Lean and Full Congruence Formats for Recursion

Rob van Glabbeek
Data61, CSIRO, Sydney, Australia
School of Computer Science & Engineering, University of New South Wales, Sydney, Australia

***Abstract*—In this paper I distinguish two (pre)congruence requirements for semantic equivalences and preorders on processes given as closed terms in a system description language with a recursion construct. A *lean congruence* preserves equivalence when replacing closed subexpressions of a process by equivalent alternatives. A *full congruence* moreover allows replacement within a recursive specification of subexpressions that may contain recursion variables bound outside of these subexpressions.**

I establish that bisimilarity is a lean (pre)congruence for recursion for all languages with a structural operational semantics in the ntyft/ntyxt format. Additionally, it is a full congruence for the tyft/tyxt format.

I. Introduction

Structural Operational Semantics [44], [46] is one of the main methods for defining the meaning of system description languages like CCS [44]. A system or *process* is represented by a closed term built from a collection of operators, process variables and usually a recursion construct, 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. The transitions between states are obtained from a set of proof rules called *transition rules*.

For purposes of representation and verification, several behavioural equivalence relations have been defined on processes, of which the most well-known is *(strong) bisimilarity* [44]. To allow compositional system verification, such equivalences 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.

Equally important is that the chosen equivalence relation $\sim$ is a congruence for recursion. Recursion allows the specification of a process as a canonical solution of an equation $X = E(X)$.[1] Here $E(X)$ is an expression that may contain the variable X. If W is the collection of other variables occurring in $E(X)$, not *bound* by the recursive specification, then the canonical solution of $X = E(X)$ is a W-ary function that returns a process for each valuation of these variables as processes. I call $\sim$ a *lean congruence* for recursion if each such operator satisfies the above-mentioned congruence requirement.

Take for example $E(X)$ to be $a.X + Y$ in the language CCS of Milner [44]. Then $W = \{Y\}$. Let $\sim$ be bisimilarity, so that $b.0 \sim b.0 + b.0$ [44]. Now the lean congruence requirement for $\sim$ insists that the selected solutions of the recursive equations $X = a.X+b.0$ and $X = a.X+(b.0+b.0)$, obtained from $X = a.X + Y$ by substituting each of these bisimilar processes for Y, are again bisimilar.

The lean congruence requirement plays a key rôle in the study of expressiveness of system description languages [33]. There, *correct translations* of one language into another up to a semantic equivalence $\sim$ are defined; and expressiveness hierarchies—one for each choice of $\sim$—are defined in terms of those translations. However, a correct translation can exist only when $\sim$ is a lean congruence for the source language, as well as for the source's image within the target language.

If $F(X)$ is an expression like $E(X)$, for simplicity assuming that neither contains variables other than X, and $E(p) \sim F(p)$ regardless which process p is substituted for the variable X, then the *full* congruence property demands that the selected solutions of the equations $X = E(X)$ and $X = F(X)$ are again equivalent. As a CCS example, suppose that a process is given as the solution of the equation $X = a.X + a.X$. Using the idempotence of $+$ under bisimilarity, one can now proceed to think of the same process, up to bisimilarity, as the solution of $X = a.X$. This type of reasoning is a central component in system verification by equivalence checking [7], [17], [6], [37], as applied in successful verification toolsets such as CADP [24] and mCRL2 [37]. Yet it is valid only if bisimilarity is a full congruence for recursion.

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 (*congruence 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. The first of these was proposed by Robert de Simone in [48], [49] and is now called the *De Simone format*. A generalisation featuring transition rules with negative premises is the *GSOS format* of Bloom, Istrail & Meyer [11], and a generalisation with *lookahead* is the *tyft/tyxt* format of Groote & Vaandrager [39]. The *ntyft/ntyxt* format of Groote [36] allows both negative premises and lookahead and generalises the GSOS as well as the tyft/tyxt format. All this work provides congruence formats for (strong) bisimilarity. Congruence formats for other *strong* semantic equivalences—treating the internal action τ like any other action—appear in [10], [21].[2] Formats for *weak* semantics—

[1] The particular solution supplied by structural operational semantics is the one whose transitions are determined by the transition rules.

[2] These congruence formats also apply to behavioural *preorders*, and then ensure that such a preorder is a *precongruence*.

abstracting from internal activity—can be found, e.g., in [50], [9], [18], [51], [52], [32], [23], [20].

Extensions to probabilistic systems appear for instance in [8], [41], [40], [25], [43], [5], [16]. Rule formats ensuring properties of operators other than being a (pre)congruence appear in [45] (commutativity), [15] (associativity), [2] (zero and unit elements), [3] (distributivity) and [1] (idempotence). Overviews on work on congruence formats and other rule formats, with many more references, can be found in [4], [38].

Yet, to the best of my knowledge, no one has proposed a congruence format for recursion. This hiatus is addressed here. I establish that bisimilarity is a lean congruence for recursion for all languages with a structural operational semantics in the ntyft/ntyxt format.[3] I did not succeed in showing that it is even a full congruence for all ntyft/ntyxt languages; nor did I find a counterexample. Even for GSOS languages this remains an open question. However, I show that bisimilarity is a full congruence for recursion for all tyft/tyxt languages.

My proof strategy follows the traditional method of [11], [39], [12]. However, for this to work smoothly, I present a new formulation—better fitted to my application—of the well-founded semantics of transition system specifications with negative premises, and show its consistency with previous formulations.

I could not establish the full congruence result directly, without using the lean congruence result as an intermediate step, even when restricting the latter to the tyft/tyxt format. Thus, I see no way around a sequence of two proofs with a large overlap.

The method of *modal decomposition* [22] yields alternative congruence proofs for operators specified in the tyft/tyxt and GSOS formats [22]. Extending this method to deal with recursion might be a way to extend my full congruence result to transition rules with negative premises.

Providing (lean and full) congruence formats for recursion for equivalences and preorders other than bisimilarity, as well as for weak versions of bisimilarity [44], [35]—supporting abstraction from internal actions—remains an important open problem.

II. Transition system specifications and their meaning

In this paper Var and A are two sets of *variables* and *actions*. Many concepts that will appear are parameterised by the choice of Var and A, but as in this paper this choice is fixed, a corresponding index is suppressed.

Definition 1 (*Signatures*) A *function declaration* is a pair (f, n) of a *function symbol* $f \notin Var$ and an *arity* $n \in \mathbb{N}$.[4] A function declaration $(c, 0)$ is also called a *constant declaration*. A *signature* is a set of function declarations. The set $\mathbb{T}(\Sigma)$ of *terms with recursion* over a signature Σ is defined inductively by:

- $Var \subseteq \mathbb{T}(\Sigma)$,
- if $(f,n) \in \Sigma$ and $t_1,...,t_n \in \mathbb{T}(\Sigma)$ then $f(t_1,...,t_n) \in \mathbb{T}(\Sigma)$,
- If $V_S \subseteq Var$, $S : V_S \rightarrow \mathbb{T}(\Sigma)$ and $X \in V_S$, then $\langle X|S\rangle \in \mathbb{T}(\Sigma)$.

A term $c()$ is abbreviated as c. A function S as appears in the last clause is called a *recursive specification*. A recursive specification S is often displayed as $\{X = S_X \mid X \in V_S\}$. An occurrence of a variable y in a term t is *free* if it does not occur in a subterm of t of the form $\langle X|S\rangle$ with $y \in V_S$. Let $var(t)$ denote the set of variables occurring free in a term $t \in \mathbb{T}(\Sigma)$, and let $\mathbb{T}(\Sigma, W)$ be the set of terms t over Σ with $var(t) \subseteq W$. $\mathrm{T}(\Sigma) := \mathbb{T}(\Sigma, \emptyset)$ is set of *closed* terms over Σ.

Example 1 Let Σ contain three unary functions $a.\_$, $b.\_$ and $d.\_$, and one infix-written binary function $\|$. Let $X, Y, z \in Var$. Then $S = \{\ X = (a.X)\|(b.Y), \quad Y = (d.Y)\|(X\|z)\ \}$ is a recursive specification, so $\langle X|S\rangle \in \mathbb{T}(\Sigma)$. Since $V_S = \{X, Y\}$, the only variable that occurs free in this term is z.

As illustrated here, I often choose upper case letters for bound variables (the ones occurring in a set V_S) and lower case ones for variables occurring free; this is a convention only.

A recursive specification S is meant to denote a V_S-tuple (in the example above a pair) of processes that—when filled in for the variables in V_S—forms a solution to the equations in S.[5] The term $\langle X|S\rangle$ denotes the X-component of such a tuple.

Definition 2 (*Substitution*) A Σ-*substitution* σ is a partial function from Var to $\mathbb{T}(\Sigma)$; it is *closed* if it is a total function from Var to $\mathrm{T}(\Sigma)$. If σ is a substitution and S any syntactic object, then $S[\sigma]$ denotes the object obtained from S by replacing, for x in the domain of σ, every free occurrence of x in S by $\sigma(x)$, while renaming bound variables if necessary to prevent name-clashes. In that case $S[\sigma]$ is called a *substitution instance* of S. A substitution instance $t[\sigma]$ where σ is given by $\sigma(x_i) = u_i$ for $i \in I$ is denoted as $t[u_i/x_i]_{i\in I}$, and for S a recursive specification $\langle t|S\rangle$ abbreviates $t[\langle Y|S\rangle/Y]_{Y\in V_S}$.

Example 2 Extend Σ from Ex. 1 with a constant c. Then $\langle X|S\rangle[b.c/z] = \langle X|\{X{=}(a.X)\|(b.Y),\ Y{=}(d.Y)\|(X\|b.c)\}\rangle$, $\langle X|S\rangle[X/z] = \langle Z|\{Z{=}(a.Z)\|(b.Y), \quad Y{=}(d.Y)\|(Z\|X)\}\rangle$ and $\langle X|S\rangle[b.c/Y] = \langle X|S\rangle$.

Structural operational semantics [46] defines the meaning of system description languages whose syntax is given by a signature Σ. It generates a transition system in which the states, or *processes*, are the closed terms over Σ—representing the remaining system behaviour from that state—and transitions between processes are supplied with labels. The transitions

[3] Some of those languages have a 3-valued transition system semantics, where bisimilarity becomes an asymmetric preorder. Here I establish that it is a precongruence.

[4] This work generalises seamlessly to operators with infinitely many arguments. Such operators occur, for instance, in [13, Appendix A.2]. Hence one may take n to be any ordinal. An operator, like the *summation* or *choice* of CCS [44], that actually takes any *set* of arguments, needs to be simulated by a family of operators with a *sequence* of arguments (but yielding the same value upon reshuffling of the arguments), one for each cardinality of this set.

[5] When S contains free variables from a set W, this solution is parameterised by the choice of a valuation of these variables as processes, thereby becoming a W-ary function.

between processes are obtained from a transition system specification, which consists of a set of transition rules.

Definition 3 (*Transition system specifications*) Let Σ be a signature. A *positive* Σ-*literal* is an expression $t \xrightarrow{a} t'$ and a *negative* Σ-*literal* an expression $t \stackrel{a}{\not\longrightarrow}$ with $t, t' \in \mathbb{T}(\Sigma)$ and $a \in A$. For $t, t' \in \mathbb{T}(\Sigma)$ the literals $t \xrightarrow{a} t'$ and $t \stackrel{a}{\not\longrightarrow}$ are said to *deny* each other. A *transition rule* over Σ is an expression of the form $\frac{H}{\alpha}$ with H a set of Σ-literals (the *premises* or *antecedents* of the rule) and α a positive Σ-literal (the *conclusion*). The terms at the left- and right-hand side of α are the *source* and *target* of the rule. A rule $\frac{H}{\alpha}$ with $H = \emptyset$ is also written α. A literal or transition rule is *closed* if it contains no free variables. A *transition system specification (TSS)* is a pair (Σ, R) with Σ a signature and R a set of transition rules over Σ; it is *positive* if all antecedents of its rules are positive.

The concept of a (positive) TSS presented above was introduced in GROOTE & VAANDRAGER [39]; the negative premises $t \stackrel{a}{\not\longrightarrow}$ were added in GROOTE [36]. The notion generalises the *GSOS rule systems* of [11] and constitutes the first formalisation of PLOTKIN's *Structural Operational Semantics (SOS)* [46] that is sufficiently general to cover many of its applications.

The following definition (from [27]) tells when a transition is provable from a TSS. It generalises the standard definition (see e.g. [39]) by (also) allowing the derivation of transition rules. The derivation of a transition $t \xrightarrow{a} t'$ corresponds to the derivation of the transition rule $\frac{H}{t \xrightarrow{a} t'}$ with $H = \emptyset$. The case $H \neq \emptyset$ corresponds to the derivation of $t \xrightarrow{a} t'$ under the assumptions H.

Definition 4 (*Proof*) Let $P = (\Sigma, R)$ be a TSS. A *proof* of a transition rule $\frac{H}{\alpha}$ from P is a well-founded, upwardly branching tree of which the nodes are labelled by Σ-literals, such that:

- the root is labelled by α, and
- if β is the label of a node q and K is the set of labels of the nodes directly above q, then
  - either $K = \emptyset$ and $\beta \in H$,
  - or $\frac{K}{\beta}$ is a substitution instance of a rule from R.

If a proof of $\frac{H}{\alpha}$ from P exists, then $\frac{H}{\alpha}$ is *provable* from P, notation $P \vdash \frac{H}{\alpha}$.

A TSS is meant to specify an LTS in which the transitions are closed positive literals. A positive TSS specifies a transition relation in a straightforward way as the set of all provable transitions.[6] But as pointed out in GROOTE [36], it is not so easy to associate a transition relation to a TSS with negative premises. In [31] several solutions to this problem were reviewed and evaluated. Arguably, the best method to assign a meaning to all TSSs is the *well-founded semantics* of VAN GELDER, ROSS & SCHLIPF [26], which in general yields a *3-valued transition relation* $T : \mathsf{T}(\Sigma) \times A \times \mathsf{T}(\Sigma) \rightarrow \{\textit{present}, \textit{undetermined}, \textit{absent}\}$. I present such a relation as a pair $\langle CT, PT \rangle$ of 2-valued transition relations—the sets of *certain* and *possible transitions*—with $CT \subseteq PT$. When insisting on 2-valued transition relations, the best method is the same, declaring meaningful only those TSSs whose well-founded semantics is 2-valued, meaning that $CT = PT$.

Below I give a new presentation of the well-founded semantics, strongly inspired by previous accounts in [47], [12], [31]. As Def. 4 does not allow the derivation of negative literals, to arrive at an approximation AT^+ of the set of transitions that are in the transition relation intended by a TSS P, one could start from an approximation AT^- of the closed negative literals that ought to be generated, and define AT^+ as the set of closed positive literals provable from P under the hypotheses AT^-. Intuitively,

1) if AT^- is an under- (resp. over-)approximation of the closed negative literals that "really" hold, then AT^+ will be an under- (resp. over-)approximation of the intended (2-valued) transition relation, and
2) if AT^+ is an under- (resp. over-)approximation of the intended transition relation, then the set of all closed negative literals that do not deny any literal in AT^+ is an over- (resp. under-)approximation of the closed negative literals that agree with the intended transition relation.

Definition 5 (*Over- and underappr. of transition relations*) Let P be a TSS. For ordinals λ the sets CT^+_λ and PT^+_λ of closed positive literals, and CT^-_λ, PT^-_λ of closed negative literals are defined inductively by:

PT^-_λ	is the set of literals that do not deny any $\beta \in CT^+_\kappa$ with $\kappa < \lambda$	$\beta \in PT^+_\lambda$ iff $P \vdash \frac{PT^-_\lambda}{\beta}$
CT^-_λ	is the set of literals that do not deny any $\beta \in PT^+_\lambda$	$\beta \in CT^+_\lambda$ iff $P \vdash \frac{CT^-_\lambda}{\beta}$.

Intuitively, CT^+_λ is an underapproximation of the set of transitions that should be in the transition relation specified by P, and PT^+_λ an overapproximation. Likewise, CT^-_λ is an underapproximation of the set of closed negative literals that should hold, and PT^-_λ an overapproximation. The approximations get better with increasing λ. To understand this inductively, note that PT^-_0 is the set of *all* closed negative literals, and thus surely an overapproximation. The induction step is given by considerations 1 and 2 above.

Lemma 1 $CT^-_\kappa \subseteq CT^-_\lambda \subseteq PT^-_\lambda \subseteq PT^-_\kappa$ and $CT^+_\kappa \subseteq CT^+_\lambda \subseteq PT^+_\lambda \subseteq PT^+_\kappa$ for $\kappa < \lambda$.

Proof: Let $\kappa < \lambda$. The definition of PT^-_λ immediately yields $PT^-_\lambda \subseteq PT^-_\kappa$. From this, applying Def. 5, one obtains

[6] Readers interested only in the restriction of my results to TSSs without negative premises—giving rise to 2-valued transition relations—can safely skip the remainder of this section, and identify $p \stackrel{a}{- - \rightarrow} p'$ with $p \xrightarrow{a} p'$. In the proofs of Prop. 3 and Thm. 2 also $p \stackrel{a}{- - \rightarrow}_\lambda p'$ and $p \xrightarrow{a}_\lambda p'$ equal $p \xrightarrow{a} p'$, for any λ; so the induction on λ can be skipped, as well as the auxiliary Claims 3 and 1, and the proof proceeds directly by induction on π.

$PT_\lambda^+ \subseteq PT_\kappa^+$, $CT_\kappa^- \subseteq CT_\lambda^-$ and $CT_\kappa^+ \subseteq CT_\lambda^+$, respectively. The remaining claims follow by induction on λ.

As PT_0^- is the universal relation, certainly $CT_0^- \subseteq PT_0^-$, so $CT_0^+ \subseteq PT_0^+$.

Let λ be a limit ordinal. Then $PT_\lambda^- = \bigcap_{\mu<\lambda} PT_\mu^-$. For any $\kappa, \mu < \lambda$ one has $CT_\kappa^- \subseteq PT_\mu^-$ by induction. Namely $CT_\kappa^- \subseteq CT_\mu^- \subseteq PT_\mu^-$ if $\kappa \leq \mu < \lambda$, and $CT_\kappa^- \subseteq PT_\kappa^- \subseteq PT_\mu^-$ if $\mu \leq \kappa < \lambda$. Hence $CT_\kappa^- \subseteq \bigcap_{\mu<\lambda} PT_\mu^- = PT_\lambda^-$ for any $\kappa < \lambda$, and hence $CT_\kappa^+ \subseteq PT_\lambda^+$. With Def. 5 this implies $CT_\lambda^- \subseteq PT_\lambda^-$ and hence $CT_\lambda^+ \subseteq PT_\lambda^+$.

Now let $\lambda = \mu{+}1$. By induction $CT_\mu^+ \subseteq PT_\mu^+$. With Def. 5 this implies $CT_\mu^- \subseteq PT_\lambda^-$, and hence $CT_\mu^+ \subseteq PT_\lambda^+$. With Def. 5 this implies $CT_\lambda^- \subseteq PT_\lambda^-$ and hence $CT_\lambda^+ \subseteq PT_\lambda^+$. $\square$

Since the closed literals over Σ form a proper set, there must be an ordinal κ such that $PT_\lambda^- = PT_\kappa^-$ for all $\lambda > \kappa$, and hence also $PT_\lambda^+ = PT_\kappa^+$, $CT_\lambda^- = CT_\kappa^-$ and $CT_\lambda^+ = CT_\kappa^+$.

Definition 6 Such an ordinal κ is called *closure ordinal*. Let $PT^- := PT_\kappa^-$, $PT^+ := PT_\kappa^+$, $CT^- := CT_\kappa^-$ and $CT^+ := CT_\kappa^+$.

Remark 1 $PT^- = \bigcap_\lambda PT_\lambda^-$, taking the intersection over all ordinals. Likewise, $PT^+ = \bigcap_\lambda PT_\lambda^+$, $CT^- = \bigcup_\lambda CT_\lambda^-$ and $CT^+ = \bigcup_\lambda CT_\lambda^+$.

Remark 2 PT^- is the set of literals that do not deny any literal in CT^+, and likewise for CT^- and PT^+. Moreover, $CT^- \subseteq PT^-$ and $CT^+ \subseteq PT^+$.

Definition 7 (*Well-founded semantics*) The 3-valued transition relation $\langle CT^+, PT^+\rangle$ constitutes the well-founded semantics of P.

Below I show that the above account of the well-founded semantics is consistent with the one in [31], and thereby with the ones in [12], [47], [26].

Definition 8 (*Well-supported proof* [31]) Let $P = (\Sigma, R)$ be a TSS. A *well-supported proof* from P of a closed literal α is a well-founded tree with the nodes labelled by closed literals, such that the root is labelled by α, and if β is the label of a node and K is the set of labels of the children of this node, then:

- either β is positive and $\frac{K}{\beta}$ is a substitution instance of a rule in R;
- or β is negative and for each set N of closed negative literals with $P \vdash \frac{N}{\gamma}$ for γ a closed positive literal denying β, a literal in K denies one in N.

$P \vdash_{ws} \alpha$ denotes that a well-supported proof from P of α exists.

Proposition 1 Let P be a TSS. Then $P \vdash_{ws} p \xrightarrow{a} q$ iff $(p \xrightarrow{a} q) \in CT^+$, and $P \vdash_{ws} p \stackrel{a}{\nrightarrow}$ iff $(p \stackrel{a}{\nrightarrow}) \in CT^-$.

Proof: $\boxed{\Rightarrow}$: Let π be a well-supported proof of a closed literal α. By consistently applying the same closed substitution to all literals occurring in π, one can assume, without loss of generality, that all literals in π are closed. With structural induction on π I show that $\alpha \in CT^+ \cup CT^-$.

Suppose α is positive and $\frac{K}{\alpha}$ is the closed substitution instance of the rule of P applied at the root of π. Then for each $\beta \in K$ the literal β is ws-provable from P by means of a strict subproof of π. By induction $\beta \in CT^+ \cup CT^-$. As CT^+ is CT_κ^+ for some ordinal κ, it is closed under deduction. Hence $\alpha \in CT^+$.

Suppose α is negative. Let β be closed positive literal denying α. By Def. 8, each set N of closed negative literals with $P \vdash \beta$ contains a literal γ_N denying a literal δ_N that is ws-provable from P by means of a strict subproof of π. By induction $\delta_N \in CT^+$. Hence $\gamma_N \notin PT^-$. Consequently $\beta \notin PT^+$. Hence $\alpha \in CT^-$.

$\boxed{\Leftarrow}$: Suppose $\alpha \in CT_\lambda^+ \cup CT_\lambda^-$. With induction on λ I show that $P \vdash_{ws} \alpha$. First suppose $\alpha \in CT_\lambda^-$. Let N be a set of closed negative literals with $P \vdash \frac{N}{\gamma}$ for γ a closed positive literal denying α. Assume that $N \subseteq PT_\lambda^-$. Then γ would be in PT_λ^+, contradicting the definition of CT_λ^-. So N contains a literal that is not in PT_λ^-, i.e., denies a literal δ_N in CT_κ^+ for some $\kappa < \lambda$. By induction, $P \vdash_{ws} \delta_N$. It follows that $P \vdash_{ws} \alpha$.

Now suppose $\alpha \in CT_\lambda^+$. Then $P \vdash \frac{CT_\lambda^-}{\alpha}$. By the case above $P \vdash_{ws} \beta$ for each $\beta \in CT_\lambda^-$. Hence $P \vdash_{ws} \alpha$. $\square$

The above result, together with Theorem 1 in [31], and the observation in [31] that literals $t \stackrel{a}{\nrightarrow} t'$ can be eliminated from consideration (as done here), implies that the well-founded semantics given above agrees with the one from [31].

In [31] it was shown that $\vdash_{ws}$ is consistent, in the sense that no TSS admits well-supported proofs of two literals that deny each other. This also follows directly from the material above. A TSS P is called *complete* [31] if for each p and a, either $P \vdash_{ws} p \stackrel{a}{\nrightarrow}$ or $P \vdash_{ws} p \xrightarrow{a} q$ for some q. This implies that CT^- is exactly the set of closed negative literals that do not deny any literal in CT^+. Hence $CT^- = PT^-$ and thus $CT^+ = PT^+$. So the 3-valued transition system associated to a complete TSS is 2-valued.

Below I write $P \vdash p \xrightarrow{a}_\lambda q$ for $(p \xrightarrow{a} q) \in CT_\lambda^+$, $P \vdash p \stackrel{a}{\nrightarrow}_\lambda$ for $(p \stackrel{a}{\nrightarrow}) \in CT_\lambda^-$, $P \vdash p \stackrel{a}{\dashrightarrow}_\lambda q$ for $(p \xrightarrow{a} q) \in PT_\lambda^+$ and $P \vdash p \stackrel{a}{\not\dashrightarrow}_\lambda$ for $(p \stackrel{a}{\nrightarrow}) \in PT_\lambda^-$. Moreover, $p \xrightarrow{a} q$, resp. $p \stackrel{a}{\dashrightarrow} q$, will abbreviate $p \xrightarrow{a}_\kappa q$, resp. $p \stackrel{a}{\dashrightarrow}_\kappa q$, where κ is the closure ordinal of Def. 6.

In my forthcoming lean congruence proof I will apply structural induction on "the proof of a transition $p \xrightarrow{a}_\lambda q$ or $p \stackrel{a}{\dashrightarrow}_\lambda q$ from P". There I will mean the proofs of $\frac{CT_\lambda^-}{p \xrightarrow{a} q}$ and $\frac{PT_\lambda^-}{p \xrightarrow{a} q}$, respectively, as this is what constitutes the evidence for the statement $P \vdash p \xrightarrow{a}_\lambda q$, resp. $P \vdash p \stackrel{a}{\dashrightarrow}_\lambda q$.

III. The bisimulation preorder

The goal of this paper is to show that bisimilarity is a congruence for recursion for all languages with a structural operational semantics in the ntyft/ntyxt format. Traditionally [44], bisimilarity is defined on 2-valued transition systems only, whereas the structural operational semantics of a language specified by a TSS can be 3-valued. Rather than limit my results to languages specified by complete TSSs, I use an extension of the notion of bisimilarity to 3-valued transition systems. Such an extension, called *modal refinement*, is provided in [42]. There, 3-valued transition systems are called *modal transition systems*.

Definition 9 (*Bisimilarity*) Let P be a TSS. A *bisimulation* $\mathcal{R}$ is a binary relation on the states of $\mathrm{T}(\Sigma)$ such that, for $p, q \in \mathrm{T}(\Sigma)$ and $a \in A$,

- if $p \mathrel{\mathcal{R}} q$ and $P \vdash p \xrightarrow{a} p'$, then there is a q' with $P \vdash q \xrightarrow{a} q'$ and $p' \mathrel{\mathcal{R}} q'$,
- if $p \mathrel{\mathcal{R}} q$ and $P \vdash q \dashrightarrow{a} q'$, then there is a p' with $P \vdash p \dashrightarrow{a} p'$ and $p' \mathrel{\mathcal{R}} q'$.

A process $q{\in}\mathrm{T}(\Sigma)$ is a *modal refinement* of $p{\in}\mathrm{T}(\Sigma)$, notation $p \sqsubseteq_B q$, if there exists a bisimulation $\mathcal{R}$ with $p \mathrel{\mathcal{R}} q$. I call $\sqsubseteq_B$ the *bisimulation preorder*, or *bisimilarity*. The kernel of $\sqsubseteq$, given by $\equiv_B := \sqsubseteq_B \cap \sqsupseteq_B$, is *bisimulation equivalence*.

Clearly, modal refinement is reflexive and transitive, and hence a preorder. The underlying idea is that a process p with a 3-valued transition relation $\langle CT, PT \rangle$ is a *specification* of a process with a 2-valued transition relation, in which the presence or absence of certain transitions is left open. CT contains the transitions that are *required* by the specification, and PT the ones that are *allowed*. If $p \sqsubseteq_B q$, then q may be closer to the eventual implementation, in the sense that some of the undetermined transitions have been resolved to present or absent. The requirements of Def. 9 now say that any transition that is required by p should be (matched by a transition) required by q, whereas any transition allowed by q, should certainly be (matched by a transition) allowed by p.

In case p and q are 2-valued (i.e. *implementations*) the modal refinement relation is just the traditional notion of bisimilarity [44] (and thus symmetric).

While achieving a higher degree of generality of my lean congruence theorem by interpreting incomplete TSSs as modal transition systems, I do not propose incomplete TSSs as a tool for the specification of modal transition systems.

IV. Congruence properties

In the presence of recursion, two sensible notions of precongruence come to mind. Let $\sqsubseteq$ be a preorder on the set $\mathrm{T}(\Sigma)$ of closed terms over Σ. For $\rho, \nu : Var \rightarrow \mathrm{T}(\Sigma)$ closed substitutions write $\rho \sqsubseteq \nu$ iff $\rho(x) \sqsubseteq \nu(x)$ for each $x \in Var$.

Definition 10 (*Lean precongruence*) A preorder $\sqsubseteq\, \subseteq \mathrm{T}(\Sigma) \times \mathrm{T}(\Sigma)$ is a *lean precongruence* iff $t[\rho] \sqsubseteq t[\nu]$ for any term $t \in \mathbb{T}(\Sigma)$ and any closed substitutions ρ and ν with $\rho \sqsubseteq \nu$.

Definition 11 (*Full precongruence*) A preorder $\sqsubseteq \subseteq \mathrm{T}(\Sigma) \times \mathrm{T}(\Sigma)$ is a *full precongruence* iff it satisfies

$$p_i \sqsubseteq q_i \text{ for all } i = 1, ..., n \quad \Rightarrow \quad f(p_1, ..., p_n) \sqsubseteq f(q_1, ..., q_n) \tag{1}$$

$$S_Y[\sigma] \sqsubseteq S'_Y[\sigma] \text{ for all } Y \in W \text{ and } \sigma : W \rightarrow \mathrm{T}(\Sigma) \quad \Rightarrow \quad \langle X|S \rangle \sqsubseteq \langle X|S' \rangle \tag{2}$$

for all functions $(f, n) \in \Sigma$, closed terms $p_i, q_i \in \mathrm{T}(\Sigma)$, and recursive specifications $S, S' : W \rightarrow \mathbb{T}(\Sigma, W)$ with $X \in W \subseteq Var$.

A lean (resp. full) precongruence that is symmetric (i.e. an equivalence relation) is called a *lean* (resp. *full*) *congruence*. Clearly, each full (pre)congruence is also a lean (pre)congruence, and each lean (pre)congruence satisfies (1) above. Both implications are strict, as the following examples illustrate.

Example 3 Consider the TSS given by the rules

$$a.x \xrightarrow{a} x \qquad \frac{x \xrightarrow{a} x'}{x \| y \xrightarrow{a} x' \| y} \qquad \frac{y \xrightarrow{a} y'}{x \| y \xrightarrow{a} x \| y'}$$

where a ranges over A, and the recursion rule from Def. 13 below. An *infinite trace* of a process p is a sequence $a_1 a_2 \cdots \in A^\omega$ such that there are processes $p_1, p_2, \ldots$ with $p \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \xrightarrow{a_3} \ldots$. Let $p \sqsubseteq q$ iff for each infinite trace σ of p there is an infinite trace of q that has a suffix in common with σ. This is a preorder indeed. It is not hard to check that $\sqsubseteq$ is a precongruence for both action prefixing $a.\_$ and parallel composition $\_\|\_$, in the sense that (1) holds. However, it fails to be a lean congruence, because $a.\langle X|X{=}c.X\rangle \equiv b.\langle X|X{=}c.X\rangle$, yet when filled in for Y in $\langle Z|Z{=}Y\|Z\rangle$ (which can be seen as $!Y$, an infinite parallel composition of copies of Y) the two are no longer equivalent.

I did not find a pair of a TSS and a preorder known from the literature showing the same. This suggests that most common preorders that are (pre)congruences for a selection of common operators are also lean (pre)congruences for recursion.

Example 4 Consider the TSS with a constant $\mathbf{0}$ and action prefixing, and only the rules for recursion from Def. 13 and $a.x \xrightarrow{a} x$ for $a \in A$, with $\tau \in A$ the *internal action*. Consider any semantic equivalence $\sim$ satisfying $x \sim \tau.x$, and such that *divergence* $\langle X|X{=}\tau.X\rangle$ differs from *deadlock* or *inaction* $\mathbf{0}$. Such semantic equivalences are abound in the literature and include the *failures* semantics of CSP [14], [28] and *branching bisimilarity with explicit divergence* [34], [28]. They are all lean congruences (at least when no other operators are present). Yet, since $0 \sim \langle X|X{=}X\rangle \not\sim \langle X|X{=}\tau.X\rangle$, they fail to be full congruences.

A lean congruence is required for treating processes as equivalence classes of closed terms rather than as the closed terms themselves, in such a way that each term $t{\in}\mathbb{T}(\Sigma, W)$ with free variables drawn from the set W models a W-ary operator on such processes. As explained in the introduction, this notion of congruence facilitates a formal comparison of the expressive power of system description languages [33]. However, it does not allow equivalence preserving modifications of recursive specifications themselves, as contemplated in the introduction. That requires a full congruence.

V. THE PURE NTYXT/NTYFT FORMAT WITH RECURSION

Definition 12 (*ntytt, ntyft, ntyxt, nxytt rules*) An *ntytt rule* is a rule in which the right-hand sides of positive premises are variables that are all distinct, and that do not occur in the source. An ntytt rule is an *ntyxt rule* if its source is a variable, an *ntyft rule* if its source contains exactly one function symbol and no multiple occurrences of variables, and an *nxytt rule* if the left-hand sides of its premises are variables.

The idea behind the names of the rules is that the 'n' in front refers to the presence of negative premises, and the following four letters refer to the allowed forms of left- and right-hand sides of premises and of the conclusion, respectively. For example, ntyft means a rule with negative premises (n), where left-hand sides of premises are general terms (t), right-hand sides of positive premises are variables (y), the source contains exactly one function symbol (f), and the target is a general term (t).

Definition 13 A TSS is in the *ntyft/ntyxt format with recursion* if for every recursive specification S and $X \in V_S$ it has a rule

$$\frac{\langle S_X|S\rangle \xrightarrow{a} z}{\langle X|S\rangle \xrightarrow{a} z}$$

and all of its other rules are ntyft or ntyxt rules.

Definition 14 (*Well-founded and pure rules; distance*) The *dependency graph* of an ntytt rule with $\{t_i \xrightarrow{a_i} y_i \mid i \in I\}$ as set of positive premises is the directed graph with edges $\{\langle x, y_i\rangle \mid x \in var(t_i)$ for some $i \in I\}$. A ntytt rule is *well-founded* if each backward chain of edges in its dependency graph is finite. A variable in a rule is *free* if it occurs neither in the source nor in the right-hand sides of the premises of this rule. A rule is *pure* if it is well-founded and does not contain free variables. A TSS is *well-founded*, resp. *pure*, if all of its rules are.

Let $r = \frac{H}{t \xrightarrow{a} u}$ be a pure ntytt rule. The *distance* of a variable $y \in var(r)$ to the source of r is the ordinal number given by

$$\begin{array}{ll} \mathit{dist}(x) = 0 & \text{if } x \in var(t), \\ \mathit{dist}(y) = 1 + \sup(\{\mathit{dist}(x) \mid x \in var(t)\}) & \text{if } (t \xrightarrow{a} y) \in H. \end{array}$$

BOL & GROOTE show that bisimilarity is a congruence for any language specified by a complete TSS in the well-founded ntyft/ntyxt format (without recursion) [12]. This generalises a result by GROOTE [36], showing the same for stratified TSSs in the well-founded ntyft/ntyxt format; here *stratified* is a more restrictive criterion than completeness, guaranteeing that a TSS has a well-defined meaning as a 2-valued transition relation. That result, in turn, generalises the congruence formats of GROOTE & VAANDRAGER [39] for the well-founded tyft/tyxt format (obtained by leaving out negative premises) and for the GSOS format of BLOOM, ISTRAIL & MEYER [11]. Both of these generalise the De Simone format [48], [49].

FOKKINK AND VAN GLABBEEK show that for any complete TSS in tyft/tyxt (resp. ntyft/ntyxt) format there exists a pure (and thus well-founded) complete TSS in tyft (resp. ntyft) format that generates the same transition relation [19]. From this it follows that the restriction to well-founded TSSs can be dropped from the congruence formats of [12] and [39]. The result of [19] generalises straightforwardly to incomplete TSSs, and to formats with recursion.

Theorem 1 For each TSS in the tyft/tyxt (resp. ntyft/ntyxt) format with recursion there exists a pure TSS in the tyft (resp. ntyft) format with recursion, generating the same (3-valued) transition relation.

Proof: [19, Theorem 5.4] shows that for each TSS P in ntyft/ntyxt format there exists a TSS P' in pure ntyft format, such that for any closed transition rule $\frac{N}{\alpha}$ with only negative premises, one has $P \vdash \frac{N}{\alpha} \Leftrightarrow P' \vdash \frac{N}{\alpha}$. This result generalises seamlessly to TSS in the ntyft/ntyxt format with recursion; I leave it to the reader to check that recursion causes no new complications in the proof.

[19] obtains the quoted result for complete TSSs from Thm. 5.4 by means of an application of [19, Prop. 5.2], which says that if P and P' are TSSs such that $P \vdash \frac{N}{\alpha} \Leftrightarrow P' \vdash \frac{N}{\alpha}$ for any closed transition rule $\frac{N}{\alpha}$ with only negative premises, then P is complete iff P' is, and in that case they determine the same transition relation. This Prop. 5.2 was taken verbatim from [30, Prop. 29].

In [31], the journal version of [30], Prop. 29 was extended to also conclude, under the same assumption, that P and P' determine the same 3-valued transition relation according to the well-founded semantics. Using this version of Prop. 29 instead of Prop. 5.2 yields the required result. □

The next two propositions (not used in the rest of the paper) tell that any language specified by TSS in the ntyft/ntyxt format with recursion satisfies two sanity requirements from [29]. The first is that, up to $\equiv_B$, the meaning of a closed term $\langle X|S\rangle$ is the X-component of a solution of S:

Proposition 2 Let $P = (\Sigma, R)$ be a TSS in the ntyft/ntyxt format with recursion and S a recursive specification with $X \in V_S$. Then $\langle X|S\rangle \equiv_B \langle S_X|S\rangle$.

Proof: $P \vdash \langle X|S\rangle \xrightarrow{a} q$ for some $a \in A$ and $q \in \mathrm{T}(\Sigma)$ iff $P \vdash \langle S_X|S\rangle \xrightarrow{a} q$. □

For the second, *invariance under α-conversion*, write $t \stackrel{\alpha}{=} u$ if the terms $t, u \in \mathbb{T}(\Sigma)$ differ only in the names of their bound variables (the variables from V_S within a subexpression of the form $\langle X|S\rangle$).

Proposition 3 Let $P = (\Sigma, R)$ be a TSS in the ntyft/ntyxt format with recursion. Then $p \stackrel{\alpha}{=} q \Rightarrow p \equiv_B q$ for all $p, q \in \mathrm{T}(\Sigma)$.

Proof: By Thm. 1 I may assume, without loss of generality, that P is in the pure ntyft format with recursion. I show that $\stackrel{\alpha}{=}$ is a bisimulation on $\mathrm{T}(\Sigma)$—since $\stackrel{\alpha}{=}$ is also symmetric, this yields the required result.

Thus I need to show that, for $p, q \in \mathrm{T}(\Sigma)$ and $a \in A$,

- if $p \stackrel{\alpha}{=} q$ and $P \vdash p \xrightarrow{a} p'$, then there is a q' with $P \vdash q \xrightarrow{a} q'$ and $p' \stackrel{\alpha}{=} q'$,
- if $p \stackrel{\alpha}{=} q$ and $P \vdash q \stackrel{a}{\dashrightarrow} q'$, then there is a p' with $P \vdash p \stackrel{a}{\dashrightarrow} p'$ and $p' \stackrel{\alpha}{=} q'$.

To this end it suffices to establish, for all ordinals λ, that

4. if $p \stackrel{\alpha}{=} q$ and $P \vdash p \xrightarrow{a}_\lambda p'$, then there is a q' with $P \vdash q \xrightarrow{a} q'$ and $p' \stackrel{\alpha}{=} q'$,
2. if $p \stackrel{\alpha}{=} q$ and $P \vdash q \stackrel{a}{\dashrightarrow} q'$, then there is a p' with $P \vdash p \stackrel{a}{\dashrightarrow}_\lambda p'$ and $p' \stackrel{\alpha}{=} q'$.

The desired result is then obtained by taking λ to be the closure ordinal κ of Def. 6. This I will do by induction on λ, at the same time establishing that

3. if $p \stackrel{\alpha}{=} q$ and $P \vdash p \stackrel{a}{\not\longrightarrow}_\lambda$, then $P \vdash q \stackrel{a}{\not\longrightarrow}$,
1. if $p \stackrel{\alpha}{=} q$ and $P \vdash q \stackrel{a}{\not\dashrightarrow}$, then $P \vdash p \stackrel{a}{\not\dashrightarrow}_\lambda$.

So assume Claims 1–4 have been established for all $\kappa < \lambda$.

Suppose $p \stackrel{\alpha}{=} q$ and $P \vdash q \stackrel{a}{\not\dashrightarrow}$. By Remark 2 there is no $q' \in \mathrm{T}(\Sigma)$ with $P \vdash q \xrightarrow{a} q'$. So by induction, using Claim 4 above, there is no $p' \in \mathrm{T}(\Sigma)$ with $P \vdash p \xrightarrow{a}_\kappa p'$ for some $\kappa < \lambda$. By Def. 5 $P \vdash p \stackrel{a}{\not\dashrightarrow}_\lambda$. This yields Claim 1.

Now suppose $p \stackrel{\alpha}{=} q$ and $P \vdash q \stackrel{a}{\dashrightarrow} q'$. I need to find a p' with $P \vdash q \stackrel{a}{\dashrightarrow}_\lambda q'$ and $p' \stackrel{\alpha}{=} q'$. This I will do by structural induction on the proof π of $p \stackrel{a}{\dashrightarrow} p'$ from P, making a case distinction based on the shape of p.

- Let $p = f(p_1, \ldots, p_n)$. Then $q = f(q_1, \ldots, q_n)$ where $p_i \stackrel{\alpha}{=} q_i$ for $i{=}1, \ldots, n$. Let π be a proof of $P \vdash q \stackrel{a}{\dashrightarrow} q'$ from P. By Defs. 4 and 13, there must be a pure ntyft rule $r = \frac{H}{f(x_1,\ldots,x_n) \xrightarrow{a} t}$ in R and a closed substitution ν with $\nu(x_i) = q_i$ for $i = 1, ..., n$ and $t[\nu] = q'$, such that for each $(t_y \xrightarrow{c} y) \in H$ the transition $P \vdash t_y[\nu] \stackrel{c}{\dashrightarrow} \nu(y)$ is provable from P by means of a strict subproof of π, and $P \vdash u[\nu] \stackrel{c}{\not\dashrightarrow}$ for each $(u \stackrel{c}{\not\longrightarrow}) \in H$. Next, I define a substitution $\sigma : \mathit{var}(r) \rightarrow \mathrm{T}(\Sigma)$ such that
  (i) $\sigma(x_i) = p_i$ for $i = 1, \ldots, n$,
  (ii) $\sigma(y) \stackrel{\alpha}{=} \nu(y)$ for each $y \in \mathit{var}(r)$,
  (iii) $P \vdash t_y[\sigma] \stackrel{c}{\dashrightarrow}_\lambda \sigma(y)$ for each $(t_y \xrightarrow{c} y) \in H$.

  The definition of $\sigma(y)$ and the inference of (i)–(iii) above proceed with induction on the distance of $y \in \mathit{var}(r)$ from the source of r,

  Base case: Let $\sigma(x_i) := p_i$ for $i = 1, \ldots, n$, so that Property (i) is satisfied. Regarding Property (ii), $\sigma(x_i) \stackrel{\alpha}{=} \nu(x_i)$ for $i = 1, \ldots, n$.

  Induction step: When defining $\sigma(y)$ for some $y \in \mathit{Var}$ with $(t_y \xrightarrow{c} y) \in H$, by induction $\sigma(x)$ has been defined already for all $x \in \mathit{var}(t_y)$, so I may assume that $\sigma(x) \stackrel{\alpha}{=} \nu(x)$ for all $x \in \mathit{var}(t_y)$ and hence $t_y[\sigma] \stackrel{\alpha}{=} t_y[\nu]$. By induction on π, there is a p_y with $P \vdash t_y[\sigma] \stackrel{c}{\dashrightarrow}_\lambda p_y$ and $p_y \stackrel{\alpha}{=} \nu(y)$. Define $\sigma(y) := p_y$. Properties (ii) and (iii) now hold for y.

  Take $p' := t[\sigma]$. So $p' = t[\sigma] \stackrel{\alpha}{=} t[\nu] = q'$ by Property (ii) of σ. For each premise $(u \stackrel{c}{\not\longrightarrow}) \in H$ one has $u[\sigma] \stackrel{\alpha}{=} u[\nu]$ by Property (ii) of σ. So $P \vdash u[\sigma] \stackrel{c}{\not\dashrightarrow}_\lambda$ by Claim 1. By Defs. 4 and 13, together with Property (iii) of σ, this implies $P \vdash p = f(p_1, \ldots, p_n) \stackrel{a}{\dashrightarrow}_\lambda t[\sigma] = p'$.
- Let $p = \langle X|S\rangle$. Then $q = \langle \alpha(X)|S'[\alpha]\rangle$ for some recursive specification $S' : V_S \rightarrow \mathbb{T}(\Sigma)$ with $S_Y \stackrel{\alpha}{=} S'_Y$ for all $Y \in V_S$, and an injective substitution $\alpha : V_S \rightarrow \mathit{Var}$ such that the range of α contains no variables occurring free in $\langle S'_Y|S\rangle$ for some $Y \in V_S$. Now $\langle S_X|S\rangle \stackrel{\alpha}{=} \langle S_X|S'\rangle \stackrel{\alpha}{=} \langle S'_{\alpha(X)}|S'[\alpha]\rangle$. Let π be a proof of $P \vdash q \stackrel{a}{\dashrightarrow} q'$ from P. By Defs. 4 and 13 $P \vdash \langle S'_{\alpha(X)}|S'[\alpha]\rangle \stackrel{a}{\dashrightarrow} q'$ is provable from P by means of a strict subproof of π. So by induction there is a p' such that $P \vdash \langle S_X|S\rangle \stackrel{a}{\dashrightarrow}_\lambda p'$ and $p' \stackrel{\alpha}{=} q'$. By Defs. 4 and 13, $P \vdash p = \langle X|S\rangle \stackrel{a}{\dashrightarrow}_\lambda p'$.

This establishes Claim 2.

Next, suppose that $p \stackrel{\alpha}{=} q$ and $P \vdash p \stackrel{a}{\not\longrightarrow}_\lambda$. By Def. 5 there is no $p' \in \mathrm{T}(\Sigma)$ with $P \vdash p \stackrel{a}{\dashrightarrow}_\lambda p'$. Using Claim 2, there is no $q' \in \mathrm{T}(\Sigma)$ with $P \vdash q \stackrel{a}{\dashrightarrow} q'$. By Remark 2, $P \vdash q \stackrel{a}{\not\longrightarrow}$. This yields Claim 3.

Claim 4 follows by structural induction on the proof of $p \xrightarrow{a}_\lambda p'$ from P, pretty much in the same way as Claim 2 above. $\square$

Prop. 3 could be classified as "self-evident". One reason to spell out the proof above is to obtain a template for bisimilarity proofs in the setting of the well-founded semantics. I will use this template in the forthcoming lean congruence proof.

VI. A LEAN CONGRUENCE RESULT

The following congruence proof is strongly inspired by the one in [12].

Theorem 2 Bisimilarity is a lean precongruence for any language specified by a TSS in the ntyft/ntyxt format with recursion.

Proof: By Thm. 1 I may assume, without loss of generality, that $P = (\Sigma, R)$ is a TSS in the pure ntyft format with recursion. Let $\mathcal{R}$ be the smallest lean precongruence containing bisimilarity, i.e., $\mathcal{R} \subseteq \mathrm{T}(\Sigma) \times \mathrm{T}(\Sigma)$ is the smallest relation on processes satisfying

- if $p \sqsubseteq_B q$ then $p \mathrel{\mathcal{R}} q$,
- if $(f, n) \in \Sigma$ and $p_i \mathrel{\mathcal{R}} q_i$ for all $i = 1, ..., n$, then $f(p_1, \ldots, p_n) \mathrel{\mathcal{R}} f(q_1, \ldots, q_n)$,
- and if $S : V_S \rightarrow \mathbb{T}(\Sigma)$ with $Z \in V_S \subseteq \mathit{Var}$, and $\rho, \nu : \mathit{Var}\backslash V_S \rightarrow \mathrm{T}(\Sigma)$ satisfy $\rho(x) \mathrel{\mathcal{R}} \nu(x)$ for all $x \in \mathit{Var}\backslash V_S$, then $\langle Z|S\rangle[\rho] \mathrel{\mathcal{R}} \langle Z|S\rangle[\nu]$.

A trivial structural induction on $t \in \mathbb{T}(\Sigma)$, using the last two clauses, shows that if $\rho, \nu : \mathit{Var} \rightarrow \mathrm{T}(\Sigma)$ satisfy $\rho(x) \mathrel{\mathcal{R}} \nu(x)$ for all $x \in \mathit{var}(t)$, then $t[\rho] \mathrel{\mathcal{R}} t[\nu]$. (*)
As $\langle\_|S\rangle[\rho] : V_S \rightarrow \mathrm{T}(\Sigma)$ and $\langle\_|S\rangle[\nu] : V_S \rightarrow \mathrm{T}(\Sigma)$, this implies that in the last clause one even has $\langle t|S\rangle[\rho] \mathrel{\mathcal{R}} \langle t|S\rangle[\nu]$ for all terms $t \in \mathbb{T}(\Sigma, V_S)$. ($)

It suffices to show that $\mathcal{R}$ is a bisimulation, because this implies $\mathcal{R} \subseteq \sqsubseteq_B$, so that $\mathcal{R}$ equals $\sqsubseteq_B$, and (*) says that $\mathcal{R}$ is a lean precongruence. Thus I need to show that, for $p, q \in \mathrm{T}(\Sigma)$ and $a \in A$,

- if $p \mathrel{\mathcal{R}} q$ and $P \vdash p \xrightarrow{a} p'$, then there is a q' with $P \vdash q \xrightarrow{a} q'$ and $p' \mathrel{\mathcal{R}} q'$,
- if $p \mathrel{\mathcal{R}} q$ and $P \vdash q \stackrel{a}{\dashrightarrow} q'$, then there is a p' with $P \vdash p \stackrel{a}{\dashrightarrow} p'$ and $p' \mathrel{\mathcal{R}} q'$.

To this end it suffices to establish, for all ordinals λ, that

4. if $p \mathrel{\mathcal{R}} q$ and $P \vdash p \xrightarrow{a}_\lambda p'$, then there is a q' with $P \vdash q \xrightarrow{a} q'$ and $p' \mathrel{\mathcal{R}} q'$,
2. if $p \mathrel{\mathcal{R}} q$ and $P \vdash q \overset{a}{\dashrightarrow} q'$, then there is a p' with $P \vdash p \overset{a}{\dashrightarrow}_\lambda p'$ and $p' \mathrel{\mathcal{R}} q'$.

The desired result is then obtained by taking λ to be the closure ordinal κ of Def. 6. This I will do by induction on λ, at the same time establishing that

3. if $p \mathrel{\mathcal{R}} q$ and $P \vdash p \overset{a}{\not\rightarrow}_\lambda$, then $P \vdash q \overset{a}{\not\rightarrow}$,
1. if $p \mathrel{\mathcal{R}} q$ and $P \vdash q \overset{a}{\not\dashrightarrow}$, then $P \vdash p \overset{a}{\not\dashrightarrow}_\lambda$.

So assume Claims 1–4 have been established for all $\kappa < \lambda$.

Suppose $p \mathrel{\mathcal{R}} q$ and $P \vdash q \overset{a}{\not\dashrightarrow}$. By Remark 2 there is no $q' \in \mathrm{T}(\Sigma)$ with $P \vdash q \xrightarrow{a} q'$. So by induction, using Claim 4 above, there is no $p' \in \mathrm{T}(\Sigma)$ with $P \vdash p \xrightarrow{a}_\kappa p'$ for some $\kappa < \lambda$. By Def. 5 $P \vdash p \overset{a}{\not\dashrightarrow}_\lambda$. This yields Claim 1.

Now suppose $p \mathrel{\mathcal{R}} q$ and $P \vdash q \overset{a}{\dashrightarrow} q'$. I need to find a p' with $P \vdash p \overset{a}{\dashrightarrow}_\lambda p'$ and $p' \mathrel{\mathcal{R}} q'$. This I will do by structural induction on the proof π of $q \overset{a}{\dashrightarrow} q'$ from P. I make a case distinction based on the derivation of $p \mathrel{\mathcal{R}} q$.

- Let $p \sqsubseteq_B q$. Using that $\sqsubseteq_B$ is a bisimulation, there must be a process p' such that $P \vdash p \overset{a}{\dashrightarrow} p'$ and $p' \sqsubseteq_B q'$, hence $p' \mathrel{\mathcal{R}} q'$. Since $P \vdash p \overset{a}{\dashrightarrow} p'$, certainly $P \vdash p \overset{a}{\dashrightarrow}_\lambda p'$, by Remark 1.
- Let $p = f(p_1, \ldots, p_n)$ and $q = f(q_1, \ldots, q_n)$ where $p_i \mathrel{\mathcal{R}} q_i$ for $i = 1, \ldots, n$. Let π be a proof of $q \overset{a}{\dashrightarrow} q'$ from P. By Defs. 4 and 13, there must be a pure ntyft rule $r = \frac{H}{f(x_1,\ldots,x_n) \xrightarrow{a} t}$ in R and a closed substitution ν with $\nu(x_i) = q_i$ for $i = 1, ..., n$ and $t[\nu] = q'$, such that for each $(t_y \xrightarrow{c} y) \in H$ the transition $t_y[\nu] \overset{c}{\dashrightarrow} \nu(y)$ is provable from P by means of a strict subproof of π, and $P \vdash u[\nu] \overset{c}{\not\dashrightarrow}$ for each $(u \overset{c}{\not\rightarrow}) \in H$. Next, I define a substitution $\sigma : var(r) \rightarrow \mathrm{T}(\Sigma)$ such that
  (i) $\sigma(x_i) = p_i$ for $i = 1, \ldots, n$,
  (ii) $\sigma(y) \mathrel{\mathcal{R}} \nu(y)$ for each $y \in var(r)$,
  (iii) $P \vdash t_y[\sigma] \overset{c}{\dashrightarrow}_\lambda \sigma(y)$ for each $(t_y \xrightarrow{c} y) \in H$.

  The definition of $\sigma(y)$ and the inference of (i)–(iii) above proceed with induction on the distance of $y \in var(r)$ from the source of r,

  Base case: Let $\sigma(x_i) := p_i$ for $i = 1, \ldots, n$, so that Property (i) is satisfied. Regarding Property (ii), $\sigma(x_i) \mathrel{\mathcal{R}} \nu(x_i)$ for $i = 1, \ldots, n$.

  Induction step: When defining $\sigma(y)$ for some $y \in Var$ with $(t_y \xrightarrow{c} y) \in H$, by induction $\sigma(x)$ has been defined already for all $x \in var(t_y)$, so I may assume that $\sigma(x) \mathrel{\mathcal{R}} \nu(x)$ for all $x \in var(t_y)$ and hence $t_y[\sigma] \mathrel{\mathcal{R}} t_y[\nu]$ by (*). By induction on π, there is a p_y with $P \vdash t_y[\sigma] \overset{c}{\dashrightarrow}_\lambda p_y$ and $p_y \mathrel{\mathcal{R}} \nu(y)$. Define $\sigma(y) := p_y$. Properties (ii) and (iii) now hold for y.

  Take $p' := t[\sigma]$. So $p' = t[\sigma] \mathrel{\mathcal{R}} t[\nu] = q'$ by (*) and Property (ii) of σ. For each premise $(u \overset{c}{\not\rightarrow}) \in H$ one has $u[\sigma] \mathrel{\mathcal{R}} u[\nu]$ by (*) and Property (ii) of σ. So $P \vdash u[\sigma] \overset{c}{\not\dashrightarrow}_\lambda$ by Claim 1. By Defs. 4 and 13, together with Property (iii) of σ, this implies $P \vdash p = f(p_1, \ldots, p_n) \overset{a}{\dashrightarrow}_\lambda t[\sigma] = p'$.
- Let $p = \langle Z|S\rangle[\rho] = \langle Z|S[\rho]\rangle$ and $q = \langle Z|S\rangle[\sigma] = \langle Z|S[\sigma]\rangle$ where $S : V_S \rightarrow \mathbb{T}(\Sigma)$ with $Z \in V_S \subseteq Var$, $\rho, \sigma : Var \backslash V_S \rightarrow \mathrm{T}(\Sigma)$, and for all $x \in Var \setminus V_S$ one has $\rho(x) \mathrel{\mathcal{R}} \sigma(x)$. Let π be a proof of $q \overset{a}{\dashrightarrow} q'$ from P. By Defs. 4 and 13 $\langle S_Z|S[\sigma]\rangle \overset{a}{\dashrightarrow} q'$ is provable from P by means of a strict subproof of π. By (\$) above one has $\langle S_Z|S[\rho]\rangle \mathrel{\mathcal{R}} \langle S_Z|S[\sigma]\rangle$. So by induction there is a p' such that $P \vdash \langle S_Z|S[\rho]\rangle \overset{a}{\dashrightarrow}_\lambda p'$ and $p' \mathrel{\mathcal{R}} q'$. By Defs. 4 and 13, $P \vdash p = \langle Z|S[\sigma]\rangle \overset{a}{\dashrightarrow}_\lambda p'$.

Next, suppose that $p \mathrel{\mathcal{R}} q$ and $P \vdash p \overset{a}{\not\rightarrow}_\lambda$. By Def. 5 there is no $p' \in \mathrm{T}(\Sigma)$ with $P \vdash p \overset{a}{\dashrightarrow}_\lambda p'$. Using Claim 2, there is no $q' \in \mathrm{T}(\Sigma)$ with $P \vdash q \overset{a}{\dashrightarrow} q'$. By Remark 2, $P \vdash q \overset{a}{\not\rightarrow}$. This yields Claim 3.

Finally, suppose $p \mathrel{\mathcal{R}} q$ and $P \vdash p \xrightarrow{a}_\lambda p'$. I need to find a q' with $P \vdash q \xrightarrow{a} q'$ and $p' \mathrel{\mathcal{R}} q'$. This I will do by structural induction on the proof π of $p \xrightarrow{a}_\lambda p'$ from P. I make a case distinction based on the derivation of $p \mathrel{\mathcal{R}} q$.

- Let $p \sqsubseteq_B q$. Since $P \vdash p \xrightarrow{a}_\lambda p'$, certainly $P \vdash p \xrightarrow{a} p'$, by Remark 1. Using that $\sqsubseteq_B$ is a bisimulation, there must be a process q' such that $P \vdash q \xrightarrow{a} q'$ and $p' \sqsubseteq_B q'$, hence $p' \mathrel{\mathcal{R}} q'$.
- Let $p = f(p_1, \ldots, p_n)$ and $q = f(q_1, \ldots, q_n)$ where $p_i \mathrel{\mathcal{R}} q_i$ for $i = 1, \ldots, n$. Let π be a proof of $p \xrightarrow{a}_\lambda p'$ from P. By Defs. 4, 5 and 13, there must be a pure ntyft rule $r = \frac{H}{f(x_1,\ldots,x_n) \xrightarrow{a} t}$ in R and a closed substitution σ with $\sigma(x_i) = p_i$ for $i = 1, ..., n$ and $t[\sigma] = p'$, such that for each $(t_y \xrightarrow{c} y) \in H$ the transition $t_y[\sigma] \xrightarrow{c}_\lambda \sigma(y)$ is provable from P by means of a strict subproof of π, and $P \vdash u[\sigma] \overset{c}{\not\rightarrow}_\lambda$ for each $(u \overset{c}{\not\rightarrow}) \in H$. Next, I define a substitution $\nu : var(r) \rightarrow \mathrm{T}(\Sigma)$ such that
  (i) $\nu(x_i) = q_i$ for $i = 1, \ldots, n$,
  (ii) $\sigma(y) \mathrel{\mathcal{R}} \nu(y)$ for each $y \in var(r)$,
  (iii) $P \vdash t_y[\nu] \xrightarrow{c} \nu(y)$ for each $(t_y \xrightarrow{c} y) \in H$.

  The definition of $\nu(y)$ and the inference of (i)–(iii) above proceed with induction on the distance of $y \in var(r)$ from the source of r,

  Base case: Let $\nu(x_i) := q_i$ for $i = 1, \ldots, n$, so that Property (i) is satisfied. Regarding Property (ii), $\sigma(x_i) \mathrel{\mathcal{R}} \nu(x_i)$ for $i = 1, \ldots, n$.

  Induction step: When defining $\nu(y)$ for some $y \in Var$ with $(t_y \xrightarrow{c} y) \in H$, by induction $\nu(x)$ has been defined already for all $x \in var(t_y)$, so I may assume that $\sigma(x) \mathrel{\mathcal{R}} \nu(x)$ for all $x \in var(t_y)$ and hence $t_y[\sigma] \mathrel{\mathcal{R}} t_y[\nu]$ by (*). By induction on π, there is a q_y with $P \vdash t_y[\nu] \xrightarrow{c} q_y$ and $\sigma(y) \mathrel{\mathcal{R}} q_y$. Define $\nu(y) := q_y$. Properties (ii) and (iii) now hold for y.

  Take $q' := t[\nu]$. So $p' = t[\sigma] \mathrel{\mathcal{R}} t[\nu] = q'$ by (*) and Property (ii) of ν. For each premise $(u \overset{c}{\not\rightarrow}) \in H$ one has $u[\sigma] \mathrel{\mathcal{R}} u[\nu]$ by (*) and Property (ii) of ν. So $P \vdash u[\nu] \overset{c}{\not\rightarrow}$ by Claim 3. Since CT^+ is closed under deduction, together with Property (iii) of ν this implies $P \vdash q = f(q_1, \ldots, q_n) \xrightarrow{a} t[\nu] = q'$.
- Let $p = \langle Z|S\rangle[\rho] = \langle Z|S[\rho]\rangle$ and $q = \langle Z|S\rangle[\nu] = \langle Z|S[\nu]\rangle$ where $S : V_S \rightarrow \mathbb{T}(\Sigma)$ with $Z \in V_S \subseteq Var$, $\rho, \nu :$

$Var \backslash V_S \to \mathrm{T}(\Sigma)$, and for all $x \in Var \setminus V_S$ one has $\rho(x) \mathrel{\mathcal{R}} \nu(x)$. Let π be a proof of $p \xrightarrow{a}_\lambda p'$ from P. By Defs. 4, 5 and 13 $\langle S_Z|S[\rho]\rangle \xrightarrow{a}_\lambda p'$ is provable from P by means of a strict subproof of π. By ($) above one has $\langle S_Z|S[\rho]\rangle \mathrel{\mathcal{R}} \langle S_Z|S[\nu]\rangle$. So by induction there is a q' such that $P \vdash \langle S_Z|S[\nu]\rangle \xrightarrow{a} q'$ and $p' \mathrel{\mathcal{R}} q'$. By Defs. 4 and 13, $P \vdash q = \langle Z|S[\nu]\rangle \xrightarrow{a} q'$.

This yields Claim 4. □

The above result implies that any ntyft/ntyxt language with recursion satisfies congruence requirement (1) up to $\sqsubseteq_B$, but is not strong enough to yield (2).

VII. A FULL CONGRUENCE RESULT

In this section I deal with positive TSSs only. Here the relations $\dashrightarrow^{a}_\lambda$ and $\xrightarrow{a}_\mu$ for ordinals λ and μ all coincide, and $\sqsubseteq_B = \equiv_B$. The following auxiliary concept was used in [44] to show that CCS satisfies Condition (2) of Def. 11.

Definition 15 A symmetric relation $\mathcal{R} \subseteq \mathrm{T}(\Sigma) \times \mathrm{T}(\Sigma)$ is a *bisimulation up to* $\sim$ if $p \mathrel{\mathcal{R}} q$ and $P \vdash p \xrightarrow{a} p'$ imply that there is a q' with $P \vdash q \xrightarrow{a} q'$ and $p' \sim\mathcal{R}\sim q'$, for all $a \in A$. Here $\sim\mathcal{R}\sim := \{(r,s) \mid \exists r', s'.\ r \sim r' \mathrel{\mathcal{R}} s' \sim s\}$.

Proposition 4 ([44]) If $p \mathrel{\mathcal{R}} q$ for some bisimulation $\mathcal{R}$ up to $\equiv_B$, then $p \equiv_B q$.

Proof: Using the reflexivity of $\equiv_B$ it suffices to show that $\equiv_B\mathcal{R}\equiv_B$ is a bisimulation. Using symmetry and transitivity of $\equiv_B$ this is straightforward. □

Theorem 3 Bisimilarity is a full congruence for any language specified by a TSS in the tyft/tyxt format with recursion.

Proof: By Thm. 1 I may assume, without loss of generality, that $P = (\Sigma, R)$ is a TSS in the pure tyft format with recursion. Let $S, S' : W \to \mathbb{T}(\Sigma, W)$ be recursive specifications with $S_Y[\sigma] \equiv_B S'_Y[\sigma]$ for all $Y \in W$ and $\sigma : W \to \mathrm{T}(\Sigma)$.[7] I need to show that $\langle X|S\rangle \equiv_B \langle X|S'\rangle$ for all $X \in W$. Let $\mathcal{R} \subseteq \mathrm{T}(\Sigma) \times \mathrm{T}(\Sigma)$ be the smallest relation on processes satisfying

- $\langle X|S\rangle \mathrel{\mathcal{R}} \langle X|S'\rangle$ and $\langle X|S'\rangle \mathrel{\mathcal{R}} \langle X|S\rangle$ for all $X \in W$,
- if $(f,n) \in \Sigma$ and $p_i \mathrel{\mathcal{R}} q_i$ for all $i = 1,...,n$, then $f(p_1,\dots,p_n) \mathrel{\mathcal{R}} f(q_1,\dots,q_n)$,
- and if $S'' : V_{S''} \to \mathbb{T}(\Sigma)$ with $Z \in V_{S''} \subseteq Var$, and $\rho, \nu : Var \setminus V_{S''} \to \mathrm{T}(\Sigma)$ satisfy $\rho(x) \mathrel{\mathcal{R}} \nu(x)$ for all $x \in Var \setminus V_{S''}$, then $\langle Z|S''\rangle[\rho] \mathrel{\mathcal{R}} \langle Z|S''\rangle[\nu]$.

A trivial structural induction on $t \in \mathbb{T}(\Sigma)$, using the last two clauses, shows that if $\rho, \nu : Var \to \mathrm{T}(\Sigma)$ satisfy $\rho(x) \mathrel{\mathcal{R}} \nu(x)$ for all $x \in Var$, then $t[\rho] \mathrel{\mathcal{R}} t[\nu]$. (*)

So in the first clause one even has $\langle t|S\rangle \mathrel{\mathcal{R}} \langle t|S'\rangle$ for all $t \in \mathbb{T}(\Sigma, W)$, (#)

and in the last clause $\langle t|S''\rangle[\rho] \mathrel{\mathcal{R}} \langle t|S''\rangle[\nu]$ for all $t \in \mathbb{T}(\Sigma, V_{S''})$. ($)

It suffices to show that $\mathcal{R}$ is a bisimulation up to $\equiv_B$, because with Prop. 4 this implies $\mathcal{R} \subseteq \equiv_B$. By construction $\mathcal{R}$ is symmetric. So it suffices to show that,

$$\text{if } p \mathrel{\mathcal{R}} q \text{ and } P \vdash p \xrightarrow{a} p', \text{ then there is a } q' \text{ with } P \vdash q \xrightarrow{a} q' \text{ and } p' \mathrel{\mathcal{R}}\!\equiv_B q',$$

for all $p, q \in \mathrm{T}(\Sigma)$ and $a \in A$, This I will do by structural induction on the proof π of $p \xrightarrow{a} p'$ from P. I make a case distinction based on the derivation of $p \mathrel{\mathcal{R}} q$.

- Let $p = \langle X|S\rangle$ and $q = \langle X|S'\rangle$ with $X \in W$. Let π be a proof of $p \xrightarrow{a} p'$ from P. By Definitions 4 and 13 $\langle S_X|S\rangle \xrightarrow{a} p'$ is provable from P by means of a strict subproof of π. By (#) above one has $\langle S_X|S\rangle \mathrel{\mathcal{R}} \langle S_X|S'\rangle$. So by induction there is an r' such that $P \vdash \langle S_X|S'\rangle \xrightarrow{a} r'$ and $p' \mathrel{\mathcal{R}}\!\equiv_B r'$. Since $\langle \_\_|S'\rangle$ is a substitution of the form $\sigma : W \to \mathrm{T}(\Sigma)$, one has $\langle S_X|S'\rangle \equiv_B \langle S'_X|S'\rangle$. Hence there is a q' such that $P \vdash \langle S'_X|S'\rangle \xrightarrow{a} q'$ and $r' \equiv_B q'$. So $p' \mathrel{\mathcal{R}}\!\equiv_B q'$. By Definitions 4 and 13 $P \vdash q = \langle X|S'\rangle \xrightarrow{a} q'$.
- The case $p = \langle X|S'\rangle$ and $q = \langle X|S\rangle$ goes likewise, swapping the rôles of S'_X and S_X, and using the substitution $\langle \_\_|S\rangle$.[7]
- The remaining two cases proceed in the same way as in the proof of Claim 4 for Thm. 2, but suppressing λ and with $\mathcal{R}\!\equiv_B$ substituted for the blue occurrences of $\mathcal{R}$. In the last case there are no further changes, so I will not repeat it here. The remaining case needs a few elaborations—these involve the blue coloured segments in the proof of Claim 4:
- Let $p{=}f(p_1,\dots,p_n)$ and $q{=}f(q_1,\dots,q_n)$ where $p_i \mathrel{\mathcal{R}} q_i$ for $i = 1,\dots,n$. Let π be a proof of $p \xrightarrow{a} p'$ from P. By Defs. 4 and 13, there must be a pure tyft rule $r = \frac{H}{f(x_1,\dots,x_n) \xrightarrow{a} t}$ in R and a closed substitution σ with $\sigma(x_i) = p_i$ for $i = 1,...,n$ and $t[\sigma] = p'$, such that for each $(t_y \xrightarrow{c} y) \in H$ the transition $t_y[\sigma] \xrightarrow{c} \sigma(y)$ is provable from P by means of a strict subproof of π. Next, I define a substitution $\nu : var(r) \to \mathrm{T}(\Sigma)$ such that
  (i) $\nu(x_i) = q_i$ for $i = 1,\dots,n$,
  (ii) $\sigma(y) \mathrel{\mathcal{R}}\!\equiv_B \nu(y)$ for each $y \in var(r)$,
  (iii) $P \vdash t_y[\nu] \xrightarrow{c} \nu(y)$ for each $(t_y \xrightarrow{c} y) \in H$.

  The definition of $\nu(y)$ and the inference of (i)–(iii) above proceed with induction on the distance of $y \in var(r)$ from the source of r,

  Base case: Let $\nu(x_i) := q_i$ for $i = 1,\dots,n$, so that Property (i) is satisfied. Regarding Property (ii), $\sigma(x_i) \mathrel{\mathcal{R}} \nu(x_i)$ for $i = 1,\dots,n$.

  Induction step: When defining $\nu(y)$ for some $y \in Var$ with $(t_y \xrightarrow{c} y) \in H$, by induction $\nu(x)$ has been defined already for all $x \in var(t_y)$, so I may assume that $\sigma(x) \mathrel{\mathcal{R}}\!\equiv_B \nu(x)$ for all $x \in var(t_y)$, i.e., there exists a substitution $\rho : var(r) \to \mathrm{T}(\Sigma)$ with $\sigma(x) \mathrel{\mathcal{R}} \rho(x) \equiv_B \nu(x)$ for all $x \in var(t_y)$. Now $t_y[\sigma] \mathrel{\mathcal{R}} t_y[\rho]$ by (*) and $t_y[\rho] \equiv_B t_y[\nu]$ by Thm. 2.
  By induction on π, there is an r_y with $P \vdash t_y[\rho] \xrightarrow{c} r_y$ and $\sigma(y) \mathrel{\mathcal{R}}\!\equiv_B r_y$. By the definition of bisimilarity, there

[7] This proof shows that in the full congruence property (2) one only needs to assume $S_Y[\sigma] \equiv_B S'_Y[\sigma]$ for two specific substitutions σ: namely $\sigma(Y) := \langle Y|S'\rangle$, resp. $\langle Y|S\rangle$.

is a q_y with $P \vdash t_y[\nu] \xrightarrow{c} q_y$ and $r_y \equiv_B q_y$. Define $\nu(y) := q_y$. Properties (ii) and (iii) now hold for y.

Take $q' := t[\nu]$. So $p' = t[\sigma] \; \mathcal{R}{\equiv} \; t[\nu] = q'$ by (*), Property (ii) of ν, and Thm. 2. By Defs. 4 and 13, together with Property (iii) of ν, this implies
$P \vdash q = f(q_1, \ldots, q_n) \xrightarrow{a} t[\nu] = q'$. □

It remains an open question whether the above result can be generalised to the ntyft/ntyxt format with recursion. A direct combination of the proofs of Thms. 2 and 3 does not work, however. An attempt in this direction would substitute either $\mathcal{R}{\sqsubseteq_B}$ or ${\sqsubseteq_B}\mathcal{R}$ for the red $\mathcal{R}$ in Claim 2 in the proof of Thm. 2. Both attempts fail on the case $p = \langle X|S\rangle$ and $q = \langle X|S'\rangle$ in the proof of Thm. 3.

The first attempt would from $P \vdash \langle S'_X|S'\rangle \overset{a}{\dashrightarrow} q'$ infer $P \vdash \langle S_X|S'\rangle \overset{a}{\dashrightarrow} r'$ by bisimilarity, and then infer $P \vdash \langle S_X|S\rangle \overset{a}{\dashrightarrow}_\lambda p'$ by induction. However, one may not use induction, as the transition $\langle S_X|S'\rangle \overset{a}{\dashrightarrow} r'$ may be derived later than $\langle X|S'\rangle \overset{a}{\dashrightarrow} q'$. In fact, if a variant of this approach would work, skipping $\langle X|S'\rangle \; \mathcal{R} \; \langle X|S\rangle$ from the definition of $\mathcal{R}$, one could prove a false version of (2) that assumes the antecedent only for the single substitution $\langle \_\_ |S'\rangle$ (cf. Footnote 7); it is trivial to find a counterexample in the GSOS format with unguarded recursion.

The second attempt would from $P \vdash \langle S'_X|S'\rangle \overset{a}{\dashrightarrow} q'$ infer $P \vdash \langle S'_X|S\rangle \overset{a}{\dashrightarrow}_\lambda r'$ by induction, and then $P \vdash \langle S_X|S\rangle \overset{a}{\dashrightarrow}_\lambda p'$ by bisimilarity. The latter step is invalid, as $\langle S_X|S'\rangle \overset{a}{\dashrightarrow}_\lambda r'$ is only an overapproximation of $P \vdash \langle S_X|S'\rangle \overset{a}{\dashrightarrow} r'$.

References

[1] L. Aceto, A. Birgisson, A. Ingólfsdóttir, M.R. Mousavi & M.A. Reniers (2012): *Rule formats for determinism and idempotence*. Science of Computer Programming 77(7-8), pp. 889–907, doi:10.1016/j.scico.2010.04.002.

[2] L. Aceto, M. Cimini, A. Ingólfsdóttir, M.R. Mousavi & M.A. Reniers (2011): *SOS rule formats for zero and unit elements*. Theoretical Computer Science 412(28), pp. 3045–3071, doi:10.1016/j.tcs.2011.01.024.

[3] L. Aceto, M. Cimini, A. Ingólfsdóttir, M.R. Mousavi & M.A. Reniers (2012): *Rule formats for distributivity*. Theoretical Computer Science 458, pp. 1–28, doi:10.1016/j.tcs.2012.07.036.

[4] L. Aceto, W.J. Fokkink & C. Verhoef (2000): *Structural Operational Semantics*. In J.A. Bergstra, A. Ponse & S.A. Smolka, editors: Handbook of Process Algebra, chapter 3, Elsevier, pp. 197–292.

[5] G. Bacci & M. Miculan (2015): *Structural operational semantics for continuous state stochastic transition systems*. Journal of Computer and System Sciences 81(5), pp. 834–858, doi:10.1016/j.jcss.2014.12.003.

[6] J. C. M. Baeten, T. Basten & M. A. Reniers (2010): *Process Algebra: Equational Theories of Communicating Processes*. Cambridge University Press.

[7] J.C.M. Baeten, editor (1990): *Applications of Process Algebra*. Cambridge Tracts in Theoretical Computer Science 17, Cambridge University Press.

[8] F. Bartels (2002): *GSOS for Probabilistic Transition Systems*. Electr. Notes Theor. Comput. Sci. 65(1), pp. 29–53, doi:10.1016/S1571-0661(04)80358-X.

[9] B. Bloom (1995): *Structural operational semantics for weak bisimulations*. Theoretical Computer Science 146, pp. 25–68, doi:10.1016/0304-3975(94)00152-9.

[10] B. Bloom, W.J. Fokkink & R.J. van Glabbeek (2004): *Precongruence Formats for Decorated Trace Semantics*. Transactions on Computational Logic 5(1), pp. 26–78, doi:10.1145/963927.963929.

[11] B. Bloom, S. Istrail & A.R. Meyer (1995): *Bisimulation Can't be Traced*. Journal of the ACM 42(1), pp. 232–268, doi:10.1145/200836.200876.

[12] R.N. Bol & J.F. Groote (1996): *The meaning of negative premises in transition system specifications*. Journal of the ACM 43(5), pp. 863–914, doi:10.1145/234752.234756.

[13] E. Bres, R.J. van Glabbeek & P. Höfner (2016): *A Timed Process Algebra for Wireless Networks with an Application in Routing*. Technical Report 9145, NICTA. Available at http://arxiv.org/abs/1606.03663. Extended abstract in P. Thiemann, editor: Proc. ESOP'16, LNCS 9632, Springer, 2016, pp. 95-122.

[14] 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, doi:10.1145/828.833.

[15] S. Cranen, M. R. Mousavi & M. A. Reniers (2008): *A Rule Format for Associativity*. In F. van Breugel & M. Chechik, editors: Proc. CONCUR'08, LNCS 5201, Springer, pp. 447–461, doi:10.1007/978-3-540-85361-9_35.

[16] P.R. D'Argenio, D. Gebler & M.D. Lee (2016): *A general SOS theory for the specification of probabilistic transition systems*. Information and Computation 249, pp. 76–109, doi:10.1016/j.ic.2016.03.009.

[17] W. J. Fokkink (2000): *Introduction to Process Algebra*. Texts in Theoretical Computer Science, An EATCS Series, Springer, doi:10.1007/978-3-662-04293-9.

[18] W.J. Fokkink (2000): *Rooted Branching Bisimulation as a Congruence*. Journal of Computer and System Sciences 60(1), pp. 13–37, doi:10.1006/jcss.1999.1663.

[19] W.J. Fokkink & R.J. van Glabbeek (1996): *Ntyft/ntyxt rules reduce to ntree rules*. Information and Computation 126(1), pp. 1–10, doi:10.1006/inco.1996.0030.

[20] W.J. Fokkink & R.J. van Glabbeek (2016): *Divide and Congruence II: Delay and Weak Bisimilarity*. In: Proc. LICS'16, ACM, pp. 778–787, doi:10.1145/2933575.2933590.

[21] W.J. Fokkink & R.J. van Glabbeek (2017): *Precongruence Formats with Lookahead through Modal Decomposition*. Available at http://theory.stanford.edu/~rvg/abstracts.html#122.

[22] W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2006): *Compositionality of Hennessy-Milner Logic by Structural Operational Semantics*. Theoretical Computer Science 354(3), pp. 421–440, doi:10.1016/j.tcs.2005.11.035.

[23] W.J. Fokkink, R.J. van Glabbeek & P. de Wind (2012): *Divide and congruence: From decomposition of modal formulas to preservation of branching and η-bisimilarity*. Information and Computation 214, pp. 59–85, doi:10.1016/j.ic.2011.10.011.

[24] H. Garavel, F. Lang, R. Mateescu & W. Serwe (2011): *CADP 2010: A Toolbox for the Construction and Analysis of Distributed Processes*. In P.A. Abdulla & K.R.M. Leino, editors: Proc. TACAS'11, LNCS 6605, Springer, pp. 372–387, doi:10.1007/978-3-642-19835-9_33.

[25] D. Gebler & S. Tini (2013): *Compositionality of Approximate Bisimulation for Probabilistic Systems*. In Johannes Borgström & Bas Luttik, editors: Proc. EXPRESS/SOS'13, EPTCS 120, Open Publishing Association, pp. 32–46, doi:10.4204/EPTCS.120.4.

[26] A. van Gelder, K. Ross & J.S. Schlipf (1991): *The well-founded semantics for general logic programs*. Journal of the ACM 38(3), pp. 620–650, doi:10.1145/116825.116838.

[27] R.J. van Glabbeek (1993): *Full abstraction in structural operational semantics (extended abstract)*. In M. Nivat, C. Rattray, T. Rus & G. Scollo, editors: Proc. AMAST'93, Workshops in Computing, Springer, pp. 77–84. Available at http://theory.stanford.edu/~rvg/abstracts.html#28.

[28] R.J. van Glabbeek (1993): *The Linear Time – Branching Time Spectrum II; The semantics of sequential systems with silent moves (extended abstract)*. In E. Best, editor: Proc. CONCUR'93, LNCS 715, Springer, pp. 66–81, doi:10.1007/3-540-57208-2_6.

[29] R.J. van Glabbeek (1994): *On the expressiveness of ACP (extended abstract)*. In A. Ponse, C. Verhoef & S.F.M. van Vlijmen, editors: Proceedings First Workshop on the Algebra of Communicating Processes, ACP94, Utrecht, The Netherlands, May 1994, Workshops in Computing, Springer, pp. 188–217. Available at http://theory.stanford.edu/~rvg/abstracts.html#31.

[30] R.J. van Glabbeek (1995): *The Meaning of Negative Premises in Transition System Specifications II*. Technical Report STAN-CS-TN-95-16, Stanford University. Available at http://theory.stanford.edu/~rvg/abstracts.html#32. Extended abstract in F. Meyer auf der Heide & B. Monien, editors: Proc. ICALP'96, LNCS 1099, Springer, pp. 502–513, doi: 10.1007/3-540-61440-0_154.

[31] R.J. van Glabbeek (2004): *The Meaning of Negative Premises in Transition System Specifications II*. Journal of Logic and Algebraic Programming 60–61, pp. 229–258, doi:10.1016/j.jlap.2004.03.007.

[32] R.J. van Glabbeek (2011): *On Cool Congruence Formats for Weak Bisimulations*. Theoretical Computer Science 412(28), pp. 3283–3302, doi:10.1016/j.tcs.2011.02.036.

[33] R.J. van Glabbeek (2012): *Musings on Encodings and Expressiveness*. In B. Luttik & M.A. Reniers, editors: Proc. EXPRESS/SOS'12, EPTCS 89, Open Publishing Association, pp. 81–98, doi:10.4204/EPTCS.89.7.

[34] R.J. van Glabbeek, B. Luttik & N. Trčka (2009): *Branching Bisimilarity with Explicit Divergence*. Fundamenta Informaticae 93(4), pp. 371–392.

[35] R.J. van Glabbeek & W.P. Weijland (1989): *Branching Time and Abstraction in Bisimulation Semantics (extended abstract)*. In G.X. Ritter, editor: Information Processing 89, Proceedings of the IFIP 11th World Computer Congress, San Francisco 1989, North-Holland, pp. 613–618. Full version in Jounal of the ACM 43(3), 1996, pp. 555–600.

[36] J.F. Groote (1993): *Transition System Specifications with Negative Premises*. Theoretical Computer Science 118, pp. 263–299, doi:10.1016/0304-3975(93)90111-6.

[37] J.F. Groote & M.R. Mousavi (2014): *Modeling and Analysis of Communicating Systems*. MIT Press.

[38] J.F. Groote, M.R. Mousavi & M.A. Reniers (2006): *A Hierarchy of SOS Rule Formats*. Electr. Notes Theor. Comput. Sci. 156(1), pp. 3–25, doi:10.1016/j.entcs.2005.11.077.

[39] J.F. Groote & F.W. Vaandrager (1992): *Structured Operational Semantics and Bisimulation as a Congruence*. Information and Computation 100(2), pp. 202–260, doi:10.1016/0890-5401(92)90013-6.

[40] B. Klin & V. Sassone (2013): *Structural operational semantics for stochastic and weighted transition systems*. Information and Computation 227, pp. 58–83, doi:10.1016/j.ic.2013.04.001.

[41] R. Lanotte & S. Tini (2009): *Probabilistic bisimulation as a congruence*. ACM Transactions on Computational Logic 10(2):9, doi:10.1145/1462179.1462181.

[42] K.G. Larsen & B. Thomsen (1988): *A Modal Process Logic*. In: Proc. LICS'88, IEEE Computer Society Press, pp. 203–210, doi:10.1109/LICS.1988.5119.

[43] M. Miculan & M. Peressotti (2014): *GSOS for non-deterministic processes with quantitative aspects*. In Nathalie Bertrand & Luca Bortolussi, editors: Proc. QAPL'14, EPTCS 154, Open Publishing Association, pp. 17–33, doi:10.4204/EPTCS.154.2.

[44] 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, 1980.

[45] M.R. Mousavi, M.A. Reniers & J.F. Groote (2005): *A syntactic commutativity format for SOS*. Information Processing Letters 93(5), pp. 217–223, doi:10.1016/j.ipl.2004.11.007.

[46] G.D. Plotkin (2004): *A Structural Approach to Operational Semantics*. The Journal of Logic and Algebraic Programming 60–61, pp. 17–139, doi:10.1016/j.jlap.2004.05.001. Originally appeared in 1981.

[47] T.C. Przymusinski (1990): *The Well-founded Semantics Coincides with the Three-valued Stable Semantics*. Fundamenta Informaticae XIII(4), pp. 445–463.

[48] R. de Simone (1984): *Calculabilité et Expressivité dans l'Algebra de Processus Parallèles* MEIJE. Thèse de 3^{e} cycle, Univ. Paris 7.

[49] R. de Simone (1985): *Higher-level synchronising devices in* MEIJE-*SCCS*. Theoretical Computer Science 37, pp. 245–267, doi:10.1016/0304-3975(85)90093-3.

[50] I. Ulidowski (1992): *Equivalences on Observable Processes*. In: Proc. LICS'92, IEEE Computer Society Press, pp. 148–159, doi:10.1109/LICS.1992.185529.

[51] I. Ulidowski (2000): *Finite axiom systems for testing preorder and De Simone process languages*. Theoretical Computer Science 239(1), pp. 97–139, doi:10.1016/S0304-3975(99)00214-5.

[52] I. Ulidowski & I. Phillips (2002): *Ordered SOS Rules and Process Languages for Branching and Eager Bisimulations*. Information and Computation 178(1), pp. 180–213, doi:10.1006/inco.2002.3161.