Structural congruences and structural operational semantics

Structural Congruences and Structural Operational Semantics

MohammadReza Mousavi and Michel Reniers

Department of Computer Science,
Eindhoven University of Technology,
Post Box 513, NL-5600MB Eindhoven,
The Netherlands
{m.r.mousavi, m.a.reniers}@tue.nl

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 theses two semantic specification frameworks. In this paper, we try to fill this gap by accommodating structural congruences inside transition system specifications.
The Contributions of this paper can be summarized as follows:

1. Three interpretations of structural congruences in the SOS framework are presented;
2. The three interpretations are compared formally;
3. Syntactic criteria of a congruence format for structural congruences are given and proved correct;
4. Well-definedness criteria for transition system specifications with negative premises are extended to the setting with structural congruences;
5. Operational and equational conservative extensions of languages with structural congruences are studied.

Key words: Formal Semantics, Structural Operational Semantics (SOS), Structural Congruences, Transition System Specification, Congruence, Negative Premises.

1 Introduction

Structural congruences were introduced in [23, 24] 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 which is given in the Structural Operational Semantics (SOS) style of [30]. 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 (i.e., existence and uniqueness of the induced transition relation) and well-behavedness (e.g., congruence of bisimilarity) 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 [17]) 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 the well-definedness criteria defined in [9, 16] for SOS with negative premises do not necessarily suffice 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. [4], for example). This way, one may loose 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. Thus, this solution does not seem promising at all. 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. We propose a number of SOS interpretations for structural congruences and compare them formally. Congruence is by itself an interesting and essential property for bisimilarity (as an equivalence). Furthermore, it turns out that it plays a role in relating different interpretations of structural congruences. Thus, 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 Sections 6 and 7, we extend our format to cater for SOS rules with negative premises and study conservativity of operational extensions in the extended framework, respectively. To illustrate our congruence format with a concrete example, in Section 8, we apply it to a CCS-like process algebra. Finally, Section 9 concludes the paper and points out possible extensions of our work.

2 Related Work

This section is concerned with the origins of and recent developments in the two subject matters of this paper, namely, structural congruences and Structural Operational Semantics.

Structural congruences find their origin in the chemical models of computation [5]. The Chemical Abstract Machine (Cham) of [6] 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 [23, 24] in the semantic specification of the π-calculus. Before that, an equivalent semantics of the π-calculus had been presented in [25], in terms of "pure" SOS rules. As stated in [26], 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 [23, 24] and Section 8 of this paper).

Before that, in [21], structural congruences were presented for a subset of CCS under the name *flow-algebra* (rules). However, the use of structural congruences in [21] is essentially different from [23, 24]. In [21], structural congruences are a set of equalities on an algebraic signature. Similar to [21], in the ACP style of process algebra (e.g., in [4]), there is a conceptual distinction between such structural rules, as the equational theory of the algebra and the (transition system) semantics, as its model. From this point of view, SOS is a means to define the transition systems semantics and thus is not directly connected to the structural rules. Structural rules are to be proved sound (and possibly complete) with respect to the defined semantics and the particular notion of equivalence. In contrast, in [23, 24], they are used as a means to define or augment the operational semantics of a language. The present paper is concerned with the structural congruences in the sense of [23, 24].

The practice of using structural congruences for the specification of operational semantics, à la Milner [23], has continued since then. See [10], [11] and [12] for recent examples in defining operational semantics for Mobile Ambients, GAMMA and its coordination language, and the π-calculus, respectively.

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 [19, 20, 31, 32]. The lack of a congruent notion of bisimilarity for the semantics of the π-calculus has been known since [23] (which is not only due to structural congruences), but most attempts (e.g., [19, 20, 26, 31, 32]) were focused on deriving a suitable transition system (e.g., *contexts as labels* approach of [19, 20]) or a notion of equivalence (e.g., barbed congruence of [26]) that induces congruence. The works of [19, 20, 31, 32] deviate from the traditional interpretation of SOS deduction rules and establish a new semantic framework close to the reduction (reaction) rules of lambda calculus [18]. Arguably, this semantic framework is neither necessarily *structural*, i.e., following precisely the structure of syntax, nor *structured*, i.e., guaranteeing well-behavedness criteria such as congruence. For example, in [32], 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). Thus, 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.

Structural Operational Semantics [30, 29, 1] has been used and researched extensively since its introduction in [28]. In particular, several meta-theorems about SOS specifications have been proposed and proved that guarantee a SOS specification to induce a "decent" semantics (see [1] for an overview of such meta-theorems). The term *structured operational semantics* is coined in [17] for SOS specifications that conform to the requirements of such meta-theorems [1]. In this paper, we look at structural congruences as an extension to structured operational semantics. We consider three classes of these meta-theorems and extend them to the specifications enriched with structural congruences.

The first class of meta-theorems, considered in this paper, is about guaranteeing congruence for the notion of equivalence. The first proposal for such a meta-theorem was formulated in [33] by means of a standard syntactic format for SOS rules guaranteeing congruence for strong bisimilarity [27]. The congruence format of [33] is extended to the GSOS format (for Guardedly recursive SOS) in [8] and to the tyft format in [16]. Tyft stands for a coding of the structure of SOS rules explained in Section 3. We start with the tyft format as a simple, yet general framework for SOS and show how to add structural congruences to it. We prove sufficient conditions for this addition to preserve congruence for strong bisimilarity. Several extensions to the tyft format have been proposed to date (e.g., [16, 36] both of which have the merits of both tyft and GSOS

[1] There is a bit of dispute over what SOS should stand for. In [29], Plotkin insists that operational semantics is *structural*, i.e., syntax directed, and not *structured* as some took it. However, the word "structured" has been used in [8] and [17] in order to characterize a subset of SOS that induces a "reasonable" semantics. Thus, *structured operational semantic* is a (widely used) subset of Structural Operational Semantics.

formats). We also sketch how to generalize our meta-theorems to the more powerful formats of [16, 36].

The second class of meta-theorems, dealt with in this paper, is concerned with well-definedness of the operational semantics. It is not obvious whether an SOS specification defines a transition relation (thus, an operational semantics) and if it does, whether the induced transition relation is unique. This problem is first noted in [8] and investigated in detail in [9, 15, 16]. An extension of the tyft format, called the ntyft format, is proposed in [16] and sufficient conditions for well-definedness of the operational semantics are presented and proved. In this paper, we extend some of the well-definedness criteria of the ntyft format to structural congruences.

Finally, we consider extending conservativity meta-theorems. Due to changes in the requirements and applying languages in different applications, there is often a call for extending languages with new operators. This is the cause for another set of meta-theorems to appear within the realm of SOS specifications. These meta-theorems can provide sufficient conditions for such extensions to preserve the semantics of the original subset of the language and thus maintain the properties about the original language. This concept is called *conservativity* and has been studied extensively in the literature [14, 16, 17, 35, 37]. We show how adding definitions of new operators in terms of structural congruences may extend a language definition conservatively.

3 Preliminaries

We assume that the set of process terms, denoted by $T(\Sigma)$, 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) [17]) A *transition system specification* is a tuple (Σ, L, Rel, D) where Σ is a signature, L is a set of labels (with typical members $l, l', l_0, \ldots$), Rel is a set of transition relation symbols and D is a set of deduction rules. For all $l \in L$, $r \in Rel$ and $s, s' \in T(\Sigma)$ we define that $(s, l, s') \in \rightarrow_r$ 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 being closed and the concept of substitution are lifted to formulae in the natural way. A formula $(s, l, s') \in \rightarrow_r$ is denoted by the more intuitive notation $s \xrightarrow{l}_r s'$, as well. We refer to s as the source and to s' as the target of the transition. A deduction rule (H, c) is denoted by $\frac{H}{c}$ in the remainder. Although we define our basic concepts in a setting with more than one transition relation, most of the remainder (apart from Section 4.2) is conveniently and without loss of generality expressed in a setting with only one transition relation. In such cases, we drop the r subscript and denote the only transition relation by $\rightarrow$, $\rightarrow_0$, etc..

To define the transition relation induced by a transition system specification, we need to define its provable transitions.

Definition 2 (Provable Transitions) A *proof* of a closed formula ϕ (in a transition system specification tss) 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 exist a deduction rule $\dfrac{\{\chi_i \mid i \in I\}}{\chi}$ in tss and a substitution σ such that $\sigma(\chi) = \psi$, and for all $i \in I$, $\sigma(\chi_i) = \psi_i$.

A closed formula $p \xrightarrow{l} q$ is provable in a tss, notation $tss \vdash p \xrightarrow{l} q$, if there is a proof for it. The transition relations induced by tss are the minimal relations containing all provable formulae with the corresponding transition relation.

Note that for more complicated transition systems specifications such unique transition relations may not exist (see [1, 15] and Section 6 of the present paper for more details). Next, we define our notion of equality, namely, the infamous notions of strong bisimulation and bisimilarity.

Definition 3 (Bisimulation and Bisimilarity [27]) A relation $R \subseteq C(\Sigma) \times C(\Sigma)$ is a *bisimulation* relation with respect to a set of transition relations Rel if and only if $\forall_{p,q \in C(\Sigma)}\ (p,q) \in R \Rightarrow \forall_{r \in Rel, l \in L}$

1. $\forall_{p' \in C(\Sigma)}\ p \xrightarrow{l}_r p' \Rightarrow \exists_{q' \in C(\Sigma)}\ q \xrightarrow{l}_r q' \wedge (p', q') \in R$;
2. $\forall_{q' \in C(\Sigma)}\ q \xrightarrow{l}_r q' \Rightarrow \exists_{p' \in C(\Sigma)}\ p \xrightarrow{l}_r p' \wedge (p', q') \in R$.

Two closed terms p and q are are *bisimilar* with respect to Rel if and only if there exists a bisimulation relation R with respect to Rel such that $(p, q) \in R$. Two closed terms p and q are *bisimilar* with respect to a transition system specification tss, denoted by $tss \vdash p \leftrightarrow q$, if and only if they are bisimilar with respect to the transition relations induced by tss.

When it is clear from the context, we may drop the transition system specification from the provability and bisimilarity notations and simply use $p \xrightarrow{l} p'$ and $p \leftrightarrow q$. Bisimilarity is an equivalence relation and thus partitions the set of process terms into equivalence classes. We use this concept to define equality of transition relations and transition system specifications up to bisimilarity.

Definition 4 (Inclusion and Equality up to Bisimilarity) A transition relation $\rightarrow_0 \subseteq C(\Sigma) \times L \times C(\Sigma)$ is included in $\rightarrow_1 \subseteq C(\Sigma) \times L \times C(\Sigma)$ up to bisimilarity if and only if for all transitions $p \xrightarrow{l}_0 p'$, there exists a closed term p'' such that $p \xrightarrow{l}_1 p''$ and $p' \leftrightarrow_1 p''$ (where $\leftrightarrow_1$ is bisimilarity with respect to $\rightarrow_1$). Two transition relations are equal up to bisimilarity if and only if the inclusions hold in both directions.

Two transition system specifications are called equal if they induce the same transition relations. They are called equal up to bisimilarity if their induced transition relations are equal up to bisimilarity.

Note that inclusion (equality) of transition relations implies inclusion (equality) up to bisimilarity but not vice versa. Inclusion (up to bisimilarity) is reflexive and transitive and equality (up to bisimilarity) is an equivalence relation.

Corollary 1 Suppose that $\rightarrow_0 \subseteq C(\Sigma) \times L \times C(\Sigma)$ is equal to $\rightarrow_1 \subseteq C(\Sigma) \times L \times C(\Sigma)$ up to bisimilarity; then for all closed terms $p, q \in C(\Sigma)$, $p \leftrightarrow q$ with respect to $\rightarrow_0$ if and only if $p \leftrightarrow q$ with respect to $\rightarrow_1$.

Proof. Immediate consequence of Definition 4. ⊠

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

Definition 5 (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 $(p_i, q_i) \in R$ 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.

The following lemma (from [17]) represents the intuitive meaning of congruence, i.e., in a congruent relation, piecewise replacing of components with their related counterparts is allowed.

Lemma 1 Consider a relation R which is closed under congruence. If, for two substitutions σ and σ', it holds that $\forall_{x \in V}(\sigma(x), \sigma'(x)) \in R$, then we have $\forall_{t \in T(\Sigma)}\ (\sigma(t), \sigma'(t)) \in R$.

A result that comes in handy is that congruence of bisimilarity with respect to equal transition relations up to bisimilarity coincides.

Corollary 2 If transition relations $\rightarrow_0$ and $\rightarrow_1$ are equal up to bisimilarity then bisimilarity with respect to $\rightarrow_0$ is a congruence if and only if bisimilarity with respect to $\rightarrow_1$ is a congruence.

Proof. Immediate result of Corollary 1. ⊠

Bisimilarity is not in general a congruent relation. 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 [17] as a sufficiently general example of such formats for our purposes. Extensions to more general formats (such as the PANTH format of [36]) are discussed in Section 6.

Definition 6 (Tyft Format [17]) A rule is in tyft format[2] if and only if it has the following shape.

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

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.

For simplicity in our proofs, we assume that in a tyft rule all backward chains of premises in the variable dependency graph (defined below) are finite.

Definition 7 (Variable Dependency Graph) For a deduction rule, its variable dependency graph is a directed graph in which the nodes are the premises of the rule and edges denote dependencies between two premises, i.e., when a source of a premise uses a variable that appears in the target of the other, there is an edge from the latter to the former.

In [13], it is shown that the well-foundedness constraint can be relaxed and that for every tyft rule with a non-well-founded set of premises (with respect to the variable dependency ordering) there exists a rule that induces the same transition relation and is indeed well-founded. Thus, our results can be generalized to settings where the well-foundedness is not satisfied.

Theorem 1 (Congruence for tyft [17]) For a transition system specification in tyft format, bisimilarity is a congruence.

[2] Tyft is a code representing the structure of symbols in the deduction rule, namely a general term (t) in the source of the premises, a variable (y) in the target of the premises, a function symbol (f) in the source of the conclusion and a term (t) in the target of the conclusion.

4 Structural Congruences: Three Operational Interpretations

Structural congruences consist of a set of equations on open terms, denoted by $t \equiv t'$ on a given signature. As interpreted by [23], these equations induce a congruence (and equivalence) relation on closed terms. Then, they are connected to an SOS specification by means of a special deduction rule, stating that if a term can perform a transition, its congruent terms can mimic the same transition.

We take this interpretation as the original and intuitive meaning of structural congruences and give it a formal meaning in Section 4.1. Moreover, we present two alternative interpretations in Sections 4.2 and 4.3. In Section 4.2, we introduce the notation $\equiv$ as a new transition relation in the transition system specification. This way, equations of structural congruence, naturally turn into SOS axioms. Section 4.3 considers structural congruences as specifications of bisimilar terms. Thus, it adds two deduction rules for each equation, stating that if one side of a structural congruence equation can perform a transition, the other side can perform the same transition and vice versa.

Informally speaking, the first interpretation is the closest to the intuition behind structural congruences but as we move on to the second and the third, the resulting interpretation fits more in the transition system specification framework. While for the first interpretation, the notions of proof and provable transitions have to be adapted, for the second we only have to add a new transition relation and a number of deduction rules (thus, only a syntactic manipulation of transition system specifications) and in the third, even structural congruences do not show up in the transition system specification and just new deduction rules have to be added.

We also present a formal comparison of the three interpretations of structural congruence. In particular, we show that the first and the second interpretations, despite their different presentations, coincide. However, the third interpretation only coincides with the first two if the original transition system specification is in tyft format and furthermore bisimilarity is a congruence. In fact, the congruence condition turns out to be tricky, as structural congruences may jeopardize it even if they are added to a set of tyft rules (which by themselves guarantee the congruence). This sets the scene for Section 5, where we define syntactic criteria on structural congruences to derive congruence for strong bisimilarity.

4.1 External Interpretation

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

It can easily be checked that $\equiv_{sc}$ is symmetric and thus, alternatively, $\equiv_{sc}$ is the smallest equivalence and congruence relation satisfying structural congruences 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. In section 7, we study the situation where structural congruences may be used to extend the signature of a transition system specification.

To link structural congruences to a transition system specification, a special rule is used, which we call *the structural congruence rule*.

Definition 9 (The Structural Congruence Rule [23]) 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 $tss = (\Sigma, L, \{\rightarrow\}, D)$ and structural congruences sc on the same signature. The extension of tss with sc, denoted by $tss \cup \{\textbf{(struct)}\}$, is defined by the tuple $(\Sigma, L, \{\rightarrow\}, D \cup \{\textbf{(struct)}\})$.

There remains a problem concerning Definition 9, 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 we have not assigned any meaning to it, as yet. In fact, this subsection and the following two are concerned with different interpretations of the symbol $\equiv$. In this subsection, we do not interpret $\equiv$ directly, but rather exploit the structural congruence relation to extend the notion of proof. Syntactically, we allow for deduction rules of the following form:

$$\frac{\{\chi_i | i \in I\} \quad \{t_j \equiv t'_j | j \in J\}}{\chi}$$

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 10 (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 $\frac{\{\chi_i \mid i \in I\} \quad \{t_j \equiv t'_j | j \in J\}}{\chi}$ (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.

The following lemma shows that in our new framework, congruent (closed) terms are indeed bisimilar.

Lemma 2 Consider a transition system specification tss and structural congruences sc, and the bisimilarity relation $\underline{\leftrightarrow}$ with respect to $tss \cup \{\textbf{(struct)}\}$. It holds that $\equiv_{sc} \subseteq \underline{\leftrightarrow}$.

Proof. Immediate, from the definition of bisimilarity and the structural congruence rule by taking $\equiv_{sc}$ as a bisimulation relation. ⊠

We may simplify Definition 8 by adding a simpler rule to the transition system specification. The following (simpler) rule only allows for congruent terms to be replaced in the source of the transition.

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

Suppose that we replace rule **(struct)** in Definition 8 with **(struct')** and denote the new transition system specification by $tss \cup \{\textbf{(struct')}\}$. It turns out that this slight simplification amounts to a transition system specification that is equal up to bisimilarity, if the original tss is in tyft format and bisimilarity is a congruence. The following two Lemmas establish this fact:

Lemma 3 For an arbitrary transition system specification tss and structural congruences sc, the transition relation induced by $tss \cup \{\textbf{(struct)}\}$ is a subset of the transition relation induced by $tss \cup \{\textbf{(struct')}\}$ up to bisimilarity.

Proof. Let $\rightarrow_0$ and $\rightarrow_1$ be the two transition relations induced by $tss \cup \{\textbf{(struct)}\}$ and $tss \cup \{\textbf{(struct')}\}$, respectively. We have to show that $\rightarrow_0$ is included in $\rightarrow_1$ up to bisimilarity. Thus, we have to show that if $tss \cup \{\textbf{(struct)}\} \vdash p \xrightarrow{l}_0 p'$, for arbitrary closed terms p and p' and label l, then $tss \cup \{\textbf{(struct')}\} \vdash p \xrightarrow{l}_1 p''$ where $p'' \equiv_{sc} p'$. (We assumed a common set of structural congruences and thus $\equiv_{sc}$ is the same for both transition relations. It is a subset of bisimilarity for both, as well). We prove this by an induction on the depth of the proof tree resulting in this transition (see Definitions 2 and 8).

If transition $tss \cup \{\textbf{(struct)}\} \vdash p \xrightarrow{l}_0 p'$ has a proof of depth 1, then it should be due to an axiom in tss and a substitution σ (transitions due to **(struct)** have a proof depth of at least 2 since they always have a transition in their premises). Then, it immediately follows from the same axiom (using substitution σ) that there is a proof for $p \xrightarrow{l}_1 p'$ in $tss \cup \{\textbf{(struct')}\}$ (and $p' \equiv_{sc} p'$ holds trivially).

For the induction step, suppose that the transfer condition holds for all transitions with a proof of depth $n-1$ or less. Consider transition $p \xrightarrow{l}_0 p'$ which has a proof of depth n. If the transition is due to a rule in tss of the following form (for an arbitrary n-ary function symbol f):

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

then there exists a substitution σ such that $\sigma(x_i) = p_i$ $(0 \leq i < n)$, $\sigma(t) = p'$ and $p = f(p_0, \ldots, p_n)$.

Since we assumed acyclicity of the variable dependency graph, we can define a rank, $rank(x)$, for each variable x, as the maximum length of a backward chain starting from x in the variable dependency graph. The rank of a premise is the rank of its target variable. Then, for each $x \in vars(t_i)$ of each premise $t_i \xrightarrow{l_i} y_i$ of the deduction rule, it holds that $rank(x) < rank(y_i)$.

Take $Y = \{y_i | i \in I\}$. We define new substitution σ' such that for all $x \in V \setminus Y$, $\sigma'(x) \doteq \sigma(x)$. Note that thus far this substitution is not defined for variables in Y. We extend the definition while proving, by induction on the rank of a premise r, the preservation of three essential properties:

1. $\sigma(t_i) \equiv_{sc} \sigma'(t_i)$;
2. $\sigma'(t_i) \xrightarrow{l_i}_1 \sigma'(y_i)$;
3. $\sigma(y_i) \equiv_{sc} \sigma'(y_i)$.

Take any premise, say some $i \in I$, that has a minimal rank of which the target variable y_i is not defined under σ'. The source term of this premise (t_i) is fully interpreted under σ' (i.e., $\sigma'(t_i)$ is a closed term) and it follows from Lemma 1 (since $\equiv_{sc}$ is a congruence) and the above constraints that $\sigma(t_i) \equiv_{sc} \sigma'(t_i)$. Since $\sigma(t_i) \xrightarrow{l}_0 \sigma(y_i)$ has a proof of depth $n-1$ or less, it follows from the induction hypothesis that $\sigma'(t_i) \xrightarrow{l}_1 p'_i$ and $\sigma(y_i) \equiv_{sc} p'_i$ for some p'_i. Then take $\sigma'(y_i) \doteq p'_i$. Using this inductive procedure, we complete the definition of σ' for all y_i's and thus for all variables x, maintaining the property $\sigma(x) \equiv_{sc} \sigma'(x)$. Then, using the same rule, we have a proof for $\sigma'(f(x_0, \ldots, x_{n-1})) \xrightarrow{l}_1 \sigma'(t)$, or $p \xrightarrow{l}_1 \sigma'(t)$ and it holds that $p' = \sigma(t) \equiv_{sc} \sigma'(t)$.

If the transition is due to the congruence rule of the following form

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

then there is a substitution σ such that for some q and q', $\sigma(x) = p$, $\sigma(y) = q$, $\sigma(y') = q'$ and $\sigma(x') = p'$ and it holds that $p \equiv_{sc} q$, $p' \equiv_{sc} q'$ and $q \xrightarrow{l}_0 q'$. By applying the induction hypothesis on $q \xrightarrow{l}_0 q'$, we get $q \xrightarrow{l}_1 q''$ for some q'' such that $q'' \equiv_{sc} q'$ and by symmetry and transitivity of $\equiv_{sc}$, we get $p' \equiv q''$. Thus, by using deduction rule

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

and a substitution σ' such that $\sigma'(x) = p$, $\sigma'(y) = q$, $\sigma'(y') = q''$, we have a proof for $p \xrightarrow{l}_1 q''$ and we have already shown that $p' \equiv_{sc} q''$. This concludes the proof. ⊠

The above lemma cannot be generalized to arbitrary transition system specifications (not in tyft format). The following example illustrates this fact.

Example 1 Consider the following structural congruence sc and transition system specification tss.

$$a \equiv f(b)$$

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

Suppose that the common signature contains a and b as constants and f as a unary function symbol. Since $a \equiv f(b)$, using **(a)** and **(struct)**, we can prove that $a \xrightarrow{l_0} f(b)$ and hence it follows from **(xb)** that $a \xrightarrow{l_0} b$. However, using **(struct')** we are not able to prove $a \xrightarrow{l_0} b$ since $a \xrightarrow{l_0} f(b)$ is not provable anymore. Furthermore, we cannot prove $a \xrightarrow{l_0} p$ for any $p \leftrightarrows b$. This shows that the two transition system specifications $tss \cup \{\textbf{(struct)}\}$ and $tss \cup \{\textbf{(struct')}\}$ are not equal up to bisimilarity.

Inclusion of the transition relation induced by $tss \cup \{\textbf{(struct')}\}$ in $tss \cup \{\textbf{(struct)}\}$, however, does not require any assumption about the transition system specification.

Lemma 4 For an arbitrary transition system specification tss and structural congruences sc, the transition relation induced by $tss \cup \{\textbf{(struct')}\}$ is a subset of the transition relation induced by $tss \cup \{\textbf{(struct)}\}$.

Proof. Straightforward from an induction on the depth of the proof of transitions in $tss \cup \{\textbf{(struct')}\}$. Note that $\equiv_{sc}$ is reflexive and thus any transition provable from rule **(struct')** is also provable from **(struct)**. ⊠

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

$$a \equiv f(b)$$

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

In the above specification, both a and b can perform an l_0 transitions to a due to rules **(a)** and **(b)**, respectively. On one hand, using Definition 8, a is only congruent 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 the congruence property by structural congruences are presented in the remainder of this paper.

4.2 Transition Relation Interpretation

The second interpretation of structural congruences, considers $\equiv$ as a new transition relation in the transition system specification. Thus, structural congruence equation $t \equiv t'$ is interpreted as the following SOS axiom:

$$\textbf{(tt')}\frac{}{t \equiv t'}$$

To be more precise $\equiv$ is defined as the pair $(\rightarrow_n, l_n)$ where $\rightarrow_n$ is a fresh transition relation and l_n is a fresh label. (In fact, freshness of either of the two suffices.) Since transition relations are directed, we have to add another deduction rule to account for the natural symmetry in $\equiv$:

$$\textbf{(t't)}\frac{}{t' \equiv t}$$

Also, to account for reflexivity, transitivity and congruence of $\equiv$, we have to add the following rules to the transition system specification:

$$\textbf{(refl)}\frac{}{x \equiv x} \qquad \textbf{(trans)}\frac{x \equiv y \quad y \equiv z}{x \equiv z}$$

$$\textbf{(congf)}\frac{\{x_i \equiv y_i \mid 0 \leq i < ar(f)\}}{f(x_0, \ldots, x_{ar(f)-1}) \equiv f(y_0, \ldots, y_{ar(f)-1})} \quad \text{(for all } f \in \Sigma)$$

Then, the structural congruence rule

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

fits very well in the definition of a transition system specification (Definition 1) since $x \equiv y$ is now a valid formula.

We summarize the new interpretation of structural congruences in the following definition.

Definition 11 (Structural Congruences as a Transition Relation) The interpretation of structural congruences sc on signature Σ with a transition system specification $tss = (\Sigma, L, \{\rightarrow\}, D)$ is a

new transition system specification $tss \cup \langle\langle sc \rangle\rangle \doteq (\Sigma, L \cup \{l_n\}, \{\rightarrow, \rightarrow_n\}, D \cup \langle\langle sc \rangle\rangle)$, where $\langle\langle sc \rangle\rangle$ is defined as follows:

$$\langle\langle sc \rangle\rangle \quad \doteq \quad \{\textbf{(refl)}, \textbf{(trans)}, \textbf{(struct)}\} \cup \{\textbf{(congf)} \mid f \in \Sigma\} \cup \left\{ \begin{array}{l} \textbf{(tt')}\dfrac{}{t \equiv t'}, \\ \\ \textbf{(t't)}\dfrac{}{t' \equiv t} \end{array} \;\middle|\; (t \equiv t') \in sc \right\}$$

where **(refl)**, **(trans)**, **(congf)** and **(struct)** are the reflexivity, transitivity, congruence and structural congruence rules, respectively, as defined before.

We are now in the position to rephrase Lemma 2 for the new interpretation. Namely, we can now prove that for all provable (transitions) $p \equiv q$, p and q are bisimilar.

Lemma 5 Consider a transition system specification tss and structural congruences sc. Take Σ to be the common signature of sc and tss. Then, for all $p, q \in C(\Sigma)$ such that $tss \cup \langle\langle sc \rangle\rangle \vdash p \equiv q$, we have $tss \cup \langle\langle sc \rangle\rangle \vdash p \leftrightarrow q$.

Proof. Take R to be the set of all pairs of closed terms (p, q) such that $p \equiv q$ is a provable transition. Consider an arbitrary pair $(p, q) \in R$; we have to prove that for all $p' \in C(\Sigma)$ and for all transitions of p (both transitions of the form $p \xrightarrow{l} p'$ and $p \equiv p'$) there exists a q' such that q can make the same transitions to q' and $(p', q') \in R$ (and vice versa which is symmetric to this case).

1. If $tss \cup \langle\langle sc \rangle\rangle \vdash p \xrightarrow{l} p'$, since $(p, q) \in R$ and R is symmetric by construction, $(q, p) \in R$. Thus, $tss \cup \langle\langle sc \rangle\rangle \vdash q \equiv p$. Hence, it follows from **(struct)** that $tss \cup \langle\langle sc \rangle\rangle \vdash q \xrightarrow{l} p'$ using premises $q \equiv p$, $p \xrightarrow{l} p'$ and $p' \equiv p'$ (the last statement follows from rule **(refl)**).
2. If $tss \cup \langle\langle sc \rangle\rangle \vdash p \equiv p'$, we already know that $tss \cup \langle\langle sc \rangle\rangle \vdash q \equiv p$ (see the previous item) and thus it follows from **(trans)** that $tss \cup \langle\langle sc \rangle\rangle \vdash q \equiv p'$ and again $(p', p') \in R$ due to **(refl)**.

⊠

To compare the two interpretations given up to now, we first observe that the congruent classes of closed terms coincide for those.

Lemma 6 For two closed terms p and q, $p \equiv_{sc} q$ if and only if $p \equiv q$ is provable from $tss \cup \langle\langle sc \rangle\rangle$.

Proof. By straightforward inductions on the structure of $\equiv_{sc}$ and depth of the proof for $p \equiv q$ in $tss \cup \langle\langle sc \rangle\rangle$. ⊠

Using this lemma, we can easily show that the transition relations $\rightarrow$ induced by the two interpretations coincide.

Theorem 2 For arbitrary closed terms p and p' and arbitrary label l, $p \xrightarrow{l} p'$ is provable from $tss \cup \{\textbf{(struct)}\}$ if and only if it is provable from $tss \cup \langle\langle sc \rangle\rangle$.

Proof. By a straightforward induction on the depth of the proofs for the transition $p \xrightarrow{l} p'$ and by using Lemma 6. ⊠

Also, it can be easily proven that replacing **(struct)** with **(struct')** in the definition of $\langle\langle sc \rangle\rangle$ results in an equal transition relation up to bisimilarity, provided that the necessary condition of Lemma 3 hold. We dispense with repeating the lemma and the proof. We conclude this subsection by emphasizing that although the first and the second interpretations have different presentations, they are formally equal.

4.3 Bisimilarity Interpretation

An alternative way of interpreting structural congruences is to say that congruent terms should be able to mimic each others' transitions. In other words, the equation $t \equiv t'$ is interpreted as $\sigma(t) \leftrightarrow \sigma(t')$ for all closed terms $\sigma(t)$ and $\sigma(t')$. To realize this interpretation in terms of SOS, we define a pair of rules for each equation (and for each label and transition relation), to prove all transitions of one side for the other side and vice versa. This concept is formally defined as follows.

Definition 12 (Structural Congruences as Bisimilarity) Consider structural congruences sc on signature Σ and a transition system specification $tss = (\Sigma, L, \{\rightarrow\}, D)$. We define a new transition system specification $tss \cup [[sc]] \doteq (\Sigma, L, \{\rightarrow\}, D \cup [[sc]])$, where $[[sc]]$ is the SOS interpretation of sc, defined as follows:

$$[[sc]] \doteq \left\{ \begin{array}{l|l} \textbf{(btt')} \dfrac{t \xrightarrow{l} y}{t' \xrightarrow{l} y}(l \in L), & \\ \\ \textbf{(bt't)} \dfrac{t' \xrightarrow{l} y}{t \xrightarrow{l} y}(l \in L) & (t \equiv t') \in sc \end{array} \right\}$$

In the above definition, y is a fresh variable (i.e., $y \notin vars(t)$ and $y \notin vars(t')$).

We cannot present a direct proof for Lemma 2 (or similarly, Lemma 6) in this setting since $p \equiv_{sc} q$ (or $p \equiv q$) do not have any clear semantic counterpart in this interpretation. However, we can indirectly prove a similar result as of Lemmas 2 and 6. Namely, we can show that if $p \equiv_{sc} q$ (thus, $tss \cup \langle\langle sc \rangle\rangle \vdash p \equiv q$) , then $tss \cup [[sc]] \vdash p \leftrightarrow q$ provided that bisimilarity is a congruence. Next, we prove a lemma that establishes the aforementioned fact.

Lemma 7 Consider a transition system specification tss and structural congruences sc. If $p \equiv_{sc} q$ and bisimilarity w.r.t. $tss \cup [[sc]]$ is a congruence, then it holds that $tss \cup [[sc]] \vdash p \leftrightarrow q$.

Proof. We prove the lemma by an induction on the structure of $\equiv_{sc}$:

1. Reflexivity: If $p \equiv_{sc} q$ is due to reflexivity, then the lemma follows trivially.
2. Transitivity: If $p \equiv_{sc} q$ is due to transitivity, then there exists a closed term s such that $p \equiv_{sc} s$ and $s \equiv_{sc} q$. Then, according to the induction hypothesis $tss \cup [[sc]] \vdash p \leftrightarrow s$ and $tss \cup [[sc]] \vdash s \leftrightarrow q$ and since $\leftrightarrow$ is transitive $tss \cup [[sc]] \vdash p \leftrightarrow q$.
3. Structural congruences: Then there is an equation $t \equiv t'$ (or similarly $t' \equiv t$, which is symmetric to this case) and a substitution σ such that $\sigma(t) = p$ and $\sigma(t') = q$. It follows from Definition 12 that there is a rule **(btt')** in $tss \cup [[sc]]$ of the following form:

   $$\textbf{(btt')} \frac{t \xrightarrow{l} y}{t' \xrightarrow{l} y}(l \in L)$$

   Then, suppose that $p \xrightarrow{l} p'$ for an arbitrary l and p', by taking $\sigma'(x) \doteq \sigma(x)$ for $x \neq y$ and $\sigma'(y) \doteq p'$, we can derive $tss \cup [[sc]] \vdash q \xrightarrow{l} p'$ using the above rule and substitution σ' ($p' \leftrightarrow p'$ holds trivially).
4. Congruence: If $p \equiv_{sc} q$ is due to congruence, then, $p = f(p_0, \ldots, p_{n-1})$, $q = f(q_0, \ldots, q_{n-1})$ and $p_i \equiv_{sc} q_i$ $(0 \leq i < n)$. It follows from the induction hypothesis that $tss \cup [[sc]] \vdash p_i \leftrightarrow q_i$ and since bisimilarity is a congruence $tss \cup [[sc]] \vdash p = f(p_0, \ldots, p_{n-1}) \leftrightarrow f(q_0, \ldots, q_{n-1}) = q$.

$\boxtimes$

To compare this interpretation with the previous two interpretations, it suffices to compare it with one of them (as they are formally proved equal). Thus, we compare this interpretation with the first one. Next, we show that the transitions introduced by this interpretation are included in the transition relation induced by the first one.

Theorem 3 For arbitrary closed terms p and p' and arbitrary label l if $tss \cup [[sc]] \vdash p \xrightarrow{l} p'$ then $tss \cup \{\textbf{(struct)}\} \vdash p \xrightarrow{l} p'$.

Proof. By an induction on the proof for $tss \cup [[sc]] \vdash p \xrightarrow{l} p'$. For the induction basis, if the proof has depth one then it is due to an axiom in tss and a substitution σ. Using the same axiom and the same substitution, we can derive that $tss \cup \{\textbf{(struct)}\} \vdash p \xrightarrow{l} p'$.

For the induction step, suppose that the theorem holds for all formulae with a proof of depth $n-1$ or less and suppose that $tss \cup [[sc]] \vdash p \xrightarrow{l} p'$ is due to a proof of depth n.

If the last deduction rule in the proof tree is in tss then since all the premises of the rule have a proof of depth $n-1$ or less, they are all provable in $tss \cup \{\textbf{(struct)}\}$. Thus, using the same rule and same substitution, we have $tss \cup \{\textbf{(struct)}\} \vdash p \xrightarrow{l} p'$.

If the last deduction rule has the following form:

$$\textbf{(btt')}\ \frac{t \xrightarrow{l} y}{t' \xrightarrow{l} y}(l \in L)$$

then, there exists a substitution σ such that $\sigma(t') = p$ and $\sigma(y) = p'$. According to Definition 12, there exists an equation $t \equiv t'$ (or symmetrically, $t' \equiv t$) in sc. On one hand, using σ, we can derive that $\sigma(t) \equiv_{sc} p$ and since $\equiv_{sc}$ is symmetric, $p \equiv_{sc} \sigma(t)$. Also, it follows from reflexivity of $\equiv_{sc}$ that $p' \equiv_{sc} p'$. On the other hand, since $\sigma(t) \xrightarrow{l} p'$ has a proof of depth $n-1$, it follows from the induction hypothesis that $tss \cup \{\textbf{(struct)}\} \vdash \sigma(t) \xrightarrow{l} p'$. Using premises $p \equiv_{sc} \sigma(t)$, $\sigma(t) \xrightarrow{l} p'$ and $p' \equiv_{sc} p'$, we can prove from **(struct)** that $tss \cup \{\textbf{(struct)}\} \vdash p \xrightarrow{l} p'$. $\boxtimes$

The above theorem establishes an inclusion result in one direction. To give a full comparison, it remains to give a comparison in the other direction. Next, we give an indirect result leading to such a full comparison.

Theorem 4 Consider a transition system specification tss in tyft format and structural congruences sc. Suppose that bisimilarity with respect to $tss \cup [[sc]]$ is a congruence then the transition relation induced by $tss \cup \{\textbf{(struct)}\}$ is included in the transition relation induced by $tss \cup [[sc]]$ up to bisimilarity.

Proof. We have to prove for arbitrary closed terms p and p' and arbitrary label l that if $tss \cup \{\textbf{(struct)}\} \vdash p \xrightarrow{l} p'$ then $tss \cup [[sc]] \vdash p \xrightarrow{l} p''$ and $tss \cup [[sc]] \vdash p'' \underline{\leftrightarrow} p'$. We show this by an induction on the depth of the proof for $p \xrightarrow{l} p'$ in $tss \cup \{\textbf{(struct)}\}$.

If transition $p \xrightarrow{l} p'$ is provable from $tss \cup \{\textbf{(struct)}\}$ with a proof of depth one, then it is due to an axiom in tss and a substitution σ. By taking the same axiom and substitution, we can prove $tss \cup [[sc]] \vdash p \xrightarrow{l} p'$.

For the induction step, if the transition $p \xrightarrow{l} p'$ is provable from $tss \cup \{\textbf{(struct)}\}$ with a proof of depth n, then we distinguish the following two cases.

If the transition is due to a rule in tss of the following form:

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

and a substitution σ such that $\sigma(x_i) = p_i$ $(0 \leq i < n)$, $\sigma(t) = p'$ and $p = f(p_0,\ldots,p_n)$ and take $Y = \{y_i | i \in I\}$. We aim at defining a new substitution σ' in a similar way as we have defined it in the proof of Lemma 3. Namely, we define that for all $x \in V \setminus Y$, $\sigma'(x) \doteq \sigma(x)$. To complete the definition of σ' for the variables in Y, we start with the premise of which all variables in the source are defined in σ' and the variable in the target is undefined. Such a premise should exist due to well-foundedness of premises with respect to variable-dependency order. Suppose such a premise is $t_i \xrightarrow{l_i} y_i$. Then, since bisimilarity is a congruence and according to the construction of σ', it follows from Lemma 1 that $\sigma(t_i) \leftrightarrows \sigma'(t'_i)$. Transition $\sigma(t_i) \xrightarrow{l_i} \sigma(y_i)$ has a proof of depth $n-1$ or less and according to the induction hypothesis, there exists a closed term p'_i such that $\sigma'(t_i) \xrightarrow{l_i} p'_i$ and $\sigma(y_i) \leftrightarrows p'_i$. Then, take $\sigma'(y_i) \doteq p'_i$. Using this scheme, we are able to define σ' inductively for all variables y_i in such a way that $\forall_{x \in V} \sigma(x) \leftrightarrows \sigma'(x)$. This way, we complete a proof for $p \xrightarrow{l} \sigma'(t)$ using σ' and **(d)** and it follows from the construction of σ' and Lemma 1 that $\sigma(t) \leftrightarrows \sigma'(t')$.

It remains to prove the case where the transition $p \xrightarrow{l} p'$ is due to **(struct)**:

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

and substitution σ such that $\sigma(x) = p$, $\sigma(x') = p'$ and there exists two closed terms q and q' such that $\sigma(y) = q$, $\sigma(y') = q'$, $p \equiv_{sc} q$, $q' \equiv_{sc} p'$ and $q \xrightarrow{l} q'$. Since $q \xrightarrow{l} q'$ has a proof of depth $n-1$, there should exist a closed term q'' such that $tss \cup [[sc]] \vdash q \xrightarrow{l} q''$ and $tss \cup [[sc]] \vdash q' \leftrightarrows q''$. From $q' \equiv p'$ and Lemma 7, it follows that $tss \cup [[sc]] \vdash q' \leftrightarrows p'$ and since bisimilarity is transitive, it holds that $tss \cup [[sc]] \vdash q'' \leftrightarrows p'$ and this concludes the proof. ⊠

The above theorem implies that the three interpretation coincide up to bisimilarity, provided that some necessary conditions (i.e., *the*tyft format for the transition system specification and congruence of bisimilarity) hold. Thus, if the necessary conditions of Theorem 4 hold, one may choose among these interpretations at will and the result will always be valid for the other interpretations (up to bisimilarity). However, the coincidence result between these interpretations relies on congruence of bisimilarity (w.r.t. the third interpretation) and conformance of the transition system specification to the tyft format. The following two examples show that when these necessary conditions do not hold, this interpretation may deviate from the first two (and the first interpretation with **(struct')**). An explanation of the reason for such necessary conditions comes after each example.

Example 3 Consider the following structural congruence sc and transition system specification tss.

$$a \equiv b \qquad \textbf{(fb)}\frac{}{f(b) \xrightarrow{l_0} b} \qquad \textbf{(b)}\frac{}{b \xrightarrow{l_1} b}$$

If we interpret structural congruences according to our first interpretation, we have that $a \equiv_{sc} b$ and $f(a) \equiv_{sc} f(b)$ and it follows from Lemma 2 that $a \leftrightarrows b$ and $f(a) \leftrightarrows f(b)$. Thus, for example, in this interpretation, transition $f(a) \xrightarrow{l_0} b$ is provable using the above structural congruences and **(struct)**.

According to the third interpretation, $tss \cup [[sc]]$ comprises the following deduction rules:

$$\textbf{(bab)}\frac{a \xrightarrow{l} y}{b \xrightarrow{l} y}(l \in L) \qquad \textbf{(bba)}\frac{b \xrightarrow{l} y}{a \xrightarrow{l} y}(l \in L)$$

$$\textbf{(fb)}\frac{}{f(b) \xrightarrow{l_0} b} \qquad \textbf{(b)}\frac{}{b \xrightarrow{l_1} b}$$

However, from the above transition system specification, we can not derive that $f(a) \equiv f(b)$ and what is more important, there is no way to prove $f(a) \xrightarrow{l_0} b$, anymore.

One may notice that the above problem has to do with the lack of congruence property in our transition system specification; we can derive from Lemma 5 that $a \leftrightarrows b$ but apparently, it does not hold that $f(a) \leftrightarrows f(b)$. This is indeed the case. The deduction rules in tss do not conform to tyft format and even worse, their induced bisimilarity is not a congruence in the first place. Next, we give another counter-example, showing that even if the original transition system specification is in tyft, adding structural congruences according to the third interpretation may violate the intuition.

Example 4 Consider structural congruences sc and transition system specification tss as defined below.

$$a \equiv b \qquad f(b) \equiv f(c)$$

$$\textbf{(fx)}\frac{x \xrightarrow{l_0} y}{f(x) \xrightarrow{l_0} f(y)} \qquad \textbf{(c)}\frac{}{c \xrightarrow{l_0} c}$$

Suppose that we have a common signature with constants a, b and c and a unary function symbol f. Then, according to the first interpretation of structural congruences, we have $a \equiv_{sc} b$ and thus $f(a) \equiv_{sc} f(b)$. Since we also have $f(b) \equiv_{sc} f(c)$, it follows from the transitivity condition that $f(a) \equiv_{sc} f(c)$. According to **(fx)**, $f(c) \xrightarrow{l_0} f(c)$ and thus it follows from **(struct)** that $f(a) \xrightarrow{l_0} f(c)$.

According to the third interpretation, $tss \cup [[sc]]$ is defined as:

$$\textbf{(bab)}\frac{a \xrightarrow{l} y}{b \xrightarrow{l} y} \qquad \textbf{(bba)}\frac{b \xrightarrow{l} y}{a \xrightarrow{l} y}$$

$$\textbf{(bfbfc)}\frac{f(b) \xrightarrow{l} y}{f(c) \xrightarrow{l} y} \qquad \textbf{(bfcfb)}\frac{f(c) \xrightarrow{l} y}{f(b) \xrightarrow{l} y}$$

$$\textbf{(fx)}\frac{x \xrightarrow{l_0} y}{f(x) \xrightarrow{l_0} f(y)} \qquad \textbf{(c)}\frac{}{c \xrightarrow{l_0} c}$$

but in the above transition system specification transition $f(a) \xrightarrow{l_0} f(c)$ is not provable anymore.

Again the above problem is due to the lack of the congruence property. In the above transition system specification, it clearly holds that $a \leftrightarrows b$ but it does not hold that $f(a) \leftrightarrows f(b)$. In the next section, we aim at giving a solution to guarantee this criterion.

Figure 1 summarizes the comparison of the three interpretations. In this figure, normal and dashed arrows mean inclusion and inclusion up to bisimilarity of transition relations in the indicated direction, respectively. All the dashed arrows require the tyft format for tss as a necessary condition. For the two cases involving Theorem 4, the dashed arrows also require the congruence of bisimilarity for their target interpretation.

Regarding the congruence conditions, note that congruence for the third interpretation implies congruence for the first and the second one, provided that the tyft condition holds (following

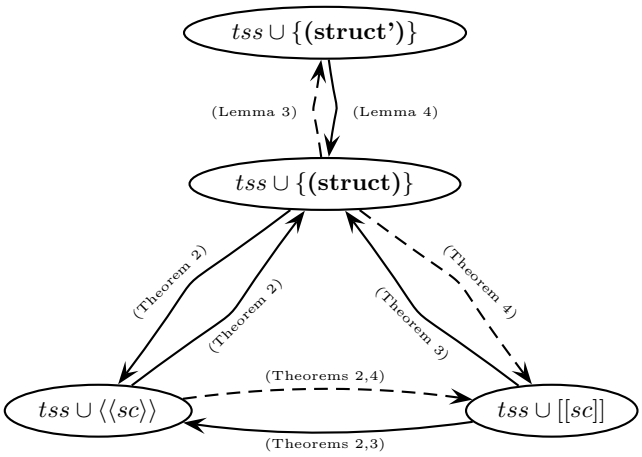

Figure 1: Interpretations of Structural Congruences

Corollary 2 as the transition relations coincide up to bisimilarity). While congruence of bisimilarity for the first and the second interpretations does not have any general implication for the congruence of bisimilarity with respect to the third one. Recall the following transition system specification and structural congruences from Example 4.

$$a \equiv b \qquad f(b) \equiv f(c)$$

$$\textbf{(fx)}\frac{x \xrightarrow{l_0} y}{f(x) \xrightarrow{l_1} f(y)} \qquad \textbf{(c)}\frac{}{c \xrightarrow{l_0} c}$$

For the above specification, it can be checked that bisimilarity is a congruence according to the first interpretation (derivable bisimilarities are $a \underline{\leftrightarrow} b$ and $f(a) \underline{\leftrightarrow} f(b) \underline{\leftrightarrow} f(c) \underline{\leftrightarrow} f(f(a)) \underline{\leftrightarrow} \ldots$). However, we have already shown that in the transition relation induced by the third interpretation, bisimilarity is not a congruence as it holds that $a \underline{\leftrightarrow} b$ but not $f(a) \underline{\leftrightarrow} f(b)$. Thus, for our congruence format to be useful for all the three notions, we have to prove it correct with respect to the third interpretation. This way, the congruence format not only induces congruence with respect to the other two notions, it also guarantees that for specifications in the standard format, all the three interpretations coincide and they can be freely chosen at one's convenience.

5 Congruence for Structural Congruences

In this section, we propose a syntactic format for structural congruences and prove that structural congruences conforming to this format are safe for the purpose of congruence when added to a set of tyft rules. As justified in Section 4, we use the third interpretation of structural congruences to prove our format correct. Then, 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 13 (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, $t \equiv f(x_0, \ldots, x_{ar(f)-1})$ which we do not mention in the remainder due to symmetry) 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 his 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. Note that 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 t may consist of any number of constants and function symbols.

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 induce a congruent bisimilarity relation (with respect to all the three interpretations) when added to a set of tyft rules.

Theorem 5 (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 is a congruence for all the transition relations induced by the three interpretations of tss extended with sc.

Proof. We prove the theorem for the third interpretation and it follows from Theorem 3 and 4 and Corollary 2 that bisimilarity is a congruence for the first interpretation. Also, from Theorem 2, it follows that bisimilarity is congruence for the second interpretation. Since tss is in tyft format, it also follows from Lemmas 3 and 4 that bisimilarity is a congruence for $tss \cup \{\textbf{(struct')}\}$, as well.

We give an indirect proof for this theorem. First, we give a slightly simplified interpretation of structural congruences in the cfsc format, denoted by $tss \cup [[sc]]^*$. The simplification is only concerned with defining rules. Consider a defining equation of the form $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$; this equation is aimed at defining the operational behavior of f, thus the following rule introduced by the third interpretation seems redundant.

$$\textbf{(bft)} \frac{f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} y}{t \xrightarrow{l} y}$$

The simplification only eliminates rules of the above shape. As a consequence of this simplification the resulting transition system specification is naturally in tyft format. Thus, the congruence of bisimilarity follows from Theorem 1. Then, we prove that for a tss in tyft and sc in cfsc, $tss \cup [[sc]]$ and $tss \cup [[sc]]^*$ are equal and we conclude that bisimilarity is a congruence for $tss \cup [[sc]]$, as well.

Definition 14 (Structural Congruences as Bisimilarity: Simplified) Consider structural congruences sc on signature Σ and a transition system specification $tss = (\Sigma, L, \{\rightarrow\}, D)$. We define a new transition system specification $tss \cup [[sc]]^* \doteq (\Sigma, L, \{\rightarrow\}, D)$, where $[[sc]]^*$ is the SOS

interpretation of sc, defined as follows:

Fx equations: $[[f(x_0,\ldots,x_{ar(f)-1}) \equiv g(y_0,\ldots,y_{ar(g)-1})]]^* \doteq$

$$\left\{ \begin{array}{l} \textbf{(bfg)}\dfrac{f(x_0,\ldots,x_{ar(f)-1}) \xrightarrow{l} y}{g(y_0,\ldots,y_{ar(g)-1}) \xrightarrow{l} y}(l \in L), \\ \\ \textbf{(bgf)}\dfrac{g(y_0,\ldots,y_{ar(g)-1}) \xrightarrow{l} y}{f(x_0,\ldots,x_{ar(f)-1}) \xrightarrow{l} y}(l \in L) \end{array} \right\};$$

Defining equations: $[[f(x_0,\ldots,x_{ar(f)-1}) \equiv t]]^* \doteq \{\textbf{(fdef)}\dfrac{t \xrightarrow{l} y}{f(x_0,\ldots,x_{ar(f)-1}) \xrightarrow{l} y}(l \in L)\}$;

$[[sc]]^* \doteq \bigcup_{(t \equiv t') \in sc} [[t \equiv t']]^*$.

where in all the deduction rules y is a fresh variable not appearing in any other term in the source of same deduction rule (i.e., $y \notin \{x_y, y_j | 0 \leq i < ar(f) \wedge 0 \leq j < ar(g)\}$). As stated before, for equations matching both fx and defining equations, one can choose any of the above definitions at will.

It can be easily observed from the above construction that if tss is in tyft and sc is in cfsc, then $tss \cup [[sc]]^*$ is in tyft. Thus, it follows from Theorem 1 that bisimilarity is a congruence for $tss \cup [[sc]]^*$.

Next, we show that for transition systems specification tss in tyft and structural congruence sc in cfsc, $tss \cup [[sc]]$ and $tss \cup [[sc]]^*$ are equal.

For arbitrary closed terms p and p' and arbitrary label l, if $tss \cup [[sc]] \vdash p \xrightarrow{l} p'$, we prove that $tss \cup [[sc]]^* \vdash p \xrightarrow{l} p'$. We use an induction on the depth of the proof for $p \xrightarrow{l} p'$ in $tss \cup [[sc]]$. The implication in the other direction holds vacuously as the set of deduction rules of $tss \cup [[sc]]^*$ is a subset of that of $tss \cup [[sc]]$.

For the induction basis, the transition has to be due to an axiom in tss and a substitution σ, thus, using the same axiom and substitution, we can prove the same transition in $tss \cup [[sc]]^*$.

For the induction step, if the transition $p \xrightarrow{l} p'$ in $tss \cup [[sc]]$ is due to a rule that is in $tss \cup [[sc]]^*$, as well, then according to the induction hypothesis, we can prove the premises of this rule from $tss \cup [[sc]]^*$ and since the rule is $tss \cup [[sc]]^*$, we can use the same rule and the same substitution to prove $p \xrightarrow{l} p'$.

It only remains to prove the induction step for the cases where the last rule is not in $tss \cup [[sc]]^*$, thus of the shape:

$$\frac{f(x_0,\ldots,x_{n-1}) \xrightarrow{l} y}{t \xrightarrow{l} y}$$

corresponding to a defining equation $f(x_0,\ldots,x_{n-1}) \equiv t$ and there exists a substitution σ such that $\sigma(t) = p$ and $\sigma(y) = p'$ and there exist closed terms p_i $(0 \leq i < n)$ such that $\sigma(x_i) = p_i$. The transition $f(p_0,\ldots,p_{n-1}) \xrightarrow{l} p'$ has a proof of depth $n-1$ and, according to the induction hypothesis, is provable from $tss \cup [[sc]]^*$. Consider the proof of this transition in $tss \cup [[sc]]^*$. Note that since $f(x_0,\ldots,x_{n-1}) \equiv t$ is a defining equation, f does not appear in the source of conclusion of any deduction rule in tss. Further, since f does not appear in any other equation, there is no rule in $[[sc]]^*$ with f in its source of conclusion, but the following rule.

$$\frac{t \xrightarrow{l} y}{f(x_0,\ldots,x_{n-1}) \xrightarrow{l} y}$$

Thus, the transition $f(p_0, \ldots, p_{n-1}) \xrightarrow{l} p'$ is due to the above rule and there exists a substitution σ' such that $\sigma'(x_i) = p_i$ $(0 \leq i < n)$ and $\sigma'(y) = p'$. But $vars(t) \subseteq \{x_i | 0 \leq i < n\}$, and $\sigma'(x_i) = \sigma(x_i) = p_i$ $(0 \leq i < n)$ thus, $\sigma'(t) = \sigma(t)$. Hence, $\sigma(t) \xrightarrow{l} p'$ has a proof in $tss \cup [[sc]]^*$.

This concludes the proof, as we have shown that $tss \cup [[sc]]$ is equal to $tss \cup [[sc]]^*$ and $tss \cup [[sc]]^*$ is in tyft. ⊠

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 congruence of bisimilarity. The counter-examples will be in such a way that the congruence is ruined according to all three interpretations. We start with a counter-example showing that variables in each side of the fx equation need to be distinct.

Example 5

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

Similar to Example 2, it clearly holds in the above specification that $a \leftrightarrow b$. However, it does not hold that $f(a, a) \leftrightarrow f(a, b)$ since the former can perform an l_0 transition, while the latter deadlocks.

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 2 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 function symbol on the side to be defined (i.e., only f in the left-hand-side of the equation $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$). In the following example, we show that allowing more function symbols also endangers congruence.

Example 6

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

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

The remaining constraints are on defining equations. First of all, for a defining equation $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$, variables x_i should all be distinct. We have already shown in Example 5 that relaxing this constraint may be harmful, for the only structural congruence equation satisfies both the definition of fx and defining equations. The other 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 7

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

Suppose that our common signature consists of a, b, c and d as constants and f as a unary function symbol. 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 in any other structural congruence equation, nor in the source of the conclusion of a deduction rule.

Example 8

$$c \equiv a \quad c \equiv g(b) \quad \textbf{(a)}\frac{}{a \xrightarrow{l_0} a} \quad \textbf{(b)}\frac{}{b \xrightarrow{l_0} a}$$

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

Example 9

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

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 structural congruences 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. For example, suppose that we want to define a priority operator θ that based on a priority function r (from labels to natural numbers) gives priority to actions and does not allow lower priority transitions to take over higher priority ones. The semantics of such an operator can be conveniently expressed by the following rule schema [4].

$$\frac{x \xrightarrow{l} y \quad \left\{x \overset{l'}{\nrightarrow} | r(l') > r(l)\right\}}{\theta(x) \xrightarrow{l} \theta(y)} (l \in L)$$

This and several other examples (e.g., deadlock detection, sequencing and urgency, cf. [9]) show that negative premises are useful additions to transition system specifications. Thus, it seems natural to extend transition system specifications in tyft format to account for negative premises. The following definition realizes this goal.

Definition 15 (Ntyft Format [16]) A rule is in ntyft format if and only if it has the following shape.

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

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.

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. In the context of SOS, this was first observed by [2, 8] and developed further in [16, 9, 15]. The first proposal catering for negative premises in the notion of proof is the supported model [8, 16] and is defined as follows.

Definition 16 (Supported Model [16]) Consider a transition system specification $tss = (\Sigma, L, Rel, D)$ and a closed formula $\psi \in C(\Sigma)$; the supported model of the transition system specification is a set of positive closed formulae $\mathcal{M}$, defined as follows.

$$\psi \in \mathcal{M} \iff \left(\exists_{d \in D} d = \frac{\{\chi_i | i \in I\}}{\chi} \wedge \exists_\sigma \forall_{i \in I} \mathcal{M} \vDash \sigma(\chi_i) \wedge \sigma(\chi) = \psi \right)$$

where $\mathcal{M} \vDash \psi$ for a positive formula $p \xrightarrow{l}_r p'$ means that $p \xrightarrow{l}_r p' \in \mathcal{M}$ and for a negative formula $p \overset{l}{\nrightarrow}_r$ means that there exists no $p' \in C(\Sigma)$ such that $(p \xrightarrow{l}_r p') \in \mathcal{M}$.

As illustrated in the remainder, it might be that for a transition system specification in ntyft format zero, one, or even more than one supported models exist. Using the interpretations presented in Section 4, one can use the notion of supported model for transition system specifications augmented with structural congruences. However, this may lead to strange phenomena as witnessed by the following example.

Example 10 Consider the following structural congruence equation, added to a transition system specification with the empty set of rules. Suppose that the common signature comprises of constants a and b and unary function symbols f and g.

$$g(x) \equiv f(a)$$

The above equation clearly satisfies the cfsc format as a defining equation and thus bisimilarity is congruence. According to the notion of provable transitions (Definitions 2 and 10) the above combination of the transition system specification and structural congruences induces an empty transition relation. However, in addition to this intuitive transition relation, the same combination has another supported model, as well, namely $\{f(a) \xrightarrow{l} a, g(a) \xrightarrow{l} a, g(b) \xrightarrow{l} a, g(f(a)) \xrightarrow{l} a, \ldots\}$ (for which bisimilarity is not a congruence).

The problem in the above example lies in the inherent cyclicity in the structural congruence rule or the corresponding interpretations. For example, in the third interpretation the following two rules are added to the transition system specification.

$$\frac{f(a) \xrightarrow{l} y}{g(x) \xrightarrow{l} y} \qquad \frac{g(x) \xrightarrow{l} y}{f(a) \xrightarrow{l} y}$$

This problem has been observed in the context of pure SOS specifications and led to a number of alternative interpretations or restrictions on the transitions system specification with negative premises. In particular, it is shown that if the transition system specification is (strictly) stratified, as defined below, it induces a (unique) transition relation (for which bisimilarity is a congruence).

Definition 17 (Stratification [16]) 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 in tss of the following form:

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

and for all closed substitutions σ, $\forall_{i \in I} \mathcal{S}(\sigma(t_i \xrightarrow{l_i}_{r_i} y_i)) \leq \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l}_r t'))$ and $\forall_{j \in J}$ $\mathcal{S}(\sigma(t_j \xrightarrow{l_j}_{r_j} t)) < \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l}_r t))$. A transition system specification is called *stratified* if and only if there exists a stratification function for it. If the measure decreases also from the conclusion to the positive premises, then the stratification is called strict.

If the stratification function maps all formulae that differ only in the target terms to the same ordinal, then the stratification is called *target-independent*. Similarly, one can speak about *label-independent* stratifications. A practical way of defining a stratification is by defining a size measure on terms appearing in the source of the transition (by counting the function symbols). Such a stratification function is clearly both target- and label-independent.

The following theorem from [16] formalizes the advantages of stratified transition system specifications.

Theorem 6 Consider a transition system specification tss in the ntyft format. If tss is stratified, then it has a supported model. If tss is strictly stratified, then the supported model is unique. Bisimilarity is a congruence for all supported models of a stratified transition system specification.

However, as later noted in [9], strict stratification is too much to ask for a unique transition relation. There are several examples of transition system specifications that intuitively induce a unique transition relation but cannot be strictly stratified. Example 10 and any other example in which instances of deduction rules may have a cyclic reference to each other share such a phenomena and were noted in the literature [9, 15]. By the same token, strict stratification in the presence of structural congruences seems hopeless due to their symmetric nature. Several alternative approaches to the notion of supported model and strict stratification have been proposed for which [15] provides an overview and a comparison. Here, we choose the notion of *stable model* [9, 15], defined below, that in our mind gives a reasonable and intuitive semantics for transition system specification with negative premises. The definition is slightly adapted to fit our notation and past definitions.

Definition 18 (Stable Model [9]) 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 an 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 exist a deduction rule $\dfrac{\{\chi_i \mid i \in I\}}{\chi}$ in tss (where χ_i can be a negative or a positive formula) and a substitution σ such that $\sigma(\chi) = \psi$, and for all $i \in I$, $\sigma(\chi_i) = \psi_i$;
- if the label of a node q, denoted by $p \overset{l}{\not\rightarrow}$, is a negative formula then there exists no p' such that $p \overset{l}{\rightarrow} p' \in T$.

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 $(T, tss) \vdash \phi$.

It is a straightforward extension to cater for the structural congruence relation in the above definition (as in Definition 10). Note that anomalies, such as those observed in Example 10, are resolved in the stable model interpretation. Particularly, the stable model of the transition system specification in Example 10 is now the intuitive empty set. The main reason for this is that the stable model requires a complete proof for positive formulae (as in Definition 2) rather than looking for a single matching deduction rule (as in Definition 16). The following theorem from [9] establishes that a stratified transition system specification uniquely defines a transition relation which is its stable model.

Theorem 7 Consider a transition system specification tss in the ntyft format. If tss is stratified, then it has a unique stable model. Furthermore, the stable model of the transition system specification is also a supported model.

From the above theorem and Theorem 6, it follows that bisimilarity with respect to the stable model of a stratified transition systems specification is a congruence.

Now, we have enough ingredients to study the implications of negative premises on the structural congruences. But 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 11

$$\textbf{(b)}\ \frac{a \overset{l_0}{\not\rightarrow}}{b \overset{l_0}{\rightarrow} b}$$

The above transition system specification (with a and b as constants), is strictly stratified by the function $\mathcal{S}$, if we define for all closed terms p, $\mathcal{S}(a \overset{l_0}{\rightarrow} p) \doteq 1$ and $\mathcal{S}(b \overset{l_0}{\rightarrow} p) \doteq 2$. Following Theorem 6, it defines the unique transition relation (its stable model), namely $\{b \overset{l_0}{\rightarrow} b\}$.

Suppose that we add the following structural congruence (which is indeed in the cfsc format) to the above transition system specification:

$$a \equiv b$$

Suddenly, the associated transition system specification looses its well-definedness. The combination of **(b)** and $a \equiv b$ leads to a contradiction since $b \overset{l_0}{\rightarrow} b$ if and only if $a \overset{l_0}{\not\rightarrow}$ and if $b \overset{l_0}{\rightarrow} b$ then $a \overset{l_0}{\not\rightarrow} b$.

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

Definition 19 (Stratification: Extended) Consider a transition system specification tss in ntyft format and structural congruence in the cfsc format. We call the combination of tss and sc stratified, if there exists a function $\mathcal{S}$ from closed formulae to an ordinal such that for all closed substitutions σ:

1. for all rules in tss of the following form:

$$\frac{\{t_i \overset{l_i}{\rightarrow}_{r_i} y_i | i \in I\} \quad \{t_j \overset{l_j}{\not\rightarrow}_{r_j} | j \in J\}}{f(x_0, \ldots, x_{ar(f)-1}) \overset{l}{\rightarrow}_r t}$$

   it holds that $\forall_{i \in I} \mathcal{S}(\sigma(t_i \overset{l_i}{\rightarrow}_{r_i} y_i)) \leq \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \overset{l}{\rightarrow}_r t'))$ and $\forall_{j \in J, t' \in T(\Sigma)}\ \mathcal{S}(\sigma(t_j \overset{l_j}{\rightarrow}_{r_j} t')) < \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \overset{l}{\rightarrow}_r t'))$,

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 $\forall_{l \in L, t \in T(\Sigma)}\ \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \overset{l}{\rightarrow}_r t)) = \mathcal{S}(\sigma(g(x_0, \ldots, x_{ar(g)-1}) \overset{l}{\rightarrow}_r t))$,

3. for all defining equations of the form $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$ in sc, it holds that $\forall_{l \in L, t' \in T(\Sigma)}\ \mathcal{S}(\sigma(t \overset{l}{\rightarrow}_r t')) \leq \mathcal{S}(\sigma(f(x_0, \ldots, x_{ar(f)-1}) \overset{l}{\rightarrow}_r t'))$.

The above definition is inspired by the structure of the transition system specification $tss \cup [[sc]]^*$ (Definition 14). In fact, a stratification function for $tss \cup [[sc]]^*$ precisely requires the above conditions to hold.

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 8 If the combination of transition system tss in ntyft format and structural congruences sc in cfsc is stratified, then $tss \cup [[sc]]$ has a unique stable model.

Proof. Consider the transition system specification $tss \cup [[sc]]^*$, it trivially follows from the hypotheses that it is in ntyft format and stratified. Thus, according to Theorem 6, $tss \cup [[sc]]^*$ has a unique stable model. If we also show that the stable models of $tss \cup [[sc]]^*$ and $tss \cup [[sc]]$ coincide, then the thesis follows. This follows from the following lemma.

Lemma 8 Consider a set of positive closed formulae T (on the common signature Σ). For all closed terms $p, p' \in C(\Sigma)$ and label $l \in L$ then the following statement holds:

$$(T, tss \cup [[sc]]^*) \vdash p \xrightarrow{l} p' \Leftrightarrow (T, tss \cup [[sc]]) \vdash p \xrightarrow{l} p'$$

Proof. We divide this into the following two implications:

1. $(T, tss \cup [[sc]]^*) \vdash p \xrightarrow{l} p' \Rightarrow (T, tss \cup [[sc]]) \vdash p \xrightarrow{l} p'$

   This holds trivially since the deduction rules of $tss \cup [[sc]]^*$ are all in $tss \cup [[sc]]$ and thus the proof for $p \xrightarrow{l} p'$ in $tss \cup [[sc]]^*$ is still valid in $tss \cup [[sc]]$.

2. $(T, tss \cup [[sc]]) \vdash p \xrightarrow{l} p' \Rightarrow (T, tss \cup [[sc]]^*) \vdash p \xrightarrow{l} p'$

   We prove this by an induction on the depth of the proof tree for $(T, tss \cup [[sc]]) \vdash p \xrightarrow{l} p'$.

   For the induction basis, if the proof is of depth 1, then it is due to a rule that is also in $tss \cup [[sc]]^*$ and a substitution σ (rules in $[[sc]] \setminus [[sc]]^*$ cannot be used in proof of depth 1 as they have a positive formula in the premise which needs a proof). Using the same rule and the same substitution we can prove this transition from $(T, tss \cup [[sc]])$.

   For the induction step, suppose that the statement holds for closed positive formulae with a proof of depth $n-1$ or less and suppose that $(T, tss \cup [[sc]]) \vdash p \xrightarrow{l} p'$ has a proof of depth n. Then, either the last rule is in $tss \cup [[sc]]^*$, as well, from which, using the induction hypothesis on the premises, we can prove that $(T, tss \cup [[sc]]) \vdash p \xrightarrow{l} p'$, or the last rule in the proof structure is in $[[sc]] \setminus [[sc]]^*$. Then, the deduction rule should be of the following form:

   $$\frac{f(x_0, \ldots, x_{n-1}) \xrightarrow{l} y}{t \xrightarrow{l} y}$$

   for an n-ary function symbol f and there exists a substitution σ such that $\sigma(t) = p$, $\sigma(y) = p'$ and there exists a defining equation $f(x_0, \ldots, x_{n-1}) \equiv t$ in the structural congruences. Since $(T, tss \cup [[sc]]) \vdash p \xrightarrow{l} p'$, there should exist a deduction rule such that $\sigma(f(x_0, \ldots, x_{n-1})) \xrightarrow{l} p'$ is provable from $(T, tss \cup [[sc]])$. Since equation $f(x_0, \ldots, x_{n-1}) \equiv t$ is defining, there is no rule in tss with f appearing in the source of its conclusion (and there is no other equation in sc in which f appears). Thus, the only option for providing a proof for $\sigma(f(x_0, \ldots, x_{n-1})) \xrightarrow{l} p'$ is a deduction rule of the following shape

   $$\frac{t \xrightarrow{l} y}{f(x_0, \ldots, x_{n-1}) \xrightarrow{l} y}$$

   and a substitution σ' such that $\sigma'(x_i) = \sigma(x_i)$ (for all $0 \leq i < n$) and $\sigma(y) = p'$. On one hand, $\sigma'(t) \xrightarrow{l} p'$ is a positive formula, it should have a proof depth less than $n-1$ and thus it follows from the induction hypothesis that $(T, tss \cup [[sc]]^*) \vdash \sigma'(t) \xrightarrow{l} p'$. On the other hand, $vars(t) \subseteq \{x_i | 0 \leq i < n\}$ and thus, $\sigma'(t) = \sigma(t)$, thus, $(T, tss \cup [[sc]]^*) \vdash \sigma(t) \xrightarrow{l} p'$ and hence $(T, tss \cup [[sc]]^*) \vdash p' \xrightarrow{l} p'$.

⊠

Suppose that T is a stable model of $tss \cup [[sc]]^*$. Then it follows from Definition 18 that for all closed formula ϕ, $\phi \in T$ if and only if $(T, tss \cup [[sc]]^*) \vdash \phi$ and then from the above lemma that $\phi \in T$ if and only if $(T, tss \cup [[sc]]) \vdash \phi$. Thus, T is a stable model of $tss \cup [[sc]]$. The reasoning holds in the reverse direction, as well, and thus, the stable models of $tss \cup [[sc]]$ and $tss \cup [[sc]]^*$ coincide.

⊠

We do not intend to extend all the results of Section 4 to specifications with negative premises. However, it can be checked that, similar to the above case (for the coincidence of $tss \cup [[sc]]$ and $tss \cup [[sc]]^*$), all the results of Section 4 hold for transition system specifications with negative premises with the additional necessary condition of being stratified. The proofs of the above mentioned results then only need a mere change of notation from $tss \vdash \phi$ to $(T, tss) \vdash \phi$.

Possible extensions to ntyft format are the addition of ntyxt rules and predicates. The ntyft-ntyxt format is a relaxation of ntyft format that allows for variables in the source of the conclusion. In [16], it is shown how to reduce ntyft-ntyxt format to ntyft format. Adding structural congruences to transition system specifications 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 Section 7, we show that if defining equations are only meant to define behavior of new operators (and not influence the behavior of existing ones), they can be safely added to transition system specifications in ntyft-ntyxt format (after a syntactic transformation for making them *pure*) and the result of this addition is an operationally conservative extension of the original semantics.

Predicates are other ingredients of transition system specifications that are used to specify concepts such as termination and divergence on process terms [36]. 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 extend to the PANTH format of [36] which allows for both ntyft-ntyxt rules and predicates. There remains one problem that needs more attention and that is the problem of adding defining equations to transition system specifications with tyxt rules. This problem is addressed in the next section.

7 Structural Congruences and Conservativity

Programming languages and process calculi have been subject to constant extensions. It is often crucial to make sure that such extensions do not change the behavior of the old subset, or said otherwise, the extension is *conservative*. This topic has been touched upon in [16, 17] and studied in depth in [1, 14, 35, 37]. In this section, we study the conservative extension of operational semantics in the presence of structural congruences. In our study, we mainly use the results from [35].

To extend a transition system specification, we may have to combine an existing signature with a new signature. However, not all signatures can be combined into one as the arities of the function symbols may clash. To prevent this we define the notion of consistent signatures, as follows.

Definition 20 (Consistent Signatures) Two signatures Σ_0 and Σ_1 are consistent if and only if they agree on the arity of the shared function symbols.

In the traditional settings (with SOS rules only), the (conservative) extension of a transition system specification with another one is defined as follows.

Definition 21 (Extending Transition System Specifications) Consider transition system specifications $tss_0 = (\Sigma_0, L_0, D_0, Rel_0)$ and $tss_1 = (\Sigma_1, L_1, D_1, Rel_1)$. Suppose that Σ_0 and Σ_1 are consistent. Extension of tss_0 with tss_1, denoted by $tss_0 \cup tss_1$, is defined as $(\Sigma_0 \cup \Sigma_1, L_0 \cup L_1, Rel_0 \cup Rel_1, D_0 \cup D_1)$.

Definition 22 (Operational Conservative Extension [35]) Consider transition system specifications $tss_0 = (\Sigma_0, L_0, Rel_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, Rel_0, D_1)$. If $\Sigma_0 \subseteq \Sigma_1$, $L_0 \subseteq L_1$, $Rel_0 \subseteq Rel_1$ and $D_0 \subseteq D_1$ and for all closed positive formulae $p \xrightarrow{l}_r p'$ with $p \in C(\Sigma_0)$, $p' \in C(\Sigma_1)$, $r \in Rel_1$ and $l \in L_1$, $tss_1 \vdash p \xrightarrow{l}_r p'$ if and only if $tss_0 \vdash p \xrightarrow{l}_r p'$, then tss_1 is an operational conservative extension of tss_0.

Note that this is not the only possible way of defining operational conservativity; it is indeed the common notion of operational conservativity studied in the literature, though. As an alternative definition, p', r and l can be taken from Σ_0, Rel_0 and L_0, respectively. This alternative definition is weaker than the one stated above and the sufficient conditions given in the remainder implies conservativity in the alternative sense, as well. As defined above, the operational conservative extension is a reflexive, anti-symmetric and transitive relation.

Extension of a transition system specification with another transition system specification (both in ntyft format) is not always conservative. The following example illustrates a few issues that may go wrong during the extension.

Example 12 Consider the following transition system specification, called tss_0, defined on the signature containing a and b as constants and f as a unary function symbol and l_0 as the only label.

$$\textbf{(a)}\ \frac{}{a \xrightarrow{l_0} a} \qquad \textbf{(f)}\ \frac{}{f(x) \xrightarrow{l_0} y}$$

Also, consider the following transition system specification, called tss_1, defined on the signature with a and c as constants and g as a unary function symbol and $\{l_0, l_1\}$ as the set of labels.

$$\textbf{(g)}\ \frac{x \xrightarrow{l_0}}{g(x) \xrightarrow{l_1} x}$$

Neither of the two extensions of tss_0 with tss_1, nor tss_1 with tss_0 are conservative. If we extend tss_0 with tss_1, then we get $tss_0 \cup tss_1 \vdash f(a) \xrightarrow{l_0} c$ while such a transition cannot be derived from tss_0 alone (as c is not the signature Σ_0). Similarly, transition $g(a) \xrightarrow{l_1} a$ is not derivable from $tss_1 \cup tss_0$ while it is derivable from tss_1.

The first extension is not conservative, since rule **(f)** allows for an open choice in variable y. Thus, when we extend tss_0 by tss_1, y can freely range over terms in signature $\Sigma_0 \cup \Sigma_1$ and thus harm conservativity. This is prevented by the notion of *pure* ntyft rules [16]. The second extension is not conservative since tss_0 defines behavior of term a which is in signature Σ_1, as well. This is prevented by restricting the function symbols being defined by the added transition system specifications. Next, we define the notion of pure ntyft format as motivated before.

Definition 23 (The pure ntyft Format) A rule is in the pure ntyft format if and only if it is in the ntyft format and only contains variables that appear in the source of the conclusion and target of the premises. A transition system specification is in the pure ntyft format if and only if all its rules are.

Using the notion of pure ntyft format, the sufficient criteria for operational conservativity are quoted below.

Theorem 9 (Operational Conservativity Theorem [35]) Consider transition system specifications $tss_0 = (\Sigma_0, L_0, D_0, Rel_0)$ and $tss_1 = (\Sigma_1,\ L_1,\ D_1, Rel_1)$. Suppose that Σ_0 and Σ_1 are consistent. Transition system specification $tss_0 \cup tss_1$ is a conservative extension of tss_0 if:

1. tss_0 and tss_1 are both stratified and in the pure ntyft and ntyft formats, respectively;
2. for all deduction rules in D_1 of the following form

$$\frac{\{t_i \xrightarrow{l_i}_{r_i} y_i | i \in I\} \quad \{t_j \not\xrightarrow{l_j}_{r_j} | j \in J\}}{f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l}_r t}$$

   either $f \notin \Sigma_0$ or there exists an $i \in I$ such that $l_i \notin L_0$ and $t_i \in T(\Sigma_0)$.

As we have given an interpretation of structural congruences inside the transition system specification framework, it is straightforward to check whether extending a combination of a transition system specification and structural congruences with another transition system specification is operationally conservative. It can also be easily checked that adding defining equations to a transition system specification may be interpreted as a conservative extension. The following lemma states this fact.

Lemma 9 Consider a transition system specification $tss_0 = (\Sigma_0, L_0, D_0, Rel_0)$ which is in pure ntyft format and structural congruences sc on signature Σ_1. Suppose that $\Sigma_0 \subseteq \Sigma_1$ and sc only contains equations of the form $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$, where $t \in T(\Sigma_1)$, $f \in \Sigma_1 \setminus \Sigma_0$ and $vars(t) \subseteq \{x_i | 0 \leq i < ar(f)\}$. Define $tss_0' \doteq (\Sigma_1, L_0, D_0, Rel_0)$. Then, if $tss_0' \cup [[sc]]$ is stratified, it is a conservative extension of tss_0.

Proof. On one hand, we have already shown that the transition relation induced by $tss_0' \cup [[sc]]$ is the same as the one induced by $tss_0' \cup [[sc]]^*$. On the other hand, $tss_0' \cup [[sc]]^*$ satisfies the conditions of Theorem 9 as a conservative extension of tss_0'. Further, following from the same theorem, tss_0' is a conservative extension of tss_0 and since the notion of operational conservative extension is transitive, $tss_0' \cup [[sc]]^*$ is an operational conservative extension of tss_0. ⊠

Using the above lemma, we can extend transition system specifications with ntyxt rules by adding defining equations. Consider an ntyxt rule in a transition system specification tss of the following form:

$$\frac{\{t_i \xrightarrow{l_i} y_i | i \in I\} \quad \{t_j \not\xrightarrow{l_j} | j \in J\}}{x \xrightarrow{l} y}$$

defined on the signature Σ and a set of defining structural congruences sc with the set Σ_1 of defined function symbols. Suppose that defining equations are only meant to specify the behavior of defined function symbols and have no further influence on the semantics. If we copy the above deduction rule for all $f \in (\Sigma_0 \setminus \Sigma_1)$ as follows

$$\frac{\{t_i[f(x_0, \ldots, x_{ar(f)-1})/x] \xrightarrow{l_i} y_i | i \in I\} \quad \{t_j[f(x_0, \ldots, x_{ar(f)-1})/x] \not\xrightarrow{l_j} | j \in J\}}{f(x_0, \ldots, x_{ar(f)-1}) \xrightarrow{l} y[f(x_0, \ldots, x_{ar(f)-1})/x]},$$

then the above transition system specification is in ntyft format and for terms not containing $g \in \Sigma_1$ induces the same transitions as tss (following a simple induction on the depth of the proof for such transitions). Following Lemma 9, adding structural congruences is then a conservative extension and the resulting transition system specification is in ntyft again. Thus, we can re-use the congruence theorems stated before in the setting with ntyxt rules, as well.

Following the same style as of Lemma 9, the conservative extension results can easily be generalized to extend a combination of a transition system specification and structural congruences with another combination.

Another important issue in this area, called *equational conservativity*, is concerned with the equational theory of the extended languages. Informally, equational conservativity means that provable equalities of the original language should be respected by the extension of the equational theory. There are two main results to be mentioned in this regard: First, if an extension is operationally conservative, then a sound extension of the equational theory of the extended language is conservative. The second result states that if the original language has a sound and complete equational theory and the conservative extension is provided with a set of eliminating sound axioms for the new function symbols, then the combination has a sound and complete axiomatization, as well. Before quoting the equational conservativity theorem, we fix some basic definition regarding equational theories and their extensions.

Definition 24 (Equational Theory) A set of equalities E on a signature Σ of the form $t = t'$, where $t, t' \in T(\Sigma)$ is called an equational theory or axiomatization over Σ. A closed instance $p = p'$ is derivable from E, denoted by $E \vdash p = p'$ if it is in the smallest equality and congruence relation on closed terms induced by E. An equational theory E on Σ is sound with respect to a transition system specification tss (also on signature Σ) if for all $p, p' \in C(\Sigma)$ such that $E \vdash p = p'$, it holds that $tss \vdash p \leftrightarrows p'$. It is complete if the implication holds in the other direction.

An equational theory E on Σ is eliminating function symbols from $\Sigma' \subseteq \Sigma$ if and only if for all $p \in C(\Sigma)$ there exists a term $p' \in C(\Sigma \setminus \Sigma')$ such that $E \vdash p = p'$.

Definition 25 (Equational Conservative Extension [35]) An equational theory E_1 on signature Σ_1 is an equational conservative extension of E_0 on Σ_0 if and only if $\Sigma_0 \subseteq \Sigma_1$, $E_0 \subseteq E_1$ and for all $p, p' \in C(\Sigma_0)$, $E_0 \vdash p = p' \Leftrightarrow E_1 \vdash p = p'$.

Theorem 10 (Equational Conservativity Theorem [35]) Consider transition system specifications $tss_0 = (\Sigma_0, L_0, Rel_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, Rel_1, D_1)$ where tss_1 is an operational conservative extension of tss_0. Also let E_0 on Σ_0 be a sound and complete equational theory of tss_0 and E_1 on Σ_1 (with $E_0 \subseteq E_1$) a sound axiomatization of tss_1. Then, E_1 is an equational conservative extension of E_0. Furthermore, if E_1 is eliminating function symbols from $\Sigma_1 \setminus \Sigma_0$ then E_1 is a complete axiomatization of tss_1.

Following the same approach as of Lemma 9, we extend the above theorem to equational extension of a transition system specification with structural congruences.

Lemma 10 Consider a transition system specification $tss = (\Sigma, L, D, Rel)$ which is in pure ntyft format and structural congruences sc on signature Σ'. Suppose that $\Sigma \subseteq \Sigma'$ and sc only contains of the form $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$, where $t \in T(\Sigma')$, $f \in \Sigma' \setminus \Sigma$ and $vars(t) \subseteq \{x_i | 0 \leq i < ar(f)\}$. Suppose that E is a sound an complete axiomatization of tss. Then, $E' \doteq E \cup \{f(x_0, \ldots, x_{ar(f)-1}) = t \mid (f(x_0, \ldots, x_{ar(f)-1}) \equiv t) \in sc\}$ is an equational conservative extension of E. Furthermore, if in all equations of the form $f(x_0, \ldots, x_{ar(f)-1}) \equiv t$, $t \in T(\Sigma_0)$, then E' is a sound and complete axiomatization for $tss' \cup [[sc]]$, where $tss' = (\Sigma_1, L_0, D_0, Rel_0)$.

Proof. In Lemma 9, we have already shown that $tss' \cup [[sc]]$ is an operational conservative extension of tss. Thus, to apply Theorem 10, we only have to prove that E' is a sound axiomatization for $tss' \cup [[sc]]$. This is an straightforward consequence of Lemma 5. ⊠

The above lemma shows that in certain cases, when structural congruences are used to extend a language the same equations can be used as axioms of the equational theory and even guarantee the completeness of the extended equational theory. In the above lemma, when $t \in T(\Sigma_0)$, then the defining equation has the shape as the *explicit definitions* of [3].

8 Case Study

In this section, we quote an SOS semantics of CCS from [22] (with restriction to finite sum and introduction of replication operator) and then introduce structural congruences, à la [23], 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 $\overline{l}$. By extending the same notation, let $\overline{\overline{l}}$ 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 in Figure 2. In this 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. A particular label τ is added for inactions resulting from communication and $\overline{\tau}$ is defined as τ. 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 [23] for the π-calculus and thus, eliminate some of the deduction rules. The result is shown in Figure 3.

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 5 that strong bisimilarity is a congruence with respect to the induced transition relation. This can already be considered an achievement. However, one may argue that we could not specify some, may be more interesting, structural congruences of [23] 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 first, in this particular case, all of these properties can be proven from the above specification as theorems and second, there are cases where 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 13 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 below (note that the semantic rule conforms to tyft format):

$$\textbf{(LMer)}\ \frac{x_0 \xrightarrow{\alpha} y_0}{x_0 \bullet x_1 \xrightarrow{\alpha} y_0 \mid\mid x_1}$$

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

$\textbf{(Act)}\dfrac{}{\alpha.x \xrightarrow{\alpha} x}$	$\textbf{(Res)}\dfrac{x \xrightarrow{\alpha} y}{x \setminus L \xrightarrow{\alpha} y \setminus L}(\alpha, \overline{\alpha} \notin L)$
$\textbf{(Sum0)}\dfrac{x_0 \xrightarrow{\alpha} y}{x_0 + x_1 \xrightarrow{\alpha} y}$ $\quad$ $\textbf{(Sum1)}\dfrac{x_1 \xrightarrow{\alpha} y}{x_0 + x_1 \xrightarrow{\alpha} y}$	
$\textbf{(Com0)}\dfrac{x_0 \xrightarrow{\alpha} y_0}{x_0 \mid\mid x_1 \xrightarrow{\alpha} y_0 \mid\mid x_1}$ $\quad$ $\textbf{(Com1)}\dfrac{x_1 \xrightarrow{\alpha} y_1}{x_0 \mid\mid x_1 \xrightarrow{\alpha} x_0 \mid\mid y_1}$ $\quad$ $\textbf{(Com2)}\dfrac{x_0 \xrightarrow{l} y_0 \quad x_1 \xrightarrow{\overline{l}} y_1}{x_0 \mid\mid x_1 \xrightarrow{\tau} y_0 \mid\mid y_1}$	
$\textbf{(Con)}\dfrac{t \xrightarrow{\alpha} y}{A \xrightarrow{\alpha} y}(A \doteq t)$	$\textbf{(Rep)}\dfrac{x \mid\mid !x \xrightarrow{\alpha} y}{!x \xrightarrow{\alpha} y}$
$l, \overline{l} \in \mathcal{L}, \alpha \in \mathcal{L} \cup \{\tau\}$	

Figure 2: Semantics of CCS: SOS rules

$\textbf{(Act)}\dfrac{}{\alpha.x \xrightarrow{\alpha} x}$	$\textbf{(Res)}\dfrac{x \xrightarrow{\alpha} y}{x \setminus L \xrightarrow{\alpha} y \setminus L}(\alpha, \overline{\alpha} \notin L)$	
$\textbf{(NSum0)}\dfrac{x_0 \xrightarrow{\alpha} y}{x_0 + x_1 \xrightarrow{\alpha} y}$		
$\textbf{(NCom0)}\dfrac{x_0 \xrightarrow{\alpha} y_0}{x_0 \mid\mid x_1 \xrightarrow{\alpha} y_0 \mid\mid x_1}$ $\quad$ $\textbf{(NCom1)}\dfrac{x_0 \xrightarrow{l} y_0 \quad x_1 \xrightarrow{\overline{l}} y_1}{x_0 \mid\mid x_1 \xrightarrow{\tau} y_0 \mid\mid y_1}$		
$\textbf{(struct)}\dfrac{x \equiv y \quad y \xrightarrow{l} y' \quad y' \equiv x'}{x \xrightarrow{l} x'}(l \in L)$	$x_0 + x_1 \equiv x_1 + x_0$ $A \equiv t \quad (A \doteq t)$	$x_0 \mid\mid x_1 \equiv x_1 \mid\mid x_0$ $!x \equiv x \mid\mid !x$
$l, \overline{l} \in \mathcal{L}, \alpha \in \mathcal{L} \cup \{\tau\}$		

Figure 3: Semantics of CCS: SOS rules with Structural Congruences

is similar to the left-merge operator of [4] 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 following equation to our set of structural congruences, to make it associative.

$$x_0 \bullet (x_1 \bullet x_2) \equiv (x_0 \bullet x_1) \bullet x_2$$

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 rule). While the right-hand-term is congruent to $(\alpha \bullet 0) \bullet \alpha$ and this new term can perform two consecutive α actions after the first of which it turns into $(0 \parallel 0) \parallel \alpha$.

9 Conclusions

In this paper, we presented a number of ways to interpret structural congruences inside the transition system specification framework and compared the outcomes formally. We also defined a syntactic format for structural congruences that makes them safe with respect to the congruence of strong bisimilarity, once they are used in combination with a set of standard (e.g., tyft) SOS rules. To allow for negative premises in the transition system specifications, the relationship between negative premises in the deduction rules, structural congruences and well-definedness of the transition relation was investigated and sufficient well-definedness criteria were established. Furthermore, the extension of operational and equational conservativity theorems to the setting with structural congruences were studied. 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 (following the approach of other standard formats for weaker notions of bisimulation, e.g., RBB format of [7]). Studying structural congruences in the bi-algebraic framework of [34] may lead to a foundational framework for this mixed settings, as well.

References

[1] Luca Aceto, Wan J. Fokkink, and Chris Verhoef. Structural operational semantics. In Jan A. Bergstra, Alban Ponse, and Scott A. Smolka, editors, *Handbook of Process Algebra, Chapter 3*, pages 197–292. Elsevier Science, Dordrecht, The Netherlands, 2001.

[2] Jos C. M. Baeten and Jan A. Bergstra. Processen en procesexpressies. *Informatie*, 30(3):177–248, 1988. (in Dutch).

[3] Jos C. M. Baeten and Cornelis A. Middelburg. *Process Algebra with Timing*. EATCS Monographs. Springer-Verlag, Berlin, Germany, 2002.

[4] Jos C. M. Baeten and W. Peter Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambrdige University Press, 1990.

[5] Jean-Pierre Banâtre, Pascal Fradet, and Daniel Le Métayer. Gamma and the chemical reaction model: Fifteen years after. In Cristian S. Calude, Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa, editors, *Multiset Processing: Mathematical, Computer Science, and Molecular Computing Points of View*, volume 2235 of *Lecture Notes in Computer Science*, pages 17–44. Springer-Verlag, Berlin, Germany, 2001.

[6] Gérard Berry and Gérard Boudol. The chemical abstract machine. *Theoretical Computer Science (TCS)*, 96:217–248, 1992.

[7] Bard Bloom. Structural operational semantics for weak bisimulations. *Theoretical Computer Science (TCS)*, 146:25–68, 1995.

[8] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *Journal of the ACM (JACM)*, 42(1):232–268, January 1995.

[9] Roland Bol and Jan Friso Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM (JACM)*, 43(5):863–914, September 1996.

[10] Luca Cardelli and Andrew D. Gordon. Mobile ambients. *Theoretical Computer Science (TCS)*, 240(1):177–213, 2000.

[11] Michel R. V. Chaudron and Edwin de Jong. Schedules for multiset transformer programs. In *Coordination Programming: Mechanisms, Models and Semantics*, pages 195–210. Imperial College Press, London, UK, 1996.

[12] Joost Engelfriet and Tjalling Galsema. Multisets and structutal congruence of pi-calculus with replication. *Acta Informatica*, 211(2):311–337, 2004.

[13] Wan J. Fokkink and Rob J. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation (I&C)*, 126(1):1–10, 1996.

[14] Wan J. Fokkink and Chris Verhoef. A conservative look at operational semantics with variable binding. *Information and Computation (I & C)*, 146(1):24–54, 1998.

[15] Rob J. van Glabbeek. The meaning of negative premises in transition system specifications II. *Journal of Logic and Algebraic Programming (JLAP)*, 60-61:229–258, 2004.

[16] Jan Friso Groote. Transition system specifications with negative premises. *Theoretical Computer Science (TCS)*, 118(2):263–299, 1993.

[17] Jan Friso Groote and Frits W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation (I&C)*, 100(2):202–260, October 1992.

[18] Kohei Honda and Nobuko Yoshida. On reduction-based process semantics. *Theoretical Computer Science (TCS)*, (2):437–486, 1995.

[19] James J. Leifer. *Operational Congruences for Reactive Systems*. PhD thesis, Computer Laboratory, University of Cambridge, Cambridge, UK, 2001.

[20] James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In Catuscia Palamidessi, editor, *Proceedings of 11th International Conference on Concurrency Theory (CONCUR'00)*, volume 1877 of *Lecture Notes in Computer Science*, pages 259–274. Springer-Verlag, Berlin, Germany, 2000.

[21] Robin Milner. Flowgraphs and flow algebras. *Journal of the ACM (JACM)*, 26(2):794–818, 1979.

[22] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[23] Robin Milner. Functions as processes. *Mathematical Structures in Computer Science*, 2:119–141, 1992. An earlier version of this paper appeared as Technical Report N.1154 of INRIA, Sophia-Antipolis, 1990.

[24] Robin Milner. The polyadic π-calculus: a tutorial. In Friedrich L. Bauer, Wilfried Brauer, and Helmut Schwichtenberg, editors, *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1993. An earlier version of this paper appeared as Technical Report ECS-LFCS-91-180 of University of Edinburgh, 1991.

[25] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part II. *Information and Computation (I&C)*, 100(1):41–77, 1992. An earlier version of this paper appeared as Technical Report ECS-LFCS-89-86 of University of Edinburgh, 1989.

[26] Robin Milner and Davide Sangiorgi. Barbed bisimulation. In Werner Kuich, editor, *Proceedings of 19th International Colloquium on Automata, Languages and Programming (ICALP'92)*, volume 623 of *Lecture Notes in Computer Science*, pages 85–695. Springer-Verlag, Berlin, Germany, 1992.

[27] David M.R. Park. Concurrency and automata on infinite sequences. In *Proceedings of 5th GI Conference*, volume 104 of *Lecture Notes in Coputer Science*, pages 167–183. Springer-Verlag, Berlin, Germany, 1981.

[28] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, Aarhus, Denmark, September 1981.

[29] Gordon D. Plotkin. The origins of structural operational semantics. *Journal of Logic and Algebraic Programming (JLAP)*, 60:3–15, 2004.

[30] Gordon D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Progamming (JLAP)*, 60:17–139, 2004. An earlier version of this aritcle appeared as [28].

[31] Vladimiro Sassone and Paweł Sobociński. Deriving bisimulation congruences: 2-categories vs precategories. In Andrew Gordon, editor, *Proceedings of 6th International Conference on Foundations of Software Science and Computation Structures (FOSSACS '03)*, volume 2620 of *Lecture Notes in Computer Science*, pages 409–424. Springer-Verlag, Berlin, Germany, 2003.

[32] Peter Sewell. From rewrite rules to bisimulation congruences. *Theoretical Computer Science (TCS)*, 274(1-2):183–230, 2002.

[33] Robert de Simone. Higher-level synchronizing devices in MEIJE-SCCS. *Theoretical Computer Science (TCS)*, 37:245–267, 1985.

[34] Daniele Turi and Gordon D. Plotkin. Towards a mathematical operational semantics. In *Proceedings of 12th Annual IEEE Symposium on Logic in computer Science (LICS'97)*, pages 280–291. IEEE Computer Society Press, 1997.

[35] Chris Verhoef. A general conservative extension theorem in process algebra. In Ernst-Rüdiger Olderog, editor, *Proceedings of third IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET'94)*, volume A-56 of *IFIP Transactions*, pages 274–302. Elsevier Science Publishers, 1994.

[36] Chris Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274–302, 1995.

[37] Chris Verhoef, Luca Aceto, and Wan Fokkink. Conservative extension in structural operational semantics. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 69:110–132, 1999.