Longrightarrow\xrightarrow{a}\Longrightarrow \rho(v) = \nu(t)$. $\Box$

The **Proof of Lemma RDB** proceeds likewise, using a DB cool counterpart of Lemma 4. $\Box$

This completes the **Proof of Theorem 2.** $\Box$

Example 11 The following fragment of CCS has the constant 0, unary operators $a.\_$, binary operators $+$ and $\|$ (usually written $|$), and instances of the GSOS rules below. Here a ranges over $\mathit{Act} = \mathcal{N} \dot\cup \overline{\mathcal{N}} \dot\cup \{\tau\}$ with $\mathcal{N}$ a set of *names* and $\overline{\mathcal{N}} = \{\overline{a} \mid a \in \mathcal{N}\}$ the set of *co-names*. The function $\bar{\cdot}$ extends to $\mathcal{N} \cup \overline{\mathcal{N}}$ (but not to Act) by $\overline{\overline{a}} = a$.

$$\frac{x_1 \xrightarrow{a} y_1}{x_1 + x_2 \xrightarrow{a} y_1} \qquad \frac{x_2 \xrightarrow{a} y_2}{x_1 + x_2 \xrightarrow{a} y_2} \qquad a.x_1 \xrightarrow{a} x_1$$

$$\frac{x_1 \xrightarrow{a} y_1}{x_1 \| x_2 \xrightarrow{a} y_1 \| x_2} \qquad \frac{x_2 \xrightarrow{a} y_2}{x_1 \| x_2 \xrightarrow{a} x_1 \| y_2} \qquad \frac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{\overline{a}} y_2}{x_1 \| x_2 \xrightarrow{\tau} y_1 \| y_2}$$

The sublanguage without the $+$ is simply WB cool, and the entire GSOS language is simply RWB cool. Clause 3 of Definition 15 applied to the i^{th} rule for the $+$ is satisfied by taking $u = x$, $\sigma(x) = x_i$, and the ruloid $\frac{x \xrightarrow{a} y_i}{x \xrightarrow{a} y_i}$; indeed the language has a ruloid $\frac{x_i \xrightarrow{\tau} y_i}{x_1 + x_2 \xrightarrow{\tau} y_i}$.

Example 12 Extend the GSOS language of Example 11 with the binary operators $\|\!\!\_$ and $|$ with rules $\dfrac{x_1 \xrightarrow{a} y_1}{x_1 \|\!\!\_\, x_2 \xrightarrow{a} y_1 \| y_2}$ and $\dfrac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{\overline{a}} y_2}{x_1 | x_2 \xrightarrow{\tau} y_1 \| y_2}$ with a ranging over Act. These are the *left merge* and the *communication merge* of Bergstra & Klop [3] but adapted to fit (the communication format of) CCS rather then ACP. (I wrote $\|$ for the CCS parallel composition to avoid name clashes with this communication merge.) In [3] these operators are introduced in order to give a finite complete equational axiomatisation of strong bisimulation equivalence on the language ACP, and the same approach works for CCS.

The first argument of $\|\!\!\_$ and both arguments of $|$ are active, whereas no arguments of $\|\!\!\_$ and $|$ are receiving. Since the active arguments of $\|\!\!\_$ and $|$ do not have patience rules, both operators need to be classified as wild. Requirements 1, 2 and 4 of Definition 15 are clearly satisfied. Hence this GSOS language is RHB cool, and $\leftrightarrows_{r\eta}$ and $\leftrightarrows_{rb}$ are congruences on the entire language.

Requirement 3 applied to the rule for the $\|\!\!\_$ is satisfied by taking $u = x \| x_2$, $\sigma(x) = x_1$, $\sigma(x_2) = x_2$ and the ruloid $\dfrac{x \xrightarrow{a} y_1}{x \| x_2 \xrightarrow{a} y_1 \| x_2}$. Indeed the language has a ruloid $\dfrac{x_1 \xrightarrow{\tau} y_1}{x_1 \|\!\!\_\, x_2 \xrightarrow{\tau} y_1 \| x_2}$. Hence the language with $\|\!\!\_$ but without $|$ is RWB cool, guaranteeing that also $\leftrightarrows_{rw}$ and $\leftrightarrows_{rd}$ are congruences. However, requirement 3 applied to either rule of the communication merge is not satisfied. In fact $\leftrightarrows_{rw}$ and $\leftrightarrows_{rd}$ fail to be congruences for this operator: one has $\tau.a.0 \leftrightarrows_{rd} \tau.a.0 + a.0$ but

$$0 \leftrightarrows (\tau.a.0 | \overline{a}.b.0) \not\leftrightarrows_{rw} ((\tau.a.0 + a.0) | \overline{a}.b.0) \leftrightarrows \tau.b.0.$$

This shows that this part of requirement 3 of Definition 15 cannot be skipped.

Example 13 The GSOS language with 0, action prefix operators $a.P$ for $a \in Act$ and the unary operator np with the rule $\frac{x_1 \overset{a}{\not\longrightarrow}}{np(x_1) \xrightarrow{c} 0}$ (for specific actions a, c) is not positive, but otherwise meets the requirements of Definition 15. Yet, $\leftrightarrows_{rw}$ and $\leftrightarrows_{rd}$ fail to be congruences on this language. Namely $\tau.a.0 \leftrightarrows_{rd} \tau.a.0 + a.0$ yet $np(\tau.a.0) \leftrightarrows c.0 \not\leftrightarrows_{w} 0 \leftrightarrows np(\tau.a.0 + a.0)$. This shows that the requirement that $\mathcal{L}$ be positive cannot simply be skipped in the RWB and RDB cool formats.

Negative premises of wild operators are allowed in the RHB and RBB cool formats. This is possible because the first transitions of rooted branching (or η-)bisimilar processes need to be matched without preceding τ-transitions, just as for strong bisimulation. A good example of an RBB cool GSOS operator with negative premises is the initial priority operator of [8].

Example 14 The GSOS language $\mathcal{L}$ with 0, $\Box$, $\sqcap$, action prefix operators $a.P$ for $a \in Act$ and the unary operators f, g with rules $\frac{x_1 \xrightarrow{a} y_1}{f(x_1) \xrightarrow{a} g(y_1)}$ and $g(x) \xrightarrow{c} x$ (for specific actions a, c) satisfies requirements 1 and 2 of Definition 15, provided one classifies the argument of f as wild and that of g as tame. Hence $\leftrightarrows_{rb}$ is a congruence on $\mathcal{L}$. However, $\mathcal{L}$ fails requirements 3 and 4. Here the argument of g is receiving in $\mathcal{L}$ as a whole, although not in $\mathcal{L}^{tame}$. In fact, $\leftrightarrows_{rw}$ (, $\leftrightarrows_{rd}$) and $\leftrightarrows_{r\eta}$ fail to be congruences on $\mathcal{L}$. One has $a.(b.0 \sqcap d.0) \leftrightarrows_{\eta} a.(b.0 \sqcap d.0) \Box a.b.0$, yet

$$f(a.(b.0 \sqcap d.0)) \leftrightarrows a.c.(b.0 \sqcap d.0) \not\leftrightarrows_{w} a.c.(b.0 \sqcap d.0) \Box a.c.b.0 \leftrightarrows f(a.(b.0 \sqcap d.0) \Box a.b.0)$$

as in Example 6. Thus requirement 4 cannot be skipped for the RHB cool format.

Theorem 5 On any RWB cool GSOS language, $\leftrightarroweq_{rw}$ is a congruence.

On any RDB cool GSOS language, $\leftrightarroweq_{rd}$ is a congruence.

On any RHB cool GSOS language, $\leftrightarroweq_{r\eta}$ is a congruence.

On any RBB cool GSOS language, $\leftrightarroweq_{rb}$ is a congruence.

Proof: Let $\mathcal{L}$ be RBB cool. Regard $\mathcal{L}$ as a two-tiered GSOS languages by classifying all wild operators as principal ones. The GSOS language $\mathcal{L}^*$ constructed in Definition 11 is simply RBB cool, by Observation 1, so by Theorem 2RBB $\leftrightarroweq_{rb}$ is a congruence on $\mathcal{L}^*$. Apply Corollary 2.

The other cases go likewise, except that in checking that $\mathcal{L}^*$ is simply RWB or RDB cool, one has to check that Clause 3 of Definition 15 is satisfied. Let u and σ be a term and substitution that satisfy Clause 3 for a rule $\frac{H}{f(x_1,\ldots,x_n)\xrightarrow{a}t}$ with f wild. I claim that u^* and σ are appropriate for the rule $\frac{H}{f(x_1,\ldots,x_n)\xrightarrow{a}t^*}$, existing in $\mathcal{L}^*$. Namely, by a straightforward structural induction on u, if $\frac{K}{u\xrightarrow{a}v}$ is an $\mathcal{L}$-ruloid then $\frac{K}{u^*\xrightarrow{a}v^*}$ is an $\mathcal{L}^*$-ruloid. Moreover, $\sigma(v)=t$ implies $\sigma(v^*)=t^*$. By construction, for every premise $x\xrightarrow{c}y$ in K, $\mathcal{L}^*$ has a rule $\frac{\sigma(x)\xrightarrow{\tau}y}{f(x_1,\ldots,x_n)\xrightarrow{\tau}\sigma(u^*[y/x])}$. □

As the above shows, for $X\in\{W,D,B,H\}$, the RXB cool format is exactly the two-tiered simply RXB cool format, so Theorem 5 is obtained as an application of Theorem 4 to Theorem 2.

7 Comparison with Bloom's formats

Not counting a host of notational differences, Bloom's definitions of the cool formats differ in five ways from mine.

First of all Bloom requires *bifurcation rules* for all operators in $\mathcal{L}^{tame}$, whereas I merely require patience rules for the principal operators. As principal operators in $\mathcal{L}^{tame}$ are straight, and bifurcation rules for straight operators are exactly patience rules, the difference is that I dropped the bifurcation requirement for abbreviations (non-principal operators). This is possible, because by Definition 9, which corresponds to Definition 3.5.5 in [5], the rules for the abbreviations are completely determined by the rules for their straightenings, and it turns out that a bifurcation rule of an abbreviation f is exactly what is determined by the corresponding patience rule for its straightening $f^\star$.

Bloom's WB cool format requires the existence of bifurcation/patience ruloids for receiving variables in any term, whereas I require them for receiving arguments of operators, which is a more syntactic and easy to check requirement. The two approaches are shown equivalent in Proposition 1 when using the extension of my formats of Section 5.2, this being the reason behind that extension.

Bloom's WB and RWB cool formats use a so-called *ε-presentation.* This entails that rules may have premises of the form $x\xrightarrow{\varepsilon}y$. In terms of Definition 4, the meaning of such premises is given by the requirement that $\sigma(x)=\sigma(y)$ for $(x\xrightarrow{\varepsilon}y)\in H$. By using ε-premises, any rule can be given a form in which the target is a univariate term, having no variables in common with the source. This allows a simplification of the statement of the bifurcation ruloids. Any ε-presented GSOS language can be converted to ε-free form by substitution, in each rule r, x for y for every premise $x\xrightarrow{\varepsilon}y$ of r.

$$\begin{tikzcd} s \arrow[r, "a"] \arrow[d, "\tau"'] \arrow[dr, "a"] & t \arrow[d, "\tau"] \\ t_1 \arrow[r, "a"'] \arrow[ur, "a"] \arrow[loop left, "\tau"] & t_2 \arrow[loop right, "\tau"] \end{tikzcd}$$

Bloom's rendering of the RWB cool format doesn't feature Clause 4 (and in view of Proposition 2, neither does mine), but Clause 3 is much more involved. For every rule with conclusion $s \xrightarrow{a} t$ Bloom requires the existence of two terms t_1 and t_2 and seven types of derived operational rules, such that the diagram on the right commutes. My Clause 3 stems from the observation that, given Bloom's other restrictions, t necessarily has the rules required for t_2, so that one may always choose $t_2 = t$. This leaves only t_1 (called u in Definition 15) and three types of rules, one of which (the t_1-loop in the diagram above) is in fact a bifurcation rule whose existence is already implied by the requirements of Definition 10.

In Clause 3 of Definition 15, Bloom requires that

$$var(u) = \{y' \mid y \in var(t)\} \text{ and } \sigma(y') = \begin{cases} x \text{ if } H \text{ contains a premise } x \xrightarrow{c} y \\ y \text{ otherwise.} \end{cases} \tag{6}$$

In order to match Bloom's format I could have done the same, but this condition is not needed in the proof and reduces the generality of the format.

Example 15 Consider the following GSOS language, with unary operators f and g and constant 0.

$$f(x_1) \xrightarrow{b} x_1 \qquad \frac{x_1 \xrightarrow{a} y_1}{f(x_1) \xrightarrow{a} 0} \qquad \frac{x_1 \xrightarrow{\tau} y_1}{f(x_1) \xrightarrow{\tau} g(y_1)} \qquad \frac{x_1 \xrightarrow{a} y_1}{g(x_1) \xrightarrow{a} 0} \qquad \frac{x_1 \xrightarrow{\tau} y_1}{g(x_1) \xrightarrow{\tau} g(y_1)}$$

The sublanguage without f is simply WB cool, and the entire language simply RWB cool. In fulfilling Clause 3 of Definition 15 take for the first rule $u := f(x_1)$ and for the second and third rules $u := g(x_1)$, in each case with σ the identity.

However, this language is not RWB cool in the sense of Bloom, as the restriction above forbids rules $\frac{H}{s \xrightarrow{a} t}$ with $H \neq \emptyset$ and $var(t) = \emptyset$, where the operator in s is wild.

Proposition 3 A GSOS language is WB cool, respectively RWB, BB or RBB cool, as defined here, with the extension of Section 5.2 and the restriction (6) above, iff it is WB cool, resp. RWB, BB or RBB cool, as defined in Bloom [5].

The proof consists of converting Bloom's rule formats to ε-free form and eliminating a number of other notational differences, together with the issues addressed above. □

Moreover, my proofs that cool languages are compositional for bisimulation equivalences greatly simplify the ones of Bloom [5] by using a reduction of the general case to the simple case, instead of treating the general formats directly.

8 Cool GSOS languages with lookahead

The RWB cool format can be extended by allowing wild operators f, besides GSOS rules satisfying Clause 3 of Definition 15, also to have rules of which all premises have the form $x \Longrightarrow\xrightarrow{c} y$ with $c \in A$. For such rules Clause 3 is not required, but in fulfilling Clause 4, they do count in determining which arguments are receiving. A similar extension applies to the RDB cool format.

Definition 16 Let Σ be a signature. A *GSOS rule with lookahead* is an expression of the form $\frac{H}{\alpha}$ with α a positive Σ-literal (the *conclusion*) and H a set of expressions of the form $x \Longrightarrow\xrightarrow{a} y$ (the *premises* of the rule), satisfying the four conditions of Definition 3. A *GSOS language with lookahead* is a TSS whose rules are either GSOS rules or GSOS rules with lookahead. The transitions generated by a GSOS language with lookahead $\mathcal{L}$ are defined exactly as in Definition 4, where a premise $(x_i \Longrightarrow\xrightarrow{c} y) \in H$ gives rise to a hypothesis $p_i \Longrightarrow_{\mathcal{L}}\xrightarrow{c}_{\mathcal{L}} \sigma(y)$, using the relational composition of $\Longrightarrow_{\mathcal{L}}$ and $\xrightarrow{c}_{\mathcal{L}}$. The notion of a *receiving* variable in term of Definition 7 extends straightforwardly to GSOS languages with lookahead.

Definition 17 A GSOS language with lookahead $\mathcal{L}$ is *RWB cool* if the operators can be partitioned in *tame* and *wild* ones, such that

1. the target of every rule contains only tame operations;
2. the sublanguage $\mathcal{L}^{tame}$ of tame operators in $\mathcal{L}$ is a WB cool GSOS language (without lookahead);
3. $\mathcal{L}$ is positive, and for each GSOS rule $\frac{H}{s\xrightarrow{a}t}$ (without lookahead) there is a term u and a substitution $\sigma : var(u) \rightarrow var(s)$ such that
   - there is an $\mathcal{L}$-ruloid $\frac{K}{u\xrightarrow{a}v}$ with $\sigma(K) = H$ and $\sigma(v) = t$,
   - and for every premise $x \xrightarrow{c} y$ in K, $\mathcal{L}$ has a rule $\frac{\sigma(x)\xrightarrow{\tau}y}{s\xrightarrow{\tau}\sigma(u[y/x])}$;
4. if argument $f(i)$ of operator f is receiving in $\mathcal{L}$, then argument i of $f^\star$ has a patience rule.

The format *RDB cool* is defined likewise, adapting "WB cool" in the second clause to "DB cool", but skipping the last clause. The *simply RXB cool* rule formats (X$\in$ {W,D}) are obtained by requiring the sublanguage of tame operators to be simply XB cool.

With these definitions, Lemmas RWB and RDB extend to the case that $\mathcal{L}$ is a GSOS language with lookahead, using premises $x \Longrightarrow\xrightarrow{x} y$ in case of GSOS rules with lookahead. Namely, the statements quantify over rules in $\mathcal{L}$; when such a rule is a GSOS rule, the proof given in Section 6 applies, and when it is a GSOS rule with lookahead, the statement follows from Definition 16 and Lemma 1.

Theorem 6 On any RWB cool GSOS language with lookahead, $\leftrightarroweq_{rw}$ is a congruence.
On any RDB cool GSOS language with lookahead, $\leftrightarroweq_{rd}$ is a congruence.

Proof: Using the same argument as in the proof of Theorem 3 I may restrict attention to simply RWB or RDB cool GSOS languages with lookahead. Here the proof of Section 3 applies again. □

Example 16 Extend the GSOS language of Example 11 with the binary operator $|$ and the single rule $\frac{x_1\Longrightarrow\xrightarrow{a}y_1 \quad x_2\Longrightarrow\xrightarrow{\bar{a}}y_2}{x_1|x_2\xrightarrow{\tau}y_1\|y_2}$. This is the variant of the *communication merge* of [3] (cf. Example 12), given in Bergstra & Klop [4] (and here again adapted to fit CCS rather then ACP$_\tau$). The result is a GSOS language with lookahead, that moreover is RWB cool. Thus $\leftrightarroweq_{rw}$ and $\leftrightarroweq_{rd}$ are congruences on this language. In [4] this variant of the communication merge is used as an auxiliary operator in giving a finite complete equational axiomatisation of $\leftrightarroweq_{rw}$ on a language with $\|$; this is not known to be possible using only auxiliary GSOS operators.

9 Turning GSOS Rules into Equations

This section recapitulates the method of [1] to provide finite equational axiomatisations of $\underline{\leftrightarrow}$ on an augmentation of any given GSOS language.

Definition 18 A process p, being a closed term in a GSOS language, is *finite* if there are only finitely many sequences of transitions $p \xrightarrow{a_1} p_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} p_n$. The length n of the longest sequence of this form is called the *depth* of p.

Definition 19 An *equational axiomatisation* Ax over a signature Σ is a set of equations $t = u$, called *axioms*, with $t, u \in \mathbb{T}(\Sigma)$. It *respects* an equivalence relation $\sim$ on $T(\Sigma)$ if $\sigma(t) \sim \sigma(u)$ for any closed substitution $\sigma : V \to T(\Sigma)$.

An *instance* of axiom $t = u$ is an equation $\sigma(C[t/x]) = \sigma(C[u/x])$ where σ is a substitution and C a term with $var(C) = \{x\}$, and x occurring only once in C. An equation $p = q$ is *derivable* from Ax, notation $p =_{\mathrm{Ax}} q$, if there is a sequence $p_0, \ldots, p_n$ of terms with $n \geq 0$ such that $p = p_0$, $q = p_n$ and for $i = 1, \ldots, n$ the equation $p_{i-1} = p_i$ is an instance of one of the axioms.

Ax is *sound* for $\sim$ if $p =_{\mathrm{Ax}} q$ implies $p \sim q$ for $p, q \in T(\Sigma)$. Ax is *complete for $\sim$ on finite processes* if $p \sim q$ implies $p =_{\mathrm{Ax}} q$ for finite processes p and q.

Note that Ax is sound for $\sim$ iff Ax respects $\sim$ and $\sim$ is a congruence.

Definition 20 A GSOS language $\mathcal{L}$ *extends BCCS* (*basic* CCS) if it contains the operators 0, $a.\_$ and $+$ of Example 11. A *basic process* is a closed term build from the operators mentioned above only. A *head normal form* is a closed term of the form $0 + a_1.p_1 + \cdots + a_n.p_n$ for $n \geq 0$. An axiomatisation on $\mathcal{L}$ is *head normalising* if any term $f(p_1, \ldots, p_{ar(f)})$ with the p_i basic processes can be converted into head normal form.

Proposition 4 Let $\mathcal{L}$ be a GSOS language extending BCCS, and Ax a head normalising equational axiomatisation, respecting $\underline{\leftrightarrow}$, and containing the axioms A1–4 of Table 1. Then Ax is sound and complete for $\underline{\leftrightarrow}$ on finite processes.

Proof: Using induction on the depth of p and a nested structural induction, the axioms can convert any finite process p into a basic process. Here one uses that strongly bisimilar processes have the same depth. Now apply the well-known fact that the axioms A1–4 are sound and complete for $\underline{\leftrightarrow}$ on basic processes [13]. $\square$

$x + (y + z)$	$=$	$(x + y) + z$	A1	$x \| y$	$=$	$x \mathbin{\|\!\lfloor} y + y \mathbin{\|\!\lfloor} x + x \mid y$	CM1
$x + y$	$=$	$y + x$	A2	$a.x \mathbin{\|\!\lfloor} y$	$=$	$a.(x \| y)$	CM2
$x + x$	$=$	x	A3	$0 \mathbin{\|\!\lfloor} y$	$=$	0	CM3
$x + 0$	$=$	x	A4	$(x + y) \mathbin{\|\!\lfloor} z$	$=$	$x \mathbin{\|\!\lfloor} z + y \mathbin{\|\!\lfloor} z$	CM4
				$a.x \mid \overline{a}.y$	$=$	$\tau.(x \| y)$	CM5
$a.(\tau.(x + y) + x)$	$=$	$a.(x + y)$	T1	$a.x \mid b.y$	$=$	0 (if $b \neq \overline{a}$)	CM6
$\tau.x + x$	$=$	$\tau.x$	T2	$0 \mid x$	$=$	$x \mid 0 = 0$	CM7
$a.(\tau.x + y) + a.x$	$=$	$a.(\tau.x + y)$	T3	$(x + y) \mid z$	$=$	$x \mid z + y \mid z$	CM8
				$x \mid (y + z)$	$=$	$x \mid y + x \mid z$	CM9

Table 1: Complete equational axiomatisations of BCCS and the parallel composition

For the parallel composition operator $\|$ of CCS no finite equational head normalising axiomatisation respecting strong bisimulation equivalence exists [14]. However, Bergstra & Klop [3] gave such an axiomatisation on the language obtained by adding two auxiliary operators, the *left merge* $\|\!\!\_$ and the *communication merge* $|$, with rules $\frac{x_1 \xrightarrow{a} y_1}{x_1 \|\!\!\_\, x_2 \xrightarrow{a} y_1 \| x_2}$ and $\frac{x_1 \xrightarrow{a} y_1 \quad x_2 \xrightarrow{\overline{a}} y_2}{x_1 | x_2 \xrightarrow{\tau} y_1 \| y_2}$, provided the alphabet *Act* of actions is finite. See Example 12. The axioms are CM1–9 of Table 1, in which $+$ binds weakest and $a.\_$ strongest, and a, b range over *Act*.

Aceto, Bloom & Vaandrager [1] generalise this idea to arbitrary GSOS languages with finitely many rules, each with finitely many premises, and assuming a finite alphabet *Act*. I recapitulate their method for positive languages only. A smooth operator (Definition 7) only has rules of the form $\frac{\{x_i \xrightarrow{c_i} y_i \mid i \in I\}}{f(x_1, \ldots, x_n) \xrightarrow{a} t}$. The *trigger* of such a rule is the partial function $\uparrow_r$: $\{i, \ldots, n\} \rightharpoonup Act$ given by $\uparrow_r(i) = c_i$ if $i \in I$, and $\uparrow_r(i)$ is undefined otherwise.

Definition 21 [1] A smooth GSOS operator f is *distinctive*, if no two rules of f have the same trigger, and the triggers of all rules of f have the same domain.

All operators of CCS, as well as $\|\!\!\_$ and $|$, are smooth. The operators 0, $a.\_$, $\|\!\!\_$ and $|$ are distinctive, but $\|$ is not. Its triggers have domains $\{1\}$, $\{2\}$ and $\{1, 2\}$.

For every smooth and distinctive operator f, Aceto, Bloom & Vaandrager declare four types of axioms. First of all, for every rule r as above there is an axiom $f(\sigma(x_1), \ldots, \sigma(x_n)) = a.\sigma(t)$, where $\sigma : \{x_1, \ldots, x_n\} \to \mathbb{T}(\Sigma)$ is the substitution given by $\sigma(x_i) = c_i.y_i$ for $i \in I$ and $\sigma(x_i) = x_i$ for $i \notin I$. Such an axiom is called an *action law*. Examples are CM2 and CM5 in Table 1.

Secondly, whenever I is the set of active arguments of f, but f has no rule of the form above (where the name of the variables y_i is of no importance), there is an axiom $f(\sigma(x_1), \ldots, \sigma(x_n)) = 0$, with σ as above (for an arbitrary choice of distinct variables y_i). Such an axiom is an *inaction law*. An example is CM6. If f has k active arguments, in total there are $|Act|^k$ action and inaction laws for f, one for every conceivable trigger with as domain the active arguments of f.

Finally, for any active argument i of f, there are laws

$$f(x_1, \ldots x_{i-1}, 0, x_{i+1}, \ldots, x_n) = 0 \qquad \text{and}$$
$$f(x_1, \ldots, x_i + x_i', \ldots, x_n) = f(x_1, \ldots, x_i, \ldots, x_n) + f(x_1, \ldots, x_i', \ldots, x_n).$$

Examples for the second type of inaction law are CM3 and CM7, and examples of *distributivity laws* are CM4, CM8 and CM9.

It is not hard to see that all axioms above respect $\leftrightarrows$ and that together they bring any term $f(p_1, \ldots, p_{ar(f)})$ with the p_i basic processes in head normal form.

The method of [1] makes three types of additions to a given finite GSOS language $\mathcal{L}$, and provides an equational head normalising axiomatisation on the resulting language, that respects strong bisimulation.

First of all, the operators 0, $a.\_$ and $+$ are added, if not already there. The corresponding axioms are A1–4 of Table 1. If all other operators are smooth and distinctive, for each of them the axioms just described are taken, which finishes the job. (In the presence of negative premises, this step is slightly more complex.)

In case there are operators f that are smooth but not distinctive, the set of operational rules of f is partitioned into subsets D such that no two rules in D have the same trigger, and the triggers of all rules in D have the same domain. Note that such a partition can always be found—possibly by taking exactly one rule in each subset D. Now for any subset D in the partition, an operator f_D with $ar(f_D) = ar(f)$ is added to the language, whose rules are exactly the rules in that subset, but with f_D in the source. By definition, f_D is distinctive. Now add an axiom $f(x_1, \ldots, x_{ar(f)}) = \sum f_D(x_1, \ldots, x_{ar(f)})$, where the sum is taken over all subsets in the partition, and apply the method above to the operators f_D. Again, it is trivial to check that the axioms respect $\leftrightarroweq$ and are head normalising. Applied to the $\|$ of CCS, this technique yields the left merge and communication merge as auxiliary operators, as well as a right merge, and the axiom CM1.

In case of operators f that are not smooth, a smooth operator $f^\star$ is added to $\mathcal{L}$, of which f is an abbreviation in the sense of Definition 9 (cf. Example 8). The treatment of $f^\star$ proceeds as above, and the project is finished by the axiom

$$f(p_1, ..., p_{ar(f)}) = f^\star(p_{f(1)}, ..., p_{f(ar(f^\star)))}).$$

Besides completeness for finite processes, using an infinitary induction principle the method of [1] even yields completeness for arbitrary processes. I will not treat this here, as it does not generalise to weak equivalences.

10 Turning Cool GSOS Rules into Equations

The method of [1] does not apply to $\leftrightarroweq_w$, $\leftrightarroweq_d$, $\leftrightarroweq_\eta$, and $\leftrightarroweq_b$, because these equivalences fail to be congruences for the $+$. However, Bloom [5] shows that the method applies more or less verbatim to $\leftrightarroweq_{rb}$. This section observes that the same holds for $\leftrightarroweq_{r\eta}$, and finds an adaptation to yield finite equational axiomatisations of $\leftrightarroweq_{rw}$ (resp. $\leftrightarroweq_{rd}$) that are sound and complete for finite processes on an augmentation of any RWB cool (resp. RDB cool) GSOS language.

On basic processes, the axioms A1–4 together with T1–T3 are complete for $\leftrightarroweq_{rw}$ [13], whereas complete axiomatisations for $\leftrightarroweq_{rd}$, $\leftrightarroweq_{r\eta}$ and $\leftrightarroweq_{rb}$ are obtained by dropping T3, T2 or both, respectively [12]. So in order to get axiomatisations of these equivalences that are complete for finite processes, all that is needed is head normalisation. The simplest approach is to use the same head normalising axioms as in the previous section, reasoning that axioms that respect $\leftrightarroweq$ surely respect a coarser equivalence like $\leftrightarroweq_{rb}$ or $\leftrightarroweq_{rw}$. The only way this approach could fail is when the auxiliary operators generated by [1] fail to be congruences for the equivalence relation at hand. The operators 0, $a.\_$ and $+$ are WB cool, and thus unproblematic. As observed in [5], for any RBB cool GSOS language, the augmented language is also RBB cool. Namely, the new operators do not show up in targets of new rules, so classifying all auxiliary operators as wild is sufficient. Since the auxiliary operators do not increase the collection of receiving arguments of operators either, it follows likewise that for any RHB cool GSOS language, the augmented language is also RHB cool. Hence one obtains

Theorem 7 The method of [1], together with axiom T1 (and T3), yields finite equational axiomatisations of $\leftrightarroweq_{rb}$ (resp. $\leftrightarroweq_{r\eta}$) that are sound and complete for finite processes on an augmentation of any RBB cool (resp. RHB cool) GSOS language. □

For $\leftrightarroweq_{rw}$ and $\leftrightarroweq_{rd}$ this approach fails. In particular, these equivalences fail to be congruences for the communication merge, as shown in Example 12.

Conjecture. *There exists no GSOS language including the parallel composition of CCS and* ≥ 2 *visible actions that admits a finite equational axiomatisation of weak bisimulation equivalence that is sound and complete for finite processes.*

Nevertheless, such an axiomatisation was found by Bergstra & Klop [4], using a variant of the communication merge that is not a GSOS operator; cf. Example 16. Their axiomatisation of $\|$ is obtained from the one in Table 1 by requiring $a, b \neq \tau$ in CM6, and adding the axioms $\tau.x|y = x|\tau.y = x|y$. Here I generalise their approach to arbitrary RWB cool (or RDB cool) GSOS languages.

In an RWB (or RDB) cool language, the smooth operators $f^\star$ that are needed to axiomatise a non-smooth operator f are unproblematic. For tame operators f, they are already in the language, and for a wild f it is not hard to define them in such a way that the augmented language remains RWB (or RDB) cool. Of the operators f_D needed to axiomatise a non-distinctive operator f, those that have exactly one active argument can be made to satisfy Clause 3 of Definition 15 by including the relevant τ-rule in D. This applies to the left merge, for example. All operators f_D with another number of active arguments cannot have τ-premises, by Definitions 15 and 10. These operators f_D are replaced by counterparts f'_D, obtained by replacing each premise $x \xrightarrow{c} y$ in a rule for f_D by $x \Longrightarrow\xrightarrow{c} y$. By Theorem 6, $\leftrightarrows_{rw}$ (or $\leftrightarrows_{rd}$) is a congruence for f'_D.

Lemma 5 For any processes $p_1, ..., p_n$ one has $f(p_1, \ldots, p_{ar(f)}) \leftrightarrows_{rd} \sum f'_D(p_1, \ldots, p_{ar(f)})$.

Proof: Suppose $\sum f'_D(p_1, \ldots, p_{ar(f)}) \xrightarrow{a} q$. Then there is a $\frac{H'}{f'_D(x_1,\ldots,x_n) \xrightarrow{a} t}$ and a closed substitution σ with $\sigma(x_i) = p_i$ for $i = 1, ..., n$ and $\sigma(t) = q$, such that $p_i \Longrightarrow\xrightarrow{c}_{\mathcal{L}} \sigma(y)$ for $(x_i \Longrightarrow\xrightarrow{c} y)$ in H'. This rule has been constructed from a rule $\frac{H}{f(x_1,\ldots,x_n) \xrightarrow{a} t}$, where in H the premises are of the form $x_i \xrightarrow{c} y$. By Lemma RDB, one obtains $f(p_1, \ldots, p_{ar(f)}) \Longrightarrow\xrightarrow{a} q$.

That $f(p_1, \ldots, p_{ar(f)}) \xrightarrow{a} q$ implies $\sum f'_D(p_1, \ldots, p_{ar(f)}) \xrightarrow{a} q$ follows straightforwardly. □

Now the required axiomatisation is obtained by omitting all inaction laws for the modified operators f'_D with $\sigma(x_i) = \tau.y_i$ for some active argument i, and instead adding *τ-laws*

$$f'_D(x_1, \ldots, \tau.x_i, \ldots, x_n) = f'_D(x_1, \ldots, x_i, \ldots, x_n)$$

for each active argument i of f'_D. One obtains

Theorem 8 The above adaptation of the method of [1], together with axioms T1, T2 (and T3), yields finite equational axiomatisations of $\leftrightarrows_{db}$ (resp. $\leftrightarrows_{rw}$) that are sound and complete for finite processes on an augmentation of any RDB cool (resp. RWB cool) GSOS language. □

11 Further work

The main contribution of Bloom's RBB cool format is the classification of operators in wild and tame ones. In Fokkink [8] this classification is refined into a classification on the *arguments* of operators, thereby allowing operators that have wild as well as tame arguments. This allows, for instance, capturing the Kleene star within the format, which is not possible in the approach presented here. Additionally, [8] transcends beyond the GSOS format, by allowing arbitrary terms in the left-hand side of premises. This allows capturing recursive specification, by the method of introducing constants rather than variable binding constructs. In [9] this work is generalised slightly further, using a new method to establish congruence formats, while in [10] this method is applied to generalise the simply RHB cool format presented here. Generalising the simply RWB and RDB cool formats along the same line requires further work.

12 A Challenge

All equivalences of Definition 5 are congruences of the GSOS language with rules

$$\frac{x_1 \xrightarrow{a} y}{f(x_1) \xrightarrow{a} g(y)} \qquad \frac{x_1 \xrightarrow{\tau} y}{g(x_1) \xrightarrow{\tau} g(y)} \qquad g(x_1) \xrightarrow{\tau} !x_1$$

$$\frac{x_1 \xrightarrow{a} y}{!x_1 \xrightarrow{a} y \| !x_1} \qquad \frac{x_1 \xrightarrow{a} y_1}{x_1 \| x_2 \xrightarrow{a} y_1 \| x_2} \qquad \frac{x_2 \xrightarrow{a} y_2}{x_1 \| x_2 \xrightarrow{a} x_1 \| y_2}$$

for $a \in Act$. Here, the operator $!x$ can be understood as a parallel composition of infinitely many copies of x. The rules for f, g and $\|$ are WB cool, but the one for ! is not. It is not even RBB safe in the sense of [8].

Open problem. Find a congruence format that includes the language above.

References

[1] L. Aceto, B. Bloom & F.W. Vaandrager (1994): *Turning SOS rules into equations. Information and Computation* 111(1), pp. 1–52.

[2] T. Basten (1996): *Branching bisimulation is an equivalence indeed! Information Processing Letters* 58(3), pp. 141–147.

[3] J.A. Bergstra & J.W. Klop (1986): *Algebra of communicating processes.* In J.W. de Bakker, M. Hazewinkel & J.K. Lenstra, editors: *Mathematics and Computer Science*, CWI Monograph 1, North-Holland, Amsterdam, pp. 89–138.

[4] J.A. Bergstra & J.W. Klop (1985): *Algebra of communicating processes with abstraction. Theoretical Computer Science* 37(1), pp. 77–121.

[5] B. Bloom (1995): *Structural operational semantics for weak bisimulations. Theoretical Computer Science* 146, pp. 25–68.

[6] B. Bloom, S. Istrail & A.R. Meyer (1995): *Bisimulation can't be traced. Journal of the ACM* 42(1), pp. 232–268.

[7] S.D. Brookes, C.A.R. Hoare & A.W. Roscoe (1984): *A theory of communicating sequential processes. Journal of the ACM* 31(3), pp. 560–599.

[8] W.J. Fokkink (2000): *Rooted branching bisimulation as a congruence. Journal of Computer and System Sciences* 60(1), pp. 13–37.

[9] W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2006): *Divide and Congruence: From Decomposition of Modalities to Preservation of Branching Bisimulation.* In F.S. de Boer, M.M. Bonsangue, S. Graf & W.-P. de Roever, editors: Revised Lectures Fourth International Symposium on *Formal Methods for Components and Objects*, FMCO 2005, Amsterdam, LNCS 4111, Springer, 2006, pp. 195-218.

[10] W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2005): *Divide and Congruence Applied to η-Bisimulation.* In P. Mosses & I. Ulidowski, editors: Proceedings of the Second Workshop on *Structural Operational Semantics*, SOS 2005, Lisbon, *Electronic Notes in Theoretical Computer Science* 156(1), pp. 97-113.

[11] R.J. van Glabbeek (2005): *A characterisation of weak bisimulation congruence.* In A. Middeldorp, V. van Oostrom, F. van Raamsdonk & R. de Vrijer, editors: *Processes, Terms and Cycles: Steps on the Road to Infinity: Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday*, LNCS 3838, Springer, pp. 26–39.

[12] R.J. van Glabbeek & W.P. Weijland (1996): *Branching time and abstraction in bisimulation semantics. Journal of the ACM* 43(3), pp. 555–600.

[13] R. Milner (1990): *Operational and algebraic semantics of concurrent processes.* In J. van Leeuwen, editor: *Handbook of Theoretical Computer Science*, chapter 19, Elsevier Science Publishers B.V. (North-Holland), pp. 1201–1242. Alternatively see *Communication and Concurrency*, Prentice-Hall, Englewood Cliffs, 1989, of which an earlier version appeared as *A Calculus of Communicating Systems*, LNCS 92, Springer-Verlag, 1980.

[14] F. Moller (1990): *The nonexistence of finite axiomatisations for CCS congruences.* In *Proceedings* 5^{th} *Annual Symposium on Logic in Computer Science,* Philadelphia, USA, IEEE Computer Society Press, pp. 142–153.

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

[16] G.D. Plotkin (2004): *A structural approach to operational semantics. The Journal of Logic and Algebraic Programming* 60–61, pp. 17–139. Originally appeared in 1981.

[17] R. de Simone (1985): *Higher-level synchronising devices in* Meije-*SCCS. Theoretical Computer Science* 37, pp. 245–267.

[18] I. Ulidowski (1992): *Equivalences on observable processes.* In *Proceedings* 7^{th} *Annual Symposium on Logic in Computer Science,* Santa Cruz, California, IEEE Computer Society Press, pp. 148–159.

[19] I. Ulidowski & I. Phillips (2002): *Ordered SOS rules and process languages for branching and eager bisimulations. Information and Computation* 178(1), pp. 180–213.

[20] I. Ulidowski & S. Yuen (2000): *Process languages for rooted eager bisimulation.* In C. Palamidessi, editor: Proceedings of the 11th International Conference on *Concurrency Theory*, CONCUR 2000, LNCS 1877, Springer, pp. 275–289.