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 ntft/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 applied to arguments p1, ..., pn is completely determined by the equivalence classes of these arguments.

Equally important is that the chosen equivalence relation ~ is a congruence for recursion. Recursion allows the specification of a process as a canonical solution of an equation $X = E(X)$.

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 ~ 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 ~ be bisimilarity, so that $b.0 \sim b.0 + b.0$ [44]. Now the lean congruence requirement for ~ 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 role in the study of expressiveness of system description languages [33]. There, correct translations of one language into another up to a semantic equivalence ~ are defined; and expressiveness hierarchies—one for each choice of ~—are defined in terms of those translations. However, a correct translation can exist only when ~ 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 $X = a.X + a.X$. 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 ntft/ntyxt format of GROOTE [56] 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].

Formats for weak semantics—

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

2These 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], [13], [51], [52], [53], [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. 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 ntyft/ntyxt languages.

My proof strategy follows the traditional method of [11], [69], [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 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 \(f \in \text{Var}\) and an arity \(n \in \mathbb{N}\).

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.

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 [43], 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.

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 \(\Sigma\) is defined inductively by:

- \(\text{Var} \subseteq \mathbb{T}(\Sigma)\),
- if \((f, n)\in\Sigma\) and \(t_1, \ldots, t_n \in \mathbb{T}(\Sigma)\) then \(f(t_1, \ldots, t_n) \in \mathbb{T}(\Sigma)\),
- If \(V_S \subseteq \text{Var}\), \(S : V_S \rightarrow \mathbb{T}(\Sigma)\) 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 \(\text{var}(t)\) 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 \(\Sigma\) with \(\text{var}(t) \subseteq W\). \(\mathbb{T}(\Sigma) := \mathbb{T}(\Sigma, \emptyset)\) is set of closed terms over \(\Sigma\).

Example 1 Let \(\Sigma\) contain three unary functions \(a_\ldots \ b_\ldots\) and \(d_\ldots\) and one infix-written binary function \(\parallel\). Let \(X, Y, Z \in \text{Var}\). Then \(S = \{X = (a.X)(b.Y), Y = (d.Y)(X\parallel Z)\}\) 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 \(S\).

The term \(\langle X|S\rangle\) denotes the \(X\)-component of such a tuple.

Definition 2 (Substitution) A \(\Sigma\)-substitution \(\sigma\) is a partial function from \(\text{Var} \to \mathbb{T}(\Sigma)\); it is closed if it is a total function from \(\text{Var} \to \mathbb{T}(\Sigma)\). If \(\sigma\) 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 \(\sigma\), 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 \(\sigma\) 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 \(t[S]\) abbreviates \(t[[Y|S|Y]|Y \in V_S]\).

Example 2 Extend \(\Sigma\) from Ex. 1 with a constant \(c\). Then \(\langle X|S|b.c/z\rangle = \langle X|X = (a.X)(b.Y), Y = (d.Y)(X\parallel b.c)\rangle\), \(\langle X|S|X/z\rangle = \langle Z|Z = (a.Z)(b.Y), Y = (d.Y)(Z\parallel X)\rangle\) and \(\langle X|S|b.c/Y\rangle = \langle X|S\rangle\).

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

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

The concept of a (positive) TSS presented above was introduced in Groote & Vaandrager [36]; the negative premises \( t \xrightarrow{\alpha} t' \) 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) [40] that is sufficiently general to cover many of its applications.

The following definition (from [21]) 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 \( t \xrightarrow{\beta} t' \) corresponds to the derivation of the transition rule \( \frac{H}{\beta} \) with \( H = \emptyset \). The case \( H \neq \emptyset \) corresponds to the derivation of \( t \xrightarrow{\alpha} t' \) 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 \( \Sigma \)-literals, such that:
- the root is labelled by \( \alpha \), and
- if \( \beta \) 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. But as pointed out in Groote [36] and Grellois [6], 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 [20], which in general yields a \( 3 \)-valued transition relation \( T : \text{Tr}(\Sigma) \times A \times \text{Tr}(\Sigma) \to \{\text{present}, \text{undetermined}, \text{absent}\} \). 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 underapproximation of transition relations)** Let \( P \) be a TSS. For ordinals \( \lambda \) the sets \( CT^+ \) and \( PT^+ \) of closed positive literals, and \( CT^- \), \( PT^- \) of closed negative literals are defined inductively by:

\[
\begin{align*}
PT^-_\lambda &= \text{the set of literals that do not deny any } \beta \in PT^+_\kappa \text{ with } \kappa < \lambda \\
CT^-_\lambda &= \text{the set of literals that do not deny any } \beta \in PT^+_\lambda \\
\end{align*}
\]

Intuitively, \( CT^+ \) is an underapproximation of the set of transitions that should be in the transition relation specified by \( P \), and \( PT^+ \) an overapproximation. Likewise, \( CT^- \) is an underapproximation of the set of closed negative literals that should hold, and \( PT^- \) an overapproximation. The approximations get better with increasing \( \lambda \). 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** \( CT^- \subseteq CT^- \subseteq PT^- \subseteq PT^- \) and \( CT^+ \subseteq CT^+ \subseteq PT^+ \subseteq PT^+ \) for \( \kappa < \lambda \).

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

---

6Readers 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 \( p \xrightarrow{\alpha} p' \) with \( p, p' \). In the proofs of Prop. 4 and Thm. 6 also \( p \xrightarrow{\alpha} p' \) and \( p \xrightarrow{\alpha} p' \) equal \( p \xrightarrow{\alpha} p' \); for any \( \alpha \), so the induction on \( \alpha \) can be skipped, as well as the auxiliary Claims 3 and 1, and the proof proceeds directly by induction on \( \pi \).
$PT^+_\lambda \subseteq PT^+_\lambda$, $CT^-_\kappa \subseteq CT^-_\lambda$ and $CT^+_\kappa \subseteq CT^+_\lambda$, respectively. The remaining claims follow by induction on $\lambda$.

As $PT^+_0$ is the universal relation, certainly $CT^-_0 \subseteq PT^+_0$, so $CT^+_1 \subseteq PT^+_1$.

Let $\lambda$ 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 $CT^-_\kappa \subseteq PT^-_\mu \subseteq PT^-_\mu$ 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 $CT^-_\kappa \subseteq PT^-_\kappa$ and hence $CT^-_\lambda \subseteq PT^-_\lambda$.

Now let $\lambda = \mu + 1$. By induction $CT^+_\mu \subseteq PT^+_\mu$. With Def. 5 this implies $CT^+_\mu \subseteq PT^+_\mu$, and hence $CT^+_\mu \subseteq PT^+_\lambda$. With Def. 5 this implies $CT^-_\mu \subseteq PT^-_\lambda$ and hence $CT^+_\mu \subseteq PT^+_\lambda$. □

Since the closed literals over $\Sigma$ form a proper set, there must be an ordinal $\kappa$ such that $PT^-_\kappa = PT^-_\kappa$ for all $\lambda > \kappa$, and hence also $PT^-_\lambda = PT^-_\kappa$. Moreover, $CT^-_\kappa \subseteq PT^-_\kappa$ and $CT^+_\kappa \subseteq PT^+_\kappa$.

Definition 6 Such an ordinal $\kappa$ is called a 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 $(CT^+, PT^+)$ 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 $\alpha$ is a well-founded tree with the nodes labelled by closed literals, such that the root is labelled by $\alpha$, and if $\beta$ is the label of a node and $K$ is the set of labels of the children of this node, then:

- either $\beta$ is positive and $\beta_\alpha$ is a substitution instance of a rule in $R$;
- or $\beta$ is negative and for each set $N$ of closed negative literals with $P \not\vdash N$ for $\gamma$ a closed positive literal denying $\beta$, a literal in $K$ denies one in $N$.

$P \vdash ws \alpha$ denotes that a well-supported proof from $P$ of $\alpha$ exists.

Proposition 1 Let $P$ be a TSS. Then $P \vdash ws p \not\rightarrow q$ iff $(p \not\rightarrow q) \in CT^+$, and $P \vdash ws p \rightarrow \beta$ iff $(p \rightarrow \beta) \in CT^-$. □

Proof: If $\pi$ be a well-supported proof of a closed literal $\alpha$. By consistently applying the same closed substitution to all literals occurring in $\pi$, one can assume, without loss of generality, that all literals in $\pi$ are closed. With structural induction on $\pi$ I show that $\alpha \in CT^+ \cup CT^-$. Suppose $\alpha$ is positive and $\pi_\alpha$ is the closed substitution instance of the rule of $P$ applied at the root of $\pi$. Then for each $\beta \in K$ the literal $\beta$ is ws-provable from $P$ by means of a strict subproof of $\pi$. By induction $\beta \in CT^+ \cup CT^-$. As $CT^+ = CT^+_\kappa$ for some ordinal $\kappa$, it is closed under deduction. Hence $\alpha \in CT^+$.

Suppose $\alpha$ is negative. Let $\beta$ be closed positive literal denning $\alpha$. By Def. 8 each set $N$ of closed negative literals with $P \not\vdash \beta$ contains a literal $\kappa_\alpha$ denying a literal $\delta_\alpha$ that is ws-provable from $P$ by means of a strict subproof of $\pi$. By induction $\delta_\alpha \in CT^+$. Hence $\gamma_\alpha \notin CT^-$. Consequently $\beta \notin CT^+$. Hence $\alpha \in CT^-$. □

Proposition 2 Suppose $\alpha \in CT^+_\lambda \cup CT^-_\lambda$. With induction on $\lambda$ I show that $P \vdash ws \alpha$. First suppose $\alpha \in CT^-_\lambda$. Let $N$ be a set of closed negative literals with $P \not\vdash N$ for $\gamma$ a closed positive literal denning $\alpha$. Assume that $N \subseteq PT^-_\lambda$. Then $\gamma$ would be in $PT^+_\lambda$, contradicting the definition of $CT^+_\lambda$. So $N$ contains a literal that is not in $PT^-_\lambda$, i.e., denies a literal $\delta_\lambda$ in $CT^+_\lambda$ for some $\kappa < \lambda$. By induction, $P \vdash ws \delta_\lambda$. It follows that $P \vdash ws \alpha$.

Now suppose $\alpha \in CT^+_\lambda$. Then $P \vdash CT^+_{\lambda+1}$. By the case above $P \vdash ws \beta$ for each $\beta \in CT^+_{\lambda+1}$. Hence $P \vdash ws \alpha$. □

The above result, together with Theorem 1 in [31], and the observation in [31] that literals $t \not\rightarrow t'$ 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 $P \vdash ws p \rightarrow a$ or $P \vdash ws p \not\rightarrow a$ for some $q$. This implies that $CT^-$ is exactly the set of closed negative literals that do not deny any literal in $CT^+$. Hence $CT^- = PT^+$. So 3-valued transition systems associated to a complete TSS is 2-valued.

Below I write $P \vdash p \rightarrow \lambda q$ for $(p \not\rightarrow q) \in CT^+_\lambda$, $P \vdash p \not\rightarrow \lambda q$ for $(p \not\rightarrow q) \in CT^-_\lambda$, $P \vdash p \rightarrow q$ for $p \rightarrow q \in PT^+_\lambda$ and $P \vdash p \not\rightarrow q$ for $(p \not\rightarrow q) \in PT^-_\lambda$. Moreover, $p \rightarrow a$ resp. $p \not\rightarrow a$, will abbreviate $p \rightarrow a_\lambda$ resp. $p \not\rightarrow a_\kappa$, where $\kappa$ is the closure ordinal of Def. 6.

In my forthcoming leon congruence proof I will apply structural induction on “the proof of a transition $p \rightarrow a_\lambda q$ or $p \not\rightarrow a_\lambda q$ from $P$”. There I will mean the proofs of $p \rightarrow a_\lambda$ and $p \not\rightarrow a_\lambda$, respectively, as this is what constitutes the evidence for the statement $P \vdash p \rightarrow a_\lambda q$, resp. $P \vdash p \not\rightarrow a_\lambda q$.

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 ntyf/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-valuated. Rather than
and parallel composition

\[ p \parallel q \]

Clearly, modal refinement is reflexive and transitive, and hence \( T(\Sigma) \times \rho \) contains the transitions that are a process with a 2-valued transition relation, in which the definition 11 (full precongruence) 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 \( T(\Sigma) \) such that, for \( p, q \in T(\Sigma) \) and \( a \in A \),

- if \( p \mathcal{R} q \) and \( P \vdash p \xrightarrow{a} p' \), then there is a \( q' \) with \( P \vdash q \xrightarrow{a} q' \) and \( p' \mathcal{R} q' \),
- if \( p \mathcal{R} q \) and \( \vdash p \xrightarrow{a} q \), then there is a \( p' \) with \( p \overset{a}{\rightarrow} p' \) and \( p' \mathcal{R} q' \).

A process \( q \in T(\Sigma) \) is a modal refinement of \( p \in T(\Sigma) \), notation \( p \sqsubseteq_B q \), if there exists a bisimulation \( \mathcal{R} \) with \( p \vdash q \mathcal{R} \). 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 \( T(\Sigma) \) of closed terms over \( \Sigma \). For \( \rho, \nu : \text{Var} \to T(\Sigma) \) closed substitutions write \( \rho \sqsubseteq \nu \) iff \( \rho(x) \subseteq \nu(x) \) for each \( x \in \text{Var} \).

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

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

\[ p_i \sqsubseteq q_i \text{ for all } i = 1, \ldots, n \quad \Rightarrow \quad f(p_1, \ldots, p_n) \sqsubseteq f(q_1, \ldots, q_n) \quad (1) \]

\[ S_Y[\sigma] \subseteq S'_Y[\sigma] \text{ for all } \sigma : W \to T(\Sigma) \quad \Rightarrow \quad \langle X | S \rangle \subseteq \langle X | S' \rangle \quad (2) \]

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

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

\[ a.X \xrightarrow{a} x \quad x \xrightarrow{a} x' \quad y \xrightarrow{a} y' \]

where \( a \) ranges over \( A \), and the recursion rule from Def. 13 below. An infinite trace of a process \( p \) is a sequence \( a_1 a_2 \ldots \in A^\omega \) such that there are processes \( p_1, p_2, \ldots \) with \( p \overset{a_1}{\rightarrow} p_1 \overset{a_2}{\rightarrow} p_2 \overset{a_3}{\rightarrow} \ldots \). Let \( p \sqsubseteq q \) iff for each infinite trace \( \sigma \) of \( p \) there is an infinite trace of \( q \) that has a suffix in common with \( \sigma \). This is a preorder indeed. It is not hard to check that \( \sqsubseteq \) is a precongruence for both action prefixing \( a.\_ \) and parallel composition \( \parallel \_ \), 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 \( 0 \) 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 \(0\). 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 \( 0 \sim \langle X | X = X \rangle \neq \langle X | X = \tau.X \rangle \), 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 \( t \in \text{T}(\Sigma, W) \) 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 \[53\]. 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 ntyt rule is an ntyxt rule if its source is a variable, an ntyf rule if its source contains exactly one function symbol and no multiple occurrences of variables, and an nxyt 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, ntyt 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 ∈ VS it has a rule

\[ \langle S_X | S \rangle \xrightarrow{a} z \]

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

Definition 14 (Well-founded and pure rules; distance) The dependency graph of an ntyt rule with \( \{ t_i \xrightarrow{a} y_i \mid i \in I \} \) as set of positive premises is the directed graph with edges \( \{ x_i, y_i \mid x_i \in \text{var}(t_i) \text{ for some } i \in I \} \). A ntyt 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 \( r = \frac{H}{t \xrightarrow{a} y} \) be a pure ntyt rule. The distance of a variable \( y \in \text{var}(r) \) to the source of \( r \) is the ordinal number given by

\[ \text{dist}(x) = 0 \quad \text{if } x \in \text{var}(t), \]

\[ \text{dist}(y) = 1 + \sup\{ \text{dist}(x) \mid x \in \text{var}(t) \} \quad \text{if } (t \xrightarrow{a} y) \in H. \]

Bol and 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/txtxt 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/txtxt (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/txtxt (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 ntyf format, such that for any closed transition rule \( N \) with only negative premises, one has \( P \vdash \sim N \Leftrightarrow P' \vdash \sim N \). 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 \( P \vdash \sim N \Leftrightarrow P' \vdash \sim N \) for any closed transition rule \( N \) 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: If \( \vdash \langle X | S \rangle \xrightarrow{a} q \) for some \( a \in A \) and \( q \in T(\Sigma) \) iff \( P \vdash \langle S_X | S \rangle \xrightarrow{a} q \). □

For the second, invariance under \( \alpha \)-conversion, write \( t \equiv u \) if the terms \( t, u \in T(\Sigma) \) 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 \( p \equiv q \Rightarrow p \equiv_B q \) for all \( p, q \in T(\Sigma) \).

Proof: By Thm. [1] I may assume, without loss of generality, that \( P \) is in the pure ntyft format with recursion. I show that \( \equiv \) is a bisimulation on \( T(\Sigma) \)—since \( \equiv \) is also symmetric, this yields the required result.
Thus I need to show that, for $p, q \in T(\Sigma)$ and $a \in A$,

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

To this end it suffices to establish, for all ordinals $\lambda$, that

4. if $p \equiv q$ and $P \vdash p \xrightarrow{a,\lambda} p'$, then there is a $q'$ with $P \vdash q \xrightarrow{a,\lambda} q'$ and $p' \equiv q'$.

The desired result is then obtained by taking $\lambda$ to be the closure ordinal $\kappa$ of Def. 4. This I will do by induction on $\lambda$, at the same time establishing that

3. if $p \equiv q$ and $P \vdash p \xrightarrow{\alpha, \lambda}$, then $P \vdash q \xrightarrow{\alpha, \lambda}$,

1. if $p \equiv q$ and $P \vdash q \xrightarrow{\alpha}$, then $P \vdash p \xrightarrow{\alpha}$.

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

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

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

- Let $p = \mathit{f}(p_1, \ldots, p_n)$, then $\mathit{g}(\mathit{f}(q_1, \ldots, q_m))$ where $p_i \equiv q_i$ for $i = 1, \ldots, n$. Let $\pi$ be a proof of $P \vdash q \xrightarrow{\alpha} q'$ from $P$. By Defs. 4 and 13 there must be a pure ntyf rule $r \vdash_{i(x_1, \ldots, x_n)}\mathit{f}(t(x_1, \ldots, x_n))$ in $R$ and a closed substitution $\nu$ with $\nu(x_i) = q_i$ for $i = 1, \ldots, n$ and $t[\nu] = q'$, such that for each $(t_y \xrightarrow{\alpha} y) \in H$ the transition $P \vdash t_\nu[y] \xrightarrow{\alpha} \nu(y)$ is provable from $P$ by means of a strict subproof of $\pi$, and $P \vdash u[\nu] \xrightarrow{\alpha} u[y]$ for each $(u[y]) \in H$. Next, I define a substitution $\sigma : \mathit{var}(r) \to T(\Sigma)$ such that

(i) $\sigma(x_i) = p_i$ for $i = 1, \ldots, n$,

(ii) $\sigma(y) \equiv \nu(y)$ for each $y \in \mathit{var}(r)$,

(iii) $P \vdash t_\sigma[\sigma] \xrightarrow{\alpha, \lambda} \sigma(y)$ for each $(t_y \xrightarrow{\alpha} y) \in H$.

The definition of $\sigma$ and the inference of (i)–(iii) above provide with proceed induction on the distance $y \in \mathit{var}(r)$ 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), $\sigma(x_i) \equiv \nu(x_i)$ for $i = 1, \ldots, n$.

**Induction step:** When defining $\sigma(y)$ for some $y \in \mathit{var}$ with $(t_y \xrightarrow{\alpha} y) \in H$, by induction $\sigma(x)$ has been defined already for all $x \in \mathit{var}(t_y)$, so I may assume that $\sigma(x) \equiv \nu(x)$ for all $x \in \mathit{var}(t(y))$ and hence $t_\sigma[\sigma] \equiv t_\nu[\nu]$.

By induction on $\pi$, there is a $p_y$ with $\pi \vdash t_\sigma[\sigma] \xrightarrow{\alpha, \lambda} p_y$ and $p_y \equiv \nu(y)$. Define $\sigma(y) := p_y$. Properties (ii) and (iii) now hold for $y$.

Take $p' := t[\sigma]$. So $p' = t[\sigma] \equiv \nu[\nu] = q'$ by Property (ii) of $\sigma$. For each premise $(u \xrightarrow{\alpha} y) \in H$ one has $u[\sigma] \equiv u[\nu]$ by Property (ii) of $\sigma$. So $P \vdash u[\sigma] \xrightarrow{\alpha} \lambda \chi$ by Claim 1. By Defs. 4 and 13 together with Property (iii) of $\sigma$, this implies $P \vdash p = \mathit{f}(p_1, \ldots, p_n) \xrightarrow{\alpha} \lambda t[\sigma] = p'$.

Let $p = \langle X | S \rangle$. Then $q = \langle \alpha(X) | S'[\alpha] \rangle$ for some recursive specification $S' : V_S \to T(\Sigma)$ with $S_Y \equiv S'_Y$ for all $Y \in V_S$, and an injective substitution $\alpha : V_S \to \mathit{Var}$ such that the range of $\alpha$ contains no variables occurring free in $\langle S'_X | S \rangle$ for some $Y \in V_S$. Now $\langle S_X | S \rangle \equiv \langle S'_X | S'[\alpha] \rangle$. Let $\pi$ be a proof of $P \vdash q \xrightarrow{\alpha} q'$ from $P$. By Defs. 4 and 13 $P \vdash (S'_X | S'[\alpha]) \xrightarrow{\alpha} q'$ is provable from $P$ by means of a strict subproof of $\pi$. So by induction there is a $p'$ such that $P \vdash (S_X | S) \xrightarrow{\alpha} p'$ and $p' \equiv q'$. By Defs. 4 and 13 $P \vdash p = \langle X | S \rangle \xrightarrow{\alpha} \lambda p'$. This establishes Claim 2.

Next, suppose that $p \equiv q$ and $P \vdash p \xrightarrow{\alpha} \lambda$. By Def. 3 there is no $p' \in T(\Sigma)$ with $P \vdash p \xrightarrow{\alpha} p'$. Using Claim 2, there is no $q' \in T(\Sigma)$ with $P \vdash q \xrightarrow{\alpha} q'$. By Remark 2 $P \vdash q \xrightarrow{\alpha}$. This yields Claim 3.

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

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 ntyf format with recursion. Let $\mathcal{R}$ be the smallest lean precongruence containing bisimilarity, i.e., $\mathcal{R} \subseteq T(\Sigma) \times T(\Sigma)$ is the smallest relation on processes satisfying

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

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

As $(\mathit{Var}[S] | \rho) : V_S \to T(\Sigma)$ and $(\mathit{Var}[S] | \nu) : V_S \to T(\Sigma)$, this implies that in the last clause one even has $(\mathit{Var}[S] | \rho) \mathcal{R} (\mathit{Var}[S] | \nu)$ for all terms $t \in T(\Sigma)$.

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

- if $p \mathcal{R} q$ and $P \vdash p \xrightarrow{a} p'$, then there is a $q'$ with $P \vdash q \xrightarrow{a} q'$ and $p' \mathcal{R} q'$,
- if $p \mathcal{R} q$ and $P \vdash q \xrightarrow{a} q'$, then there is a $p'$ with $P \vdash p \xrightarrow{a} p'$ and $p' \mathcal{R} q'$.
To this end it suffices to establish, for all ordinals $\lambda$, that
4. if $p \vdash q \lambda$ and $P \vdash p \lambda \rightarrow q \lambda'$, then there is a $q'$ with $P \vdash q \lambda \rightarrow q \lambda'$ and $P \vdash q \rightarrow q'$,
2. if $p \vdash q \rightarrow q'$, then there is a $p'$ with $P \vdash p \lambda \rightarrow p' \lambda$ and $P \vdash q \rightarrow q'$.

The desired result is then obtained by taking $\lambda$ to be the closure ordinal $\kappa$ of Def. 4. This I will do by induction on $\lambda$, at the same time establishing that
3. if $p \vdash q \lambda$ and $P \vdash p \lambda \rightarrow q \lambda$, then $P \vdash q \lambda \rightarrow q'$.
1. if $p \vdash q \rightarrow q'$, then $P \vdash q \lambda \rightarrow q'$.

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

Suppose $p \vdash q \rightarrow q'$ and $P \vdash p \lambda \rightarrow q'$. By Remark 3 there is no $q'' \in T(\Sigma)$ with $P \vdash q \lambda \rightarrow q''$. So by induction, using Claim 4 above, there is no $q'' \in T(\Sigma)$ with $P \vdash p \lambda \rightarrow q''$ for some $\lambda < \kappa$. By Def. 5 $P \vdash p \lambda \rightarrow q'$. This yields Claim 1.

Now suppose $p \vdash q \rightarrow q'$ and $P \vdash p \lambda \rightarrow q'$. I need to find a $p'$ with $P \vdash p \lambda \rightarrow q'$ and $P \vdash q \lambda \rightarrow q'$. This I will do by structural induction on the proof $p \rightarrow q'$ from $P$. I make a case distinction based on the derivation of $p \vdash q'$.

- Let $p \equiv B q$. Using that $\equiv B$ is a bisimulation, there must be a process $p'$ such that $P \vdash p \lambda \rightarrow p'$ and $p' \equiv B q'$, hence $p' \vdash q'$. Since $P \vdash p \lambda \rightarrow p'$, certainly $P \vdash p \lambda \rightarrow q'$, by Remark 1.

- Let $p = f(p_1, \ldots, p_n)$ and $q = f(q_1, \ldots, q_n)$ where $p_i \vdash q_i$ for $i = 1, \ldots, n$. Let $p$ be a proof of $p \rightarrow q'$ from $P$. By Defs. 4 and 5 there must be a pure ntyf rule $r = \frac{f(p_1, \ldots, p_n)}{H}$ in $R$ and a closed substitution $\nu$ with $\nu(x_i) = q_i$ for $i = 1, \ldots, n$ and $t[\nu] = q'$, such that for each $t_g \leadsto y \in H$ the transition $t_g[\nu] \leadsto \nu(y)$ is provable from $P$ by means of a strict subproof of $p$, and $P \vdash \nu(y) \rightarrow q'$. For each $(u \leftarrow y)$ in $H$, Next, I define a substitution $\sigma: \nu \rightarrow \nu$ in $T(\Sigma)$ such that

(i) $\sigma(x) = p_i$ for $i = 1, \ldots, n$,
(ii) $\sigma(y) = \nu(y)$ for each $y \in \nu(r)$,
(iii) $P \vdash t_g[\sigma] \leadsto \sigma(y)$ for each $(t_g \leftarrow y) \in H$.

The definition of $\sigma(y)$ and the inference of (i)–(iii) above proceed with induction on the distance of $y \in \nu(r)$ 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), $\sigma(x_i) \vdash \nu(x)$ for $i = 1, \ldots, n$.

**Induction step:** When defining $\sigma(y)$ for some $y \in \nu(r)$ with $(t_g \leftarrow y) \in H$, by induction $\sigma(x)$ has been defined already for all $x \in \nu(r)$, so I may assume that $\sigma(x) \vdash \nu(x)$ for all $x \in \nu(r)$ and hence $t_g[\sigma] \vdash t_g[\nu]$ by (ii). By induction on $\pi$, there is a $p_\pi$ with $P \vdash t_g[\sigma] \leadsto \lambda p_\pi$ and $p_\pi \vdash \nu(y)$. Define $\sigma(y) := p_\pi$. Properties (ii) and (iii) now hold for $y$.

Take $p' := t[\sigma]$. So $p' = t[\sigma] \vdash \nu(y) \rightarrow q'$ by (ii) and Property (ii) of $\sigma$. For each premise $(u \leftarrow y) \in H$ one has $u[\sigma] \vdash u[\nu]$ by (ii) and Property (ii) of $\sigma$. So $P \vdash u[\sigma] \rightarrow \lambda u[\nu]$ by Claim 1. By Defs. 4 and 13 together with Property (ii) of $\sigma$, this implies $P \vdash p = f(p_1, \ldots, p_n) \rightarrow \lambda t[\sigma] = p'$.

Let $p := \langle Z[S]\rangle[\sigma] = \langle Z[S]\rangle[\nu]$ and $q := \langle Z[S]\rangle[\sigma] = \langle Z[S]\rangle[\nu]$ where $S : V_S \rightarrow T(\Sigma)$ with $Z \in V_S \subseteq Var$, $\rho, \sigma : Var \vdash V_S \rightarrow T(\Sigma)$, and for all $x \in Var \ \backslash V_S$ one has $\rho(x) \vdash \sigma(x)$. Let $\pi$ be a proof of $q \rightarrow q'$ from $P$. By Defs. 3 and 13 $P \vdash p = \langle Z[S]\rangle[\sigma] \rightarrow \lambda p'$ and $P \vdash q \rightarrow q'$. Using Claim 2, there is no $p'' \in T(\Sigma)$ with $P \vdash p \lambda \rightarrow q'$. This yields Claim 3.

Finally, suppose $p \vdash q \rightarrow q'$ and $P \vdash p \lambda \rightarrow q'$. I need to find a $q'$ with $P \vdash q \rightarrow q'$ and $P \vdash q \lambda \rightarrow q'$. This I will do by structural induction on the proof $p \rightarrow q'$ from $P$. I make a case distinction based on the derivation of $p \vdash q'$.

- Let $p \equiv B q$. Since $P \vdash p \lambda \rightarrow q'$, certainly $P \vdash p \lambda \rightarrow q'$. By Remark 1. Using that $\equiv B$ is a bisimulation, there must be a process $q'$ such that $P \vdash q \lambda \rightarrow q'$ and $P \vdash \lambda q'$, hence $P \vdash q'$. Let $p = f(p_1, \ldots, p_n)$ and $q = f(q_1, \ldots, q_n)$ where $p_i \vdash q_i$ for $i = 1, \ldots, n$. Let $\pi$ be a proof of $p \lambda \rightarrow q'$. By Defs. 4 and 5 there must be a pure ntyf rule $r = \frac{f(p_1, \ldots, p_n)}{H}$ in $R$ and a closed substitution $\sigma$ with $\sigma(x_i) = p_i$ for $i = 1, \ldots, n$ and $t[\sigma] = p'$, such that for each $(t_g \leftarrow y) \in H$ the transition $t_g[\sigma] \rightarrow \sigma(y)$ is provable from $P$ by means of a strict subproof of $p$, and $P \vdash u[\sigma] \rightarrow \lambda u[\sigma]$.

The definition of $\nu(y)$ and the inference of (ii)–(iii) above proceed with induction on the distance of $y \in \nu(r)$ 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), $\nu(x_i) \vdash \nu(x_i)$ for $i = 1, \ldots, n$.

**Induction step:** When defining $\nu(y)$ for some $y \in \nu(r)$ with $(t_g \leftarrow y) \in H$, by induction $\nu(x)$ has been defined already for all $x \in \nu(r)$, so I may assume that $\sigma(x) \vdash \nu(x)$ for all $x \in \nu(r)$ and hence $t_g[\sigma] \rightarrow t_g[\nu]$ by (ii). By induction on $\pi$, there is a $q_\pi$ with $P \vdash t_g[\sigma] \rightarrow \pi q_\pi$ and $P \vdash \nu(y)$. Define $\nu(y) := q_\pi$. Properties (ii) and (iii) now hold for $y$.

Take $q' := t[\sigma]$. So $p' = t[\sigma] \vdash \nu(y) \rightarrow q'$ by (ii) and Property (ii) of $\nu$. For each premise $(u \leftarrow y) \in H$ one has $u[\sigma] \vdash u[\nu]$ by (ii) and Property (ii) of $\nu$. So $P \vdash u[\nu] \rightarrow \lambda u[\nu]$ by Claim 3. Since $CT^+$ is closed under deduction, together with Property (iii) of $\nu$ this implies $P \vdash q = f(q_1, \ldots, q_n) \rightarrow \lambda t[\sigma] = q'$.
Let \( \pi \) be a proof of \( p \xrightarrow{\lambda} p' \) from \( P \). By Definitions 4 and 13, \( S_p \) is a TSS in the tyft/tyxt format with recursion. So by induction there is a \( q' \) such that \( P \vdash \langle S_p \rangle \nu \xrightarrow{\rho} q' \) and \( p' \sim R q' \). By Definitions 4 and 13, \( P \vdash q = \langle T(\Sigma) \rangle \nu \xrightarrow{\rho} q' \).

This yields Claim 4.

The above result implies that any nyft/ntyxt language with recursion satisfies confluence requirement (1) up to \( \equiv_B \), 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 \( \sim \lambda \) and \( \sim \mu \) for ordinals \( \lambda \) and \( \mu \) all coincide, and \( \equiv_B = \equiv_B \) is true.

The following auxiliary concept was used in Section 4 to show that CCS satisfies Condition (2) of Def. 11.

Definition 15 A symmetric relation \( \mathcal{R} \subseteq T(\Sigma) \times T(\Sigma) \) is a bisimulation up to \( \sim \) if \( \mathcal{R} \subseteq \equiv_B \). Using symmetry and transitivity of \( \equiv_B \) this is straightforward.

Definition 15 Proposition 4 If \( \mathcal{R} \subseteq \equiv_B \), then \( \mathcal{R} \subseteq \equiv_B \).

Proof: Using the reflexivity of \( \equiv_B \) it suffices to show that \( \equiv_B \mathcal{R} \equiv_B \subseteq \equiv_B \). 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/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 tyft format with recursion. Let \( S, S' : T(\Sigma, W) \) be recursive specifications with \( S|_{\sigma} \equiv_B S'|_{\sigma} \) for all \( \sigma \in W \) and \( \sigma : W \rightarrow T(\Sigma) \).

I need to show that \( \langle X|S \rangle \equiv_B \langle X|S' \rangle \) for all \( X \in W \). Let \( \mathcal{R} \subseteq T(\Sigma) \times T(\Sigma) \) be the smallest relation on \( \{\bar{x}, \bar{\nu}\} \) satisfying

1. \( \langle X|S \rangle \mathcal{R} \langle X|S' \rangle \) and \( \langle X|S' \rangle \mathcal{R} \langle X|S \rangle \) for all \( X \in W \),
2. if \( (f, n) \in \Sigma \) and \( p_i \mathcal{R} q_i \) for all \( i = 1, \ldots, n \), then \( f(p_1, \ldots, p_n) \mathcal{R} f(q_1, \ldots, q_n) \), and
3. if \( S'' : V_{S''} \rightarrow T(\Sigma) \) with \( Z \subseteq V_{S''} \subseteq V_{S''} \) and \( \rho, \nu : \mathcal{R} \mathcal{V}_{S''} \rightarrow T(\Sigma) \) satisfy \( \rho(x) \mathcal{R} \mathcal{V}_{S''} \) for all \( x \in V_{S''} \), then \( Z(S''|\rho) \mathcal{R} Z(S''|\nu) \).

A trivial structural induction on \( T(\Sigma) \), using the last two clauses, shows that if \( \rho, \nu : \mathcal{V}_{S''} \rightarrow T(\Sigma) \) satisfy \( \rho(x) \mathcal{R} \nu(x) \) for all \( x \in Var \), then \( t[p]\mathcal{R}t[\nu] \). (\#)

So in the first clause one even has \( \langle t|S \rangle \mathcal{R} \langle t|S' \rangle \) for all \( t \in T(\Sigma) \), and in the last clause \( \langle t|S''|\rho \rangle \mathcal{R} \langle t|S''|\nu \rangle \) for all \( t \in T(\Sigma) \).

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, if \( p \mathcal{R} q \) and \( P \vdash p \xrightarrow{\sigma} p' \), then there is a \( q' \) with \( P \vdash q \xrightarrow{\sigma} q' \) and \( p' \mathcal{R} \equiv_B q' \), for all \( p, q \in T(\Sigma) \) and \( a \in A \). This I will do by structural induction on the proof \( \pi \) of \( p \xrightarrow{\sigma} p' \) from \( P \). I make a case distinction based on the derivations of \( p \mathcal{R} q \).

Let \( p = \langle X|S \rangle \) and \( q = \langle X|S' \rangle \) with \( X \in W \). Let \( \pi \) be a proof of \( p \xrightarrow{\sigma} p' \) from \( P \). By Definitions 4 and 13, \( S(X|S) \mathcal{R} \langle S(X|S') \rangle \mathcal{R} \langle S(X|S') \rangle \mathcal{R} \). So by induction there is a \( r' \) such that \( P \vdash \langle S(X|S') \rangle \mathcal{R} r' \) and \( p' \mathcal{R} \equiv_B r' \). Since \( \equiv_B \mathcal{R} \equiv_B \), this is straightforward. By Definitions 4 and 13, \( P \vdash q = \langle X|S' \rangle \mathcal{R} q' \).

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 \( \langle \_|S \rangle \).

The remaining two cases proceed in the same way as in the proof of Claim 4 for Thm. 2 but suppressing \( \lambda \) and with \( \mathcal{R} \subseteq \equiv_B \) 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 \( \nu = f(q_1, \ldots, q_n) \) and \( \varphi = \varphi(q_1, \ldots, q_n) \) such that for each \( (t_q \xrightarrow{\sigma} y) \in H \) the transition \( t_q \mathcal{R} \varphi(y) \) is provable from \( P \) by means of a strict subproof of \( \pi \). Next, I define a substitution \( \nu : \mathcal{V} \rightarrow T(\Sigma) \) such that

1. \( \nu(x_i) = q_i \) for \( i = 1, \ldots, n \),
2. \( \varphi \mathcal{R} \equiv_B \mathcal{V} \nu(y) \) for each \( y \in \mathcal{V} \).
3. \( P \vdash t_q \mathcal{V} \nu(y) \) for each \( (t_q \xrightarrow{\sigma} y) \in H \).

The definition of \( \mathcal{V} \nu(y) \) and the inference of (iii) above proceed with induction on the distance of \( y \in \mathcal{V} \) 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), \( \mathcal{V} \nu(x_i) \) for \( i = 1, \ldots, n \).

Induction step: When defining \( \mathcal{V} \nu(y) \) for some \( y \in \mathcal{V} \) with \( (t_q \xrightarrow{\sigma} y) \in H \), by induction \( \mathcal{V} \nu(x) \) has been defined already for all \( x \in \mathcal{V} \), so I may assume that \( \mathcal{V} \nu(x) \mathcal{R} \equiv_B \mathcal{V} \nu(x) \) for all \( x \in \mathcal{V} \). Now \( \mathcal{V} \nu(y) \equiv_B \mathcal{V} \nu(y) \) by (iii) and \( \mathcal{V} \nu(y) \mathcal{R} \equiv_B \mathcal{V} \nu(y) \). By the definition of bisimilarity, there

This proof shows that in the full congruence property 4 one only needs to assume \( S_Y \mathcal{R} \equiv_B S_Y \mathcal{R} \) for two specific substitutions \( \sigma \): namely \( \sigma(Y) = \langle Y|S \rangle \), resp. \( \langle Y|S \rangle \).

9
is a $q_y$ with $P \vdash t[y]\xrightarrow{\nu} q_y$ 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 $p' = t[\nu] \equiv R \equiv t[\nu] = q'$ by (*), Property (ii) of $\nu$, and Thm. 2. By Defs. 4 and 13 together with Property (iii) of $\nu$, this implies $P \vdash q = f(q_1, \ldots, q_n) \xrightarrow{\nu} t[\nu] = q'$.  

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 $R \sqsubseteq B$ or $B \sqsubseteq R$ for the red $R$ 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 $P \vdash \langle X | S' \rangle \xrightarrow{a} q'$ infer $P \vdash \langle X | S' \rangle \xrightarrow{a \lambda} r'$ by bisimilarity, and then infer $P \vdash \langle X | S' \rangle \xrightarrow{a \lambda} p'$ by induction. However, one may not use induction, as the transition $\langle X | S' \rangle \xrightarrow{a \lambda} r'$ may be derived later than $\langle X | S' \rangle \xrightarrow{a} q'$. 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 $\langle \mathcal{S} \rangle$ (cf. Footnote 2): it is trivial to find a counterexample in the GSOS format with unguarded recursion.

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

It is trivial to find a counterexample in the GSOS format with unguarded recursion.


