Congruence for Structural Congruences

MohammadReza Mousavi and Michel Reniers

Department of Computer Science,
Eindhoven University of Technology,
NL-5600MB Eindhoven, The Netherlands

Abstract. Structural congruences have been used to define the semantics and to capture inherent properties of language constructs. They have been used as an addendum to transition system specifications in Plotkin's style of Structural Operational Semantics (SOS). However, there has been little theoretical work on establishing a formal link between these two semantic specification frameworks. In this paper, we give an interpretation of structural congruences inside the transition system specification framework. This way, we extend a number of well-behavedness meta-theorems for SOS (such as well-definedness of the semantics and congruence of bisimilarity) to the extended setting with structural congruences.

1 Introduction

Structural congruences were introduced in [12,13] in the operational semantics specification of the π-calculus. There, structural congruences are a set of equations defining an equality and congruence relation on process terms. These equations are used as an addendum to the transition system specification, in the Structural Operational Semantics (SOS) style of [17]. The two specifications (structural congruences and SOS) are linked using a deduction rule dedicated to the behavior of congruent terms, stating that if a process term can perform a transition, all congruent process terms can mimic the same behavior.

The combination of structural congruences and SOS rules may simplify SOS specifications and make them look more compact. They can also capture inherent (so-called *spatial*) properties of composition operators (e.g., commutativity, associativity and zero element). Perhaps, the latter has been the main reason for using them in combination with SOS. However, as we argue in this paper, the interaction between the two specification styles is not as trivial as it seems. Particularly, well-definedness and well-behavedness meta-theorems for SOS such as those mentioned in [1] do not carry over trivially to this mixed setting. As an interesting example, we show that the addition of structural congruences to a set of safe SOS rules (e.g., tyft rules of [8]) can put the congruence property of bisimilarity in jeopardy. This result shows that a standard congruence format cannot be used, as is, for the combination of structural congruences and SOS rules. As another example, we show that well-definedness criteria defined by [5,6] for SOS with negative premises do not necessarily hold in the setting with structural congruences.

Three solutions can be proposed to deal with the aforementioned problems. The first is to avoid using structural congruences and use "pure" SOS specifications for defining operational semantics. In this approach, there is a conceptual distinction between the transition system semantics (as the model of the algebra) and the equational theory (cf. [2], for example). This way, one may lose the compactness and the intuitive presentation of the operational semantics, but in return, one will be able to benefit from the existing theories of SOS. This solution can be recommended as a homogenous way of specifying semantics. The second solution is to use structural congruences in combination with SOS rules and prove the well-behavedness theorems (e.g., well-definedness of the semantics and congruence of the notion of equality) manually. By taking this solution, all the tedious proofs of congruence, as a typical example, have to be done manually and re-done or adapted in the case of any single change in the syntax and semantics. Although this solution is a common practice, it does not seem very promising. The third solution is to extend meta-theorems of SOS to this mixed setting. In this paper, we pursue the third solution.

The rest of this paper is structured as follows. By reviewing the related work in Section 2, we position our work within the body of research in formal semantics. Then, in Section 3, we present basic definitions about transition system specifications, bisimilarity and congruence. Subsequently, Section 4 is devoted to accommodating structural congruences in the SOS framework. In Section 5, we study structural congruences from the congruence point of view. There, we propose a syntactic format for structural congruences that induces congruence for strong bisimilarity, if they are accompanied by a set of safe SOS rules. We show, by several abstract counter-examples, that our syntactic format cannot be relaxed in any obvious way and dropping any of the syntactic restrictions may destroy the congruence property in general. In Section 6, we extend our format to allow for SOS rules with negative premises and set the respective well-definedness criteria. To illustrate our congruence format with a concrete example, in Section 7, we apply it to a CCS-like process algebra. Finally, Section 8 concludes the paper and points out possible extensions of our work. For the sake of brevity and due to space restrictions, we omit the proofs. A detailed version of this paper (containing additional results) with proofs can be consulted in [15].

2 Related Work

Structural congruences find their origin in the chemical models of computation [3]. The Chemical Abstract Machine (Cham) of [4] is among the early instances of such models. In Cham, parallel agents are modelled by molecules floating around in a chemical solution. The solution is constantly stirred using a *magical mechanism*, in the spirit of the *Brownian motion* in chemistry, that allows for possible contacts among reacting molecules.

Inspired by the magical mechanism of Cham, structural congruences were introduced in [12,13] in the semantic specification of the π-calculus and since then, the practice of using structural congruences for the specification of operational semantics has continued. As stated in [14], structural congruences were also inspired by a curious difference between lambda-calculi and process calculi; in lambda-calculi, interacting terms are always placed adjacently in the syntax, while in process calculi, interacting agents may be dispersed around the process term due to syntactic restrictions. Thus, part of the idea is to bring interacting terms together by considering terms modulo structural changes. However, the application of structural congruences is not restricted to this concept. Structural congruence have also been used to define the semantics of new operators in terms of previously defined ones (e.g., defining the semantics of the parallel replication operator in terms of parallel composition in [12,13] and Section 7 of this paper).

There have been a number of recent works devoted to the fundamental study of formal semantics with structural congruences. Among these, we can refer to [10,18,19]. Lack of a congruent notion of bisimilarity for the semantics of the π-calculus has been known since [12] (which is not only due to structural congruences), but most attempts (e.g., [10,14,18,19]) were focused on deriving a suitable transition system (e.g., *contexts as labels* approach of [10]) or a notion of equivalence (e.g., barbed congruence of [14]) that induces congruence. The works of [10,18,19] deviate from the traditional interpretation of SOS deduction rules and establish a new semantic framework close to the reduction (reaction) rules of lambda calculus [9]. In [19], it is emphasized that the relation between this framework and the known congruence results for SOS remains to be established and the present paper realizes this goal (at least partially). Our results to date do not apply to SOS containing variable binding or name passing operators. Also, the kind of structural congruences that can be dealt with is quite limited; it is for example, not possible to consider structural congruences expressing properties such as associativity and zero elements.

To conclude, compared to the above approaches, we take a different angle to the problem, that is, to characterize the set of specifications that induce a reasonable transition relation in its commonly accepted meaning. In other words, we extend the notion of *structured operational semantics* [8,7] to cater for structural congruences. In particular, we extend the meta-theorems concerning congruence of bisimilarity [1,8] and well-definedness of the induced transition relation [5–7] to the setting with structural congruences.

3 Preliminaries

We assume that the set of process terms, denoted by $T(\Sigma)$ with typical members $t, t', t_0, \ldots$, is inductively defined on a set of variables $V = \{x, y, \ldots\}$ and a signature Σ. The signature contains a number of function symbols (composition operators: $f, g, \ldots$) with fixed arities ($ar(f), ar(g), \ldots$). Function symbols with arity 0 are called constants and are typically denoted by $a, b, \ldots$. Closed terms, denoted by $C(\Sigma)$ with typical members $p, q, p_0, \ldots$, are terms that do not contain variables. A substitution σ replaces variables in a term with other terms. The set of variables appearing in term t is denoted by $vars(t)$. A transition system specification, defined below, is a logical way of defining a transition relation on (closed) terms.

Definition 1 (Transition System Specification (TSS)) A *transition system specification* is a tuple (Σ, L, D) where Σ is a signature, L is a set of labels (with typical members $l, l', l_0, \ldots$) and D is a set of deduction rules. For all $l \in L$, and $s, s' \in T(\Sigma)$ we define that $(t, l, t') \in \to$ is a formula. A deduction rule $dr \in D$, is defined as a tuple (H, c) where H is a set of formulae and c is a formula. The formula c is called the *conclusion* and the formulae from H are called *premises*. A rule with an empty set of premises is called an axiom.

The notion of closed and the concept of substitution are lifted to formulae in the natural way. A formula $(t, l, t') \in \to$ is denoted by the more intuitive notation $t \xrightarrow{l} t'$, as well. We refer to t as the source and to t' as the target of the transition. A deduction rule (H, c) is denoted by $\frac{H}{c}$ in the remainder.

In the traditional setting, the transition relation induced by a transition system specification is the smallest set of provable closed formulae using a well-founded proof tree based on the deduction rules. Note that for more complicated transition systems specifications such a unique transition relation may not exist (see [1,6] and Section 6 of the present paper for more details). Next, we define our notion of equality, namely, the notions of strong bisimulation and bisimilarity.

Definition 2 (Bisimulation and Bisimilarity [16]) A relation $R \subseteq C(\Sigma) \times C(\Sigma)$ is a *simulation* relation with respect to a transition relation $\to \subseteq C(\Sigma) \times L \times C(\Sigma)$ if and only if $\forall_{p,q \in C(\Sigma)}\ (p, q) \in R \Rightarrow \forall_{l \in L}\ \forall_{p' \in C(\Sigma)}\ p \xrightarrow{l} p' \Rightarrow \exists_{q' \in C(\Sigma)}\ q \xrightarrow{l} q'$ $\wedge (p', q') \in R$. A *bisimulation* relation is a symmetric simulation relation. Closed terms p and q are bisimilar with respect to $\to$ if and only if there exists a bisimulation relation R with respect to $\to$ such that $(p, q) \in R$. Two closed terms p and q are *bisimilar* with respect to a transition system specification tss, if and only if they are bisimilar with respect to the transition relation induced by tss. Note that bisimilarity (with respect to a transition relation or TSS) is an equivalence relation on closed terms.

Next, we define the concept of congruence which is of central importance to our topic.

Definition 3 (Congruence) A relation $R \subseteq T(\Sigma) \times T(\Sigma)$ is a congruent relation with respect to a function symbol $f \in \Sigma$ if and only if for all terms $p_i, q_i \in T(\Sigma)$ $(0 \leq i < ar(f))$, if $(p_i, q_i) \in R$ (for all $0 \leq i < ar(f)$) then $(f(p_0, \ldots, p_{ar(f)-1}), f(q_0, \ldots, q_{ar(f)-1})) \in R$. Furthermore, R is called a *congruence* for a transition system specification if and only if it is a congruence with respect to all function symbols of the signature.

Bisimilarity is not in general a congruence. However, congruence is essential for the axiomatic treatment of bisimilarity. Furthermore, congruence of bisimilarity is of crucial importance in compositional reasoning. Several syntactic formats guaranteeing congruence for bisimilarity have been proposed (see [1] for an overview). Here, we choose the tyft format of [8] as a sufficiently general example of such formats for our purposes. Extensions to more general formats (such as the PANTH format of [20]) are discussed in Section 6.

Definition 4 (Tyft Format [8]) A rule is in tyft format if and only if it has the following shape:

$$\frac{\{t_i \xrightarrow{l_i} y_i | i \in I\}}{f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} t}$$

where x_i and y_i are all distinct variables (i.e., for all $i, i' \in I$ and $0 \leq j, j' < ar(f)$, $y_i \neq x_j$ and if $i \neq i'$ then $y_i \neq y_{i'}$ and if $j \neq j'$ then $x_j \neq x_{j'}$), f is a function symbol from the signature, I is a (possibly infinite) set of indices and t and t_i's are arbitrary terms. A transition system specification is in tyft format if and only if all its rules are.

Theorem 1 (Congruence for tyft [8]) For a TSS in tyft format, bisimilarity is a congruence.

4 Structural Congruences: An SOS Reading

Structural congruences sc on a signature Σ consist of a set of equations of the form $t \equiv t'$, where $t, t' \in T(\Sigma)$. They induce a structural congruence relation on closed terms, as defined below.

Definition 5 (Structural Congruence Relation) A structural congruence relation induced by structural congruences sc on signature Σ, denoted by $\equiv_{sc}$, is the minimal relation satisfying the following constraints:

1. $\forall_{p \in C(\Sigma)}\ p \equiv_{sc} p$ (reflexivity);
2. $\forall_{p,q \in C(\Sigma)}\ p \equiv_{sc} q \Rightarrow q \equiv_{sc} p$ (symmetry);
3. $\forall_{p,q,r \in C(\Sigma)}\ (p \equiv_{sc} q \wedge q \equiv_{sc} r) \Rightarrow p \equiv_{sc} r$ (transitivity);
4. $\forall_{f \in \Sigma} \forall_{p_i, q_i \in C(\Sigma)(0 \leq i < ar(f))}\ (\forall_{0 \leq i < ar(f)}\ p_i \equiv_{sc} q_i) \Rightarrow f(p_0, \ldots, p_{ar(f)-1}) \equiv_{sc} f(q_0, \ldots, q_{ar(f)-1})$ (congruence);
5. $\forall_{\sigma : V \to C(\Sigma)} \forall_{t,t' \in T(\Sigma)}\ (t \equiv t') \in sc \Rightarrow \sigma(t) \equiv_{sc} \sigma(t')$ (structural congruences).

In other words, $\equiv_{sc}$ is the smallest congruence satisfying $\equiv$ on closed terms.

In the remainder, we assume that the structural congruences have the same signature as the transition system specification they are added to. To link structural congruences to a transition system specification, a special rule is used, which we call *the structural congruence rule.*

Definition 6 (The Structural Congruence Rule [12]) The particular rule schema of the following form (which is in fact a set of deduction rules for all $l \in L$) is called the structural congruence rule:

$$\textbf{(struct)}\frac{x \equiv y \quad y \xrightarrow{l} y' \quad y' \equiv x'}{x \xrightarrow{l} x'}(l \in L)$$

Consider a transition system specification $tss = (\Sigma, L, D)$ and structural congruences sc on the same signature. Extension of tss with sc, denoted by $tss \cup \{\textbf{(struct)}\}$, is defined by the tuple $(\Sigma, L, D \cup \{\textbf{(struct)}\})$.

There remains a problem concerning Definition 6, namely, the structural congruence rule does not fit within the notion of a deduction rule as defined in Definition 1 since structural congruences (appearing in the premises) do not fit the definition of formulae per se. In other words, $x \equiv y$ is only a syntactic notation and has no meaning associated to it as yet. In this paper, we exploit the structural congruence relation to give a meaning to $x \equiv y$ by extending the notion of proof (see [15] for other possible interpretations). Syntactically, we allow for deduction rules of the form $\dfrac{\{\chi_i | i \in I\} \quad \{t_j \equiv t'_j | j \in J\}}{\chi}$ where χ and χ_i's are formulae as defined before (in Definition 1) and t_j and t'_j are terms from the signature. This rule format, easily accommodates the structural congruence rule. Then, we extend the notion of provable transitions to the following notion.

Definition 7 (Provable Transitions: Extended) A *proof* of a closed formula ϕ (in an extended transition system specification $tss \cup \{\textbf{(struct)}\}$) is a well-founded upwardly branching tree of which the nodes are labelled by closed formulae such that

– the root node is labelled by ϕ, and
– if ψ is the label of a node q and $\{\psi_i \mid i \in I\}$ is the set of labels of the nodes directly above q, then there is a deduction rule $\dfrac{\{\chi_i \mid i \in I\} \quad \{t_j \equiv t'_j | j \in J\}}{\chi}$ (in $tss \cup \{\textbf{(struct)}\}$) and a substitution σ such that $\sigma(\chi) = \psi$, for all $i \in I$, $\sigma(\chi_i) = \psi_i$, and for all $j \in J$, $\sigma(t_j) \equiv_{sc} \sigma(t'_j)$.

We re-use the same notations for provability of formulae in the extended setting.

We are not able to reproduce the results of Theorem 1 (concerning congruence for tyft format) in the extended setting with structural congruences. In fact, adding structural congruences to a set of tyft rules does not preserve the congruence property of bisimilarity. The following counter-example shows this fact.

Example 1 Consider the following structural congruence equation and transition system specification. The common signature is assumed to have a and b as constants and f as a unary operator.

$$a \equiv f(b) \qquad \textbf{(a)}\frac{}{a \xrightarrow{l_0} a} \qquad \textbf{(b)}\frac{}{b \xrightarrow{l_0} a}$$

In the above specification, both a and b can perform an l_0 transition to a due to rules **(a)** and **(b)**, respectively. On one hand, using Definition 5, a is only structurally congruent (by means of $\equiv_{sc}$) to itself and $f(b)$. On the other hand, b is only congruent to itself. Since $f(b)$ cannot perform any new transition, neither a nor b can perform any other transition due to **(struct)**. Thus, to this end, we have $a \leftrightarrow b$. However, it does not hold that $f(a) \leftrightarrow f(b)$ since $f(a)$ cannot perform any transition (it is only congruent to $f(f(b))$ which cannot perform any transition either), but $f(b)$ can perform an l_0 transition to a (using **(struct)** since it is congruent to a). This shows that bisimilarity is not a congruence in the above transition system specification, despite the fact that the original transition system specification is in tyft format.

Several other counter-examples of violating congruence property by structural congruences are presented in the remainder of this paper.

5 Well-Behaved Structural Congruences

In this section, we start with proposing a syntactic format for structural congruences and stating that structural congruences conforming to this format are safe for the purpose of congruence when added to a set of tyft rules. Then, in Section 5.2, by several counter-examples, we show that none of the syntactic constraints on this format can be dropped in general and thus our syntactic format cannot be relaxed trivially.

5.1 Congruence Format for Structural Congruences (cfsc)

Our syntactic criteria on structural congruences are defined below.

Definition 8 (Cfsc format) Structural congruences sc (added to a transition system specification tss) are in the cfsc format if and only if any equation in sc is of one of the following two forms.

1. An *fx equation* is of the form $f(x_0, \ldots, x_{ar(f)-1}) \equiv g(y_0, \ldots, y_{ar(g)-1})$ for function symbols f and g (which need not be different) and for variables x_i and y_j. Variables x_i and y_j are distinct among themselves (i.e., for all $i \neq j$, $x_i \neq x_j$ and $y_i \neq y_j$) but they need not form two disjoint sets (i.e., it may be that for some i and j, $x_i = y_j$).
2. A *defining equation* is of the form $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$ (or similarly, $t \equiv f(x_0, \ldots, x_{ar(f)-1})$) where f is a function symbol and t is an arbitrary term. Similar to fx equations, variables x_i have to be distinct. Two more conditions have to be satisfied for this type of equations; first, all variables in t should be bound by variables $x_0, \ldots, x_{ar(f)-1}$, i.e., $vars(t) \subseteq \{x_i | 0 \leq i < ar(f)\}$ and second, f may not appear in any other structural congruence equation and source of the conclusion of any deduction rule in tss. We have no further assumption about t, thus, there may be a repetition of variables in t, occurrences of f may appear in t and it may consist of any number of constants and function symbols.

Note that the above two categories are not disjoint; i.e., an equation may be both fx and defining. For the remainder, it does not make any difference whether such equations are taken as fx, defining, or both.

In the following theorem, we state that structural congruences conforming to the cfsc format, when added to a set of tyft rules, induce a congruent bisimilarity relation. The proof of the following theorem follows from Theorem 1 by transforming the transition system specification with structural congruences to a "pure" transitions system specification in tyft format that provably induces the same transition relation.

Theorem 2 (Congruence Theorem for cfsc) Consider a set of deduction rules tss in tyft format. If structural congruences sc (added to tss) are in the cfsc format, then bisimilarity with respect to $tss \cup \{\textbf{(struct)}\}$ is a congruence.

5.2 Impossible Relaxations of Cfsc

Next, we show that the cfsc format cannot be relaxed in any obvious way. We take every and each syntactic constraint on cfsc and by an abstract counter-example, show that removing it will result in violating the congruence. We start with a counter-example showing that variables in each side of the fx equation need to be distinct and that the variables in the $f(x_0, \ldots, x_{ar(f)})$ side of a defining equation need to be distinct.

Example 2 $\quad f(x,x) \equiv a \quad \textbf{(a)}\dfrac{}{a \xrightarrow{l_0} a} \quad \textbf{(b)}\dfrac{}{b \xrightarrow{l_0} a}$

Similar to Example 1, it clearly holds in the above specification that $a \underline{\leftrightarrow} b$. However, it does not hold that $f(a,a) \underline{\leftrightarrow} f(a,b)$ since the former can perform an l_0 transition, while the latter deadlocks. Thus, bisimilarity is not a congruence. Note that the above structural congruence can be considered both an fx equation and a defining equation.

The other condition on fx equations is that they may only have one function symbol in each side of the equation. We have already shown that this constraint cannot be relaxed in Example 1 in the previous section. There, the equation $a \equiv f(b)$ had two function symbols, namely the constant b and unary function symbol f and the congruence property is shown to be violated. A similar condition forces defining equations to have only one fresh function symbol on the side to be defined. In the following example, we show that allowing more fresh function symbols also endangers congruence.

Example 3 $\quad f(b) \equiv a \quad \textbf{(a)}\dfrac{}{a \xrightarrow{l_0} a}$

Suppose that our signature consists of three constants a, b and c and a unary function symbol f. Then, it immediately follows that $b \underline{\leftrightarrow} c$ since none of the two constants can perform any transition. However, it does not hold that $f(b) \underline{\leftrightarrow} f(c)$ since the first term can perform a transition while the latter deadlocks.

Another constraint on a defining equation $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$ is that $vars(t) \subseteq \{x_i | 0 \leq i < ar(f)\}$. The following counter-example shows that we cannot drop this constraint.

Example 4 $\quad d \equiv f(a,x) \quad \textbf{(c)}\dfrac{}{c \xrightarrow{l_0} c} \quad \textbf{(f)}\dfrac{x_1 \xrightarrow{l_0} y_1}{f(x_0,x_1) \xrightarrow{l_0} y_1}$

Suppose that the common signature consists of a, b, c and d as constants and f as a unary operator. Equation $d \equiv f(a,x)$ fits all syntactic criteria of a defining equation (for d), but the one stated above. It follows from **(f)** that $f(a,c) \xrightarrow{l_0} c$.

Since $d \equiv f(a,x)$, then $d \xrightarrow{l_0} c$ and from the same equation (in the other direction), we can deduce that $f(a,b) \xrightarrow{l_0} c$. However, it cannot be derived that $f(b,b) \xrightarrow{l_0} c$. This witnesses that bisimilarity is not a congruence, as $a \underline{\leftrightarrow} b$ but it does not hold that $f(a,b) \underline{\leftrightarrow} f(b,b)$.

The last constraint on defining equations is concerned with freshness of the function symbol being defined. In the following two counter-examples, we show that the defined function symbol cannot appear neither in any other structural congruence equation, nor in the source of the conclusion of a deduction rule.

Example 5 $\quad c \equiv a \quad c \equiv f(b) \quad \textbf{(a)}\dfrac{}{a \xrightarrow{l_0} a} \quad \textbf{(b)}\dfrac{}{b \xrightarrow{l_0} a}$

Again, in the above specification, we have $a \underline{\leftrightarrow} b$ but it is not true that $f(a) \underline{\leftrightarrow} f(b)$ since from the structural congruences, we can derive that $a \equiv_{sc} f(b)$ and hence $f(b)$ can perform an l_0 transition to a while $f(a)$ cannot perform any transition.

Example 6 $\quad f(x) \equiv g(a) \quad \textbf{(a)}\dfrac{}{a \xrightarrow{l_0} a} \quad \textbf{(b)}\dfrac{}{b \xrightarrow{l_0} a} \quad \textbf{(f)}\dfrac{}{f(x) \xrightarrow{l_0} f(x)}$

It follows from the above specification that $a \underline{\leftrightarrow} b$ but it does not hold that $g(a) \underline{\leftrightarrow} g(b)$ since the former can perform a transition due to **(struct)** and **(f)** while the latter cannot perform any transition.

6 Structural Congruences and Negative Premises

Transition system specifications are mainly used to specify transitions of process terms in terms of transitions of their subterms. Sometimes it comes handy to define a transition based on the impossibility of a transition for a particular subterm. Several instances of SOS semantics in the literature make use of this feature (e.g., for defining priority, deadlock detection, sequencing and urgency, cf. [1,5]). Thus, it seems natural to extend transition system specifications in tyft format to account for negative premises. The following definition realizes this goal.

Definition 9 (Ntyft Format [7]) A rule is in ntyft format if and only if it has the following shape.

$$\textbf{(r)}\frac{\{t_i \xrightarrow{l_i} y_i | i \in I\} \quad \{t_j \overset{l_j}{\nrightarrow} | j \in J\}}{f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} t}$$

The same conditions as of tyft format hold for the positive premises and the conclusion. There is no particular constraint on the terms appearing in the negative premises. Set J is the (possibly infinite) set of indices of negative premises.

However, in the presence of negative premises, the concepts of proof and provable transitions become more complicated. A proof, as defined before, can provide a reason for presence of a transition but not for its absence. Thus, we have to resort to another notion of proof that can account for absence of transitions, as well.

Here, we choose the notion of *stable model* of [5] as an intuitive model of the induced transition relation. The definition is slightly adapted to cater for structural congruences and to fit our notations and past definitions.

Definition 10 (Stable Model) A positive closed formula ϕ is provable from a set of positive formula T and a transition system specification tss, denoted by $(T, tss) \vdash \phi$, if and only if there is a well-founded upwardly branching tree of which the nodes are labelled by closed formulae such that

- the root node is labelled by ϕ, and
- if the label of a node q, denoted by ψ, is a positive formula and $\{\psi_i \mid i \in I\}$ is the set of labels of the nodes directly above q, then there is a deduction rule $\dfrac{\{\chi_i \mid i \in I\}\ \{t_j \equiv t'_j | j \in J\}}{\chi}$ in tss (N.B. χ_i can be a positive or a negative formula) and a substitution σ such that $\sigma(\chi) = \psi$, for all $i \in I$, $\sigma(\chi_i) = \psi_i$ and for all $j \in J$, $\sigma(t_j) \equiv_{sc} \sigma(t'_j)$;
- if the label of a node q, denoted by $p \overset{l}{\nrightarrow}$, is a negative formula then there exists no p' such that $p \xrightarrow{l} p' \in T$.

A stable model, also called a transition relation, defined by a transition system specification tss is a set of formulae T such that for all closed positive formulae ϕ, $\phi \in T$ if and only if $(T, tss) \vdash \phi$.

However, not all transition system specifications in ntyft format have a stable model and even if they have, it need not be unique. The following example shows simple instances of such phenomena.

Example 7 $\dfrac{a \overset{l_0}{\nrightarrow}}{a \xrightarrow{l_0} a}$ $\Bigg|$ $\dfrac{a \overset{l_0}{\nrightarrow}}{b \xrightarrow{l_0} b}$ $\dfrac{b \overset{l_0}{\nrightarrow}}{a \xrightarrow{l_0} a}$

Consider the above two transition system specifications, both defined on a signature with a and b as constants. The left-hand-side TSS has no stable model (as for any stable model $a \xrightarrow{l_0} a$ if and only if $a \overset{l_0}{\nrightarrow}$) and the right-hand-side one has two stable models, namely, $\{a \xrightarrow{l_0} a\}$, $\{b \xrightarrow{l_0} b\}$.

To solve this problem, in [5,7], an extra condition is imposed on transition system specifications in ntyft format. The following definition illustrates this condition.

Definition 11 (Stratification) A stratification of a transition system specification tss in the ntyft format is a function $\mathcal{S}$ from closed positive formulae to an ordinal such that for all deduction rules of tss in ntyft format (in the shape of rule **(r)** in Definition 9) and for all substitutions σ, $\forall_{i \in I} \mathcal{S}(\sigma(t_i \xrightarrow{l_i} y_i)) \leq \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} t'))$ and $\forall_{j \in J, t' \in T(\Sigma)}\ \mathcal{S}(\sigma(t_j \xrightarrow{l_j} t')) < \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} t'))$. A transition system specification is called *stratified* if and only if there exists a stratification function for it.

The following theorem from [5] formalizes the advantages of stratified transition system specifications.

Theorem 3 Consider a transition system specification tss in the ntyft format. If tss is stratified, then it has a unique stable model. Furthermore, bisimilarity is a congruence for the stable model of a stratified transition system specification.

Now we have enough ingredients to study the implications of negative premises on the structural congruences. Before doing so, we show that a naive treatment of structural congruences, i.e., neglecting them, may ruin the well-definedness of the induced transition relation.

Example 8 $\dfrac{a \overset{l_0}{\nrightarrow}}{b \xrightarrow{l_0} b}$ | $a \equiv b$

First, consider the transition system specification given in the left-hand-side (with a and b as constants). It is stratified by the function $\mathcal{S}$, if we define for all closed terms p, $\mathcal{S}(a \xrightarrow{l} p) \doteq 1$ and $\mathcal{S}(b \xrightarrow{l} p) \doteq 2$. Following Theorem 3, it defines the unique transition relation (its stable model), which is $\{b \xrightarrow{l_0} b\}$.

Then, suppose that we add the structural congruence in the right-hand-side (which is indeed in the cfsc format) to the specification. Suddenly, the associated transition system specification loses its well-definedness. The combination of the above deduction rule and $a \equiv b$ leads to a contradiction, namely $b \xrightarrow{l_0} b$ if and only if $a \overset{l_0}{\nrightarrow}$ and if $b \xrightarrow{l_0} b$ then $a \xrightarrow{l_0} b$.

To solve the above mentioned problem, we extend the notion of stratification to structural congruences as follows.

Definition 12 (Stratification: Extended) Consider a transition system specification tss in ntyft format and structural congruence in the cfsc format. Then, $tss \cup \{\textbf{(struct)}\}$ is stratified, if there exists a function $\mathcal{S}$ from closed formulae to an ordinal such that for all closed substitutions σ:

1. for all deduction rules in tss of the form $\dfrac{\{t_i \xrightarrow{l_i} y_i | i \in I\} \quad \{t_j \overset{l_j}{\nrightarrow} | j \in J\}}{f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} t}$, it holds that $\forall_{i \in I} \mathcal{S}(\sigma(t_i \xrightarrow{l_i} y_i)) \leq \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} t'))$ and $\forall_{j \in J, t' \in T(\Sigma)}$ $\mathcal{S}(\sigma(t_j \xrightarrow{l_j} t')) < \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} t'))$,
2. for all fx equations of the form $f(x_0, \ldots, x_{ar(f)-1}) \equiv g(x_0, \ldots, x_{ar(g)-1})$ in sc, it holds that $\forall_{l \in L, t \in T(\Sigma)}$ $\mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} t)) = \mathcal{S}(\sigma(g(x_0, \ldots, x_{ar(g)-1}) \xrightarrow{l} t))$,
3. for all defining equations of the form $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$ in sc, it holds that $\forall_{l \in L, t' \in T(\Sigma)}$ $\mathcal{S}(\sigma(t \xrightarrow{l} t')) \leq \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} t'))$.

Next, we extend the well-definedness theorem for the transition relation to the setting with structural congruences. The following theorem states that if a combination of a transition system specification and structural congruences is stratified, then it defines a unique transition relation.

Theorem 4 If the combination of transition system tss in ntyft format and $tss \cup \{(\textbf{struct})\}$ is stratified, then $tss \cup \{(\textbf{struct})\}$ has a unique stable model. Furthermore, for this model, bisimilarity is a congruence.

Possible extensions to the ntyft format are the addition of ntyxt rules and predicates. The ntyft-ntyxt format of [7] is a relaxation of ntyft format that allows for variables in the source of the conclusion. In [7], it is shown how to reduce the ntyft-ntyxt format to the ntyft format. Adding structural congruences to TSS's in the ntyft-ntyxt format, however, is not straightforward. The reduction of ntyft-ntyxt to ntyft requires to copy each ntyxt rule for every function symbol in the signature. This reduction thus disallows the presence of any defining equation, as the new deduction rules contain defined function symbols in the source of their conclusion. Thus, up to now, we can only guarantee congruence for a combination of structural congruences and a transition system specification with ntyxt rules if the structural congruences comprise of fx equations only. In [15], we give a solution to this problem by interpreting defining equations as conservative operational extensions to a transition system specification.

Predicates are other ingredients of transition system specifications that are used to specify concepts such as termination and divergence on process terms [20]. Unlike negative premises and ntyxt rules, addition of predicates to a transition system specification has no implication on structural congruences and the cfsc format. Predicates can be modelled as transitions with a dummy right-hand side (a dummy variable in the premises and a dummy constant in the conclusion). Thus, the results that we have proved so far easily extend to the PANTH format of [20] which allows for both ntyft-ntyxt rules and predicates.

7 Case Study

In this section, we quote an SOS semantics of CCS from [11] (with restriction of nondeterminism to finite sum and introduction of the parallel replication operator) and then introduce structural congruences, à la [12], conforming to our format. By doing this, we show how our format is able to capture a number of non-trivial structural congruences and make the presentation look more intuitive and compact. Moreover, from this specification one can still derive congruence for strong bisimilarity automatically.

The syntax of our CCS-like process algebra is given below.

$$P \quad ::= \quad 0 \mid \alpha.P \mid P + Q \mid P \mid\mid Q \mid P \setminus L \mid !P \mid A$$

In this syntax, constant 0 stands for the terminating process. The action prefix operator $\alpha.P$ (which is actually a class of unary operators parameterized by labels $\alpha \in \mathcal{L}$) shows α as its first step and proceeds with P. The set of labels $\mathcal{L}$

is partitioned into the set of names, typically denoted by l, and co-names, denoted by $\bar{l}$. By extending the same notation, let $\bar{\bar{l}}$ be defined as l. Restriction operator $P \setminus L$, parameterized by $L \subseteq \mathcal{L}$, defines the scope of local names (and co-names). Nondeterministic choice is denoted by $+$. Parallel composition is denoted by $P \mid\mid Q$. Parallel replication of process P is denoted by $!P$ which usually serves as a restricted substitute for recursion. Recursive symbols A serve as short-hands for their defining processes, denoted by $A \doteq P$ and are used to define processes hierarchically. We treat recursive symbols as constants in our signature.

The transition system specification defining the semantics of our language is given below. In this semantics, $l, \bar{l} \in \mathcal{L}$ and $\alpha \in \mathcal{L} \cup \{\tau\}$, where τ is the result of a communication ($\overline{\tau}$ is defined to be τ).

$$\textbf{(Act)}\frac{}{\alpha.x \xrightarrow{\alpha} x} \qquad \textbf{(Res)}\frac{x \xrightarrow{\alpha} y}{x \setminus L \xrightarrow{\alpha} y \setminus L}(\alpha, \overline{\alpha} \notin L) \qquad \textbf{(Con)}\frac{t \xrightarrow{\alpha} y}{A \xrightarrow{\alpha} y}(A \doteq t)$$

$$\textbf{(Sum0)}\frac{x_0 \xrightarrow{\alpha} y}{x_0 + x_1 \xrightarrow{\alpha} y} \qquad \textbf{(Sum1)}\frac{x_1 \xrightarrow{\alpha} y}{x_0 + x_1 \xrightarrow{\alpha} y} \qquad \textbf{(Rep)}\frac{x \mid\mid !x \xrightarrow{\alpha} y}{!x \xrightarrow{\alpha} y}$$

$$\textbf{(Com0)}\frac{x_0 \xrightarrow{\alpha} y_0}{x_0 \mid\mid x_1 \xrightarrow{\alpha} y_0 \mid\mid x_1} \quad \textbf{(Com1)}\frac{x_1 \xrightarrow{\alpha} y_1}{x_0 \mid\mid x_1 \xrightarrow{\alpha} x_0 \mid\mid y_1} \quad \textbf{(Com2)}\frac{x_0 \xrightarrow{l} y_0 \quad x_1 \xrightarrow{\bar{l}} y_1}{x_0 \mid\mid x_1 \xrightarrow{\tau} y_0 \mid\mid y_1}$$

In the above specification, rule **(Act)** defines that an action prefix operator can execute its first action and continue with the rest. Each rule in this specification should be considered as a rule schema, representing a possibly infinite number of rules for each $l \in \mathcal{L}$. Side conditions, in this particular case study, only govern presence and absence of such copies. Rule **(Res)** allows for performing actions beyond the restricted set L (i.e., blocks the rest). Rules **(Sum0)** and **(Sum1)** define the non-deterministic choice operator. Rules **(Com0)** and **(Com1)** define the interleaving behavior of parallel composition and rule **(Com2)** defines its communication (synchronization) behavior. Rule **(Con)** shows how recursive constants represent the behavior of their defining terms and finally, **(Rep)** defines the concept of replication.

By using our format, we can copy a number of structural congruences, defined in [12] for the π-calculus and thus, eliminate some of the deduction rules. The result is the following semantic specification.

$$\textbf{(Act)}\frac{}{\alpha.x \xrightarrow{\alpha} x} \qquad \textbf{(Res)}\frac{x \xrightarrow{\alpha} y}{x \setminus L \xrightarrow{\alpha} y \setminus L}(\alpha, \overline{\alpha} \notin L) \qquad \textbf{(NSum0)}\frac{x_0 \xrightarrow{\alpha} y}{x_0 + x_1 \xrightarrow{\alpha} y}$$

$$\textbf{(NCom0)}\frac{x_0 \xrightarrow{\alpha} y_0}{x_0 \mid\mid x_1 \xrightarrow{\alpha} y_0 \mid\mid y_1} \qquad \textbf{(NCom1)}\frac{x_0 \xrightarrow{l} y_0 \quad x_1 \xrightarrow{\bar{l}} y_1}{x_0 \mid\mid x_1 \xrightarrow{\tau} y_0 \mid\mid y_1}$$

$$\textbf{(struct)}\frac{x \equiv y \quad y \xrightarrow{l} y' \quad y' \equiv x'}{x \xrightarrow{l} x'} \qquad \begin{array}{ll} x + y \equiv y + x & x \mid\mid y \equiv y \mid\mid x \\ A \equiv P \quad (A \doteq P) & !x \equiv x \mid\mid !x \end{array}$$

Note that all of the SOS rules are in tyft format and the top two structural congruence equations are fx equations while the bottom ones are defining equations. Thus, one may easily deduce from Theorem 2 that strong bisimilarity with respect to the induced transition relation is a congruence. This can already be considered an achievement. However, one may argue that we could not specify some, may be more interesting, structural congruences of [12] such as those for associativity (for parallel composition and nondeterministic choice), idempotency (for nondeterministic choice) and zero element (again for both parallel composition and choice). Our answer to this criticism is that in general, the very same structural congruences (i.e, associativity, idempotency and zero element) can be harmful for congruence. Next, we give an intuitive example of an associativity equation that harms the congruence property.

Example 9 Take the semantics of our CCS-like language defined before. Suppose that we extend our syntax and semantics with a binary operator $\bullet$. The semantic rule for this operator is given by rule **(LMer)** $\dfrac{x_0 \overset{\alpha}{\to} y_0}{x_0 \bullet x_1 \overset{\alpha}{\to} y_0 \mid\mid x_1}$.

According to the above rule, this operator forces the first action to be taken by the left-hand-side argument and then turns into a normal parallel composition operator. (Up to here, this operator is similar to the left-merge operator of [2] which is usually used for finite axiomatization of parallel composition.) This operator, as defined by rule **(LMer)** is not associative. But, suppose that we also add the equation $x_0 \bullet (x_1 \bullet x_2) \equiv (x_0 \bullet x_1) \bullet x_2$ to our set of structural congruences, to make it associative.

Then, we can easily observe that the congruence property is ruined. For example, it holds that $0 \underline{\leftrightarrow} 0 \bullet \alpha$ (where α is a shorthand for $\alpha.0$), since none of the two can perform any action. However, it does not hold that $\alpha \bullet 0 \ \underline{\leftrightarrow}\ \alpha \bullet (0 \bullet \alpha)$. The left-hand term can only perform an α action and terminate (the structural congruence rule cannot help this term perform more actions since it should contain at least two left-merge operators to fit the structure of the equation). While the right-hand-term is congruent to $(\alpha \bullet 0) \bullet \alpha$ and this term can perform two consecutive α actions after the first of which it turns into $(0 \mid\mid 0) \mid\mid \alpha$.

8 Conclusions

In this paper, we gave an interpretation of structural congruences inside the transition system specification framework. Using this interpretation, we defined a syntactic congruence format for structural congruences. This format induces congruences for (strong) bisimilarity, once the structural congruences are used in combination with a set of standard (e.g., tyft) SOS rules. Furthermore, the relationship between negative premises in the deduction rules, structural congruences and well-definedness of the transition relation was investigated and a sufficient well-definedness criterium was established. To show the application of our format to a concrete example, we applied our syntactic format to a CCS-like process algebra.

Extending the syntactic format to other notions of equivalence and refinement is a possible extension of our work. Another important extension of our work concerns the notions of names and variable binding.

References

1. L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier Science, 2001.
2. J. C. M. Baeten and W. P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambrdige University Press, 1990.
3. J.-P. Banâtre, P. Fradet, and D. Le Métayer. Gamma and the chemical reaction model: Fifteen years after. In *Multiset Processing: Mathematical, Computer Science, and Molecular Computing Points of View*, volume 2235 of *LNCS*, pages 17–44. Springer, 2001.
4. G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96:217–248, 1992.
5. R. Bol and J. F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43(5):863–914, 1996.
6. R. van Glabbeek. The meaning of negative premises in transition system specifications II. *Journal of Logic and Algebraic Programming*, 60:229–258, 2004.
7. J. F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118(2):263–299, 1993.
8. J. F. Groote and F. W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.
9. K. Honda and N. Yoshida. On reduction-based process semantics. *Theoretical Computer Science*, 152(2):437–486, 1995.
10. J. J. Leifer and R. Milner. Deriving bisimulation congruences for reactive systems. In *Proceedings of CONCUR'00*, volume 1877 of *LNCS*, pages 259–274. Springer, 2000.
11. R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.
12. R. Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992.
13. R. Milner. The polyadic π-calculus: a tutorial. In *Logic and Algebra of Specification*, pages 203–246. Springer, 1993.
14. R. Milner and D. Sangiorgi. Barbed bisimulation. In *Proceedings of ICALP'92*, volume 623 of *LNCS*, pages 85–695. Springer, 1992.
15. M. Mousavi and M. Reniers. Structural congruences and structural operational semantics. Technical report CSR-04-28, Department of Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 2004.
16. D. M. Park. Concurrency and automata on infinite sequences. In *Proceedings of 5th GI Conference*, volume 104 of *LNCS*, pages 167–183. Springer, 1981.
17. G. D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Progamming*, 60:17–139, 2004.
18. V. Sassone and P. Sobociński. Deriving bisimulation congruences: 2-categories vs. precategories. In *Proceedings of FOSSACS'03*, volume 2620 of *LNCS*, pages 409–424. Springer, 2003.
19. P. Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science*, 274(1-2):183–230, 2002.
20. C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274–302, 1995.