Algebraic results for structured operational semantics

Vashti Galpin

Department of Computer Science, University of the Witwatersrand
`vashti@cs.wits.ac.za`
`http://www.cs.wits.ac.za/˜vashti`

Abstract

This paper presents algebraic results that are important for the extended tyft/tyxt *format [12, 13] which can be used to describe many different process algebras. This format is based on a many-sorted signature which permits both processes and labels to be treated syntactically. Existing results for this format permit the comparison of process algebra semantic equivalences by forming the sum of two transition system specifications and imposing certain conditions. The results presented in this paper involve the summing of congruences that model the actual process algebra labels, and determine under what conditions these congruences have important properties such as compatibility and conservativity. The aim of this paper is to show that the notion of sort-similarity on the sum of signatures is sufficient for the sum of the congruences induced by each label algebra to be the same as the congruence induced by the summed label algebras. Additionally, sort-similarity is sufficient for compatibility and conservativity when summing. Finally, conditions on the label algebra are given that ensure compatibility.*

Keywords: *extended* tyft/tyxt *format, many-sorted, comparison of semantic equivalences, sort-similar, process algebra, bisimulation, operational semantics*
Computing Review Categories: *D.3.1, F.1.2, F.3.2, F.4.3*

1 Introduction

This paper focusses on algebraic results which are important for the metatheory of process algebras; namely the theory that considers the general form of rules that make up the structured operational semantics [19] that are used to define process algebras.

Process algebras provide mathematical models of concurrent computations. A process algebra, in the sense of CCS (Calculus of Communicating Systems) [18], consists of three major components

Syntax This provides a language to describe concurrent processes and comprises a number of operators and actions (labels). These operators represent parallelism, communication, non-determinism, hiding, etc; and the actions represent the visible behaviour of the processes. The syntax is usually expressed in BNF, but can also be expressed by the signature of an algebra. Processes can be constructed from the BNF grammar or described by the term algebra of the signature. For example, $a.b.0 + b.a.0$ is the process that has a choice between performing an a action followed by a b action, or a b action followed by an a action, and $a.0 \mid b.0$ is the process that can perform an a action and a b action in any order.

Operational semantics The behaviour of processes is represented by a labelled transition system. Structured operational semantics given in the form of rules, describe which actions can occur for each operator. These rules are used to construct a proof that a particular transition can occur in the labelled transition system. This proof involves the deduction of a new transition from existing transitions. A transition such as $a.b.0 + b.a.0 \xrightarrow{a} b.0$ describes the ability of the process to perform the action a and become the process that can perform a b action. The process $a.0 \mid b.0$ can perform a b action and become the process that can only perform an a action; this is described by the transition $a.0 \mid b.0 \xrightarrow{b} a.0 \mid 0$.

Semantic equivalences These are equivalence relations (which can be congruences with respect to the operators of the process algebra) which relate processes that have the same behaviour, for some notion of behaviour. An example of an equivalence is bisimulation [18] which requires that a transition by one process can be matched by a similar transition by the other process and *vice versa*, and that the resulting processes have this same property. Bisimulation equates $a.0 \mid b.0$ and $a.b.0 + b.a.0$.

Many process algebras have been proposed to model with different aspects of concurrent behaviour – some examples are CCS with locations [6], CCS with local and global causes [15], CCS with causalities [8], multiprocessor CCS [16] and pomset CCS [7].

A description of the form taken by the rules of the operational semantics is called a format, and a collection of operators and rules in a specific format is called a transition system specification. Formats provide a general way to express process algebras and to reason about them. A number of formats have been defined [1, 2, 3, 4, 5, 9, 10, 11, 14, 17, 20, 21]. These focus on different aspects of process algebras such as congruence of bisimulation, negative transitions, predicates and conservative extensions.

Recently, the extended *tyft/tyxt* format has been proposed to permit the comparison of process algebra semantic equivalences [12, 13]. This format extends the *tyft/tyxt* format [14]. The motivation for this research is the development of many distinct process algebras to model different aspects of concurrent computation. This proliferation indicates that the notion of process algebra has wide application and is flexible; however, it is often not obvious how a process algebra and its semantic equivalences relate to other process algebras and equivalences. It is important both theoretically and practically to understand these relationships.

The extended *tyft/tyxt* format differs from most other formats because it is based around a many-sorted signature Σ as opposed to a single-sorted signature. Generally, with a single sorted signature, processes are treated syntactically and labels are treated schematically. A many-sorted signature permits the label terms of the process algebra to be treated in the same syntactic manner as the process terms. This allows for the expression of more complex process algebras and caters for the use of bisimulation-type equivalences where exact matching on transitions is not required. This may occur either through the definition of the bisimulation itself or by the way in which the actual process algebra labels are modelled.

As the notion of a format is a syntactic one, processes and labels are represented by elements of the term algebra (those terms that can be created from a set of variables and the operators of the signature). However, the label terms may need to be interpreted as the syntactic form may make unnecessary distinctions. Hence the actual labels of the process algebra expressed in the extended *tyft/tyxt* format are represented as terms in a Σ-algebra. Since there is a unique homomorphism from the term algebra to any Σ-algebra which induces a congruence on the elements of the term algebra, this congruence is then used to match labels in the definition of bisimulation.

The major results relating to the extended *tyft/tyxt* format are that bisimulation is a congruence for operators defined in this format when certain properties hold, and when one extended transition system specification is summed with another, and specific conditions hold, then it can be determined when the bisimulation over the summed systems equates more processes or fewer than the bisimulation on the original systems. This permits the comparison of semantic equivalences.

In this paper, the focus is on what happens with the congruences induced by the labels when summing two extended transition system specifications. Each extended transition system specification has a congruence over the label terms induced by an algebra associated with these labels. These congruences are summed for the results mentioned above, but how does this sum relate to the congruence induced by the sum of the two algebras?

This paper shows that under the condition of sort-similarity which requires that certain functions appear in both signatures, the congruence formed by summing two induced congruences is the same as the congruence induced by the summed algebras.

Additionally, with sort-similarity, the properties of compatibility and conservativity are ensured when forming sums of congruences. These two properties are required for the major results for the extended *tyft/tyxt* format. Finally, it is shown that reasonable conditions can be applied to algebras to ensure compatibility.

The main contribution of this paper is to advance the theory of formats by demonstrating that the conditions for the main results for the extended *tyft/tyxt* format are reasonable. These results are novel since sums of congruences have not been considered in this manner previously, and because compatibility and conservativity are requirements specific to the extended *tyft/tyxt* format.

Note that two different usages of congruence occur in this paper. The one describes a property of bisimulation with respect to the operators of the language. The other, which is the focus of the paper, refers to the congruences that the algebras representing the actual process algebra labels induce over the label terms.

In Section 2, the extended *tyft/tyxt* format is presented. Section 3 states the existing major results for this format. In Section 4, the results for algebras are presented. The last section looks at further work and conclusions. An earlier version of this material appeared in [12].

2 Extended *tyft/tyxt* format

This section presents definitions for many-sorted algebras (which are standard, but are presented here to fix notation) and then proceeds to present the definitions required for a format. Finally the definition of the format is given.

Definition 2.1 *For any set S, an S-*sorted set *A is a family $\{A_s\}_{s\in S}$ of sets indexed by S. Let $S' \subseteq S$, then $A_{S'}$ is the S'-sorted subset of A_S.*

Definition 2.2 *A* signature *Σ is a pair (S, F) where S is a set of* sorts *and F is a set of* function symbols (operators) *such that F is equipped with a mapping $type : F \to S^* \times S$. Write $f : s_1 \ldots s_n \to s$ for $f \in F$ with $type(f) = (s_1 \ldots s_n, s)$, and $f :\to s$ if $n = 0$. $f :\to s$ is a* constant.

Let V be an S-sorted set of variables disjoint from F. In the remainder of this section, $\Sigma = (S, F)$ is a signature.

Definition 2.3 *Let W be an S-sorted subset of V. For each $s \in S$, the set $T(\Sigma, W)_s$ of* Σ-terms of sort s *is the least set containing every $x \in W_s$ and every constant symbol $f :\to s \in F$, and every $f(t_1, \ldots, t_n)$ where $f : s_1 \ldots s_n \to s \in F$ and every t_i $(1 \leqslant i \leqslant n)$ is a term in $T(\Sigma, W)_{s_i}$.*

$T(\Sigma, W)$ *denotes* $\{T(\Sigma, W)_s\}_{s\in S}$, *the* set of Σ-terms over W. $T(\Sigma, \emptyset)$ *denotes the set of* closed *or* ground terms, *abbreviated* $\mathbf{T}(\Sigma)$, *and is called the* (ground) term algebra. $T(\Sigma, V)$, *abbreviated* $\mathbb{T}(\Sigma)$, *denotes* open terms.

Σ *is* sensible *if it admits at least one closed term for each sort, i.e. for all* $s \in S$, $\mathbf{T}(\Sigma)_s \neq \emptyset$. *The variables of sorts* $S' \subseteq S$ *in* $t \in \mathbb{T}(\Sigma)$, *Var*$_{S'}(t)$ *is defined by Var*$_{S'}(t) = \{x\}$ *if* $t = x$ *and* $x \in V_{S'}$, *and Var*$_{S'}(t) =$ *Var*$_{S'}(t_1) \cup \ldots \cup$ *Var*$_{S'}(t_n)$ *if* $t = f(t_1, \ldots, t_n)$,$f \in F$.

A substitution σ *is a mapping in* $V \to \mathbb{T}(\Sigma)$ *which preserves sorts, i.e.* $\sigma|V_s : V_s \to \mathbb{T}(\Sigma)_s$ *for each* $s \in S$. *A substitution* σ *is extended to a mapping* $\sigma : \mathbb{T}(\Sigma) \to \mathbb{T}(\Sigma)$ *in the standard way by* $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$ *for* $f : s_1 \ldots s_n \to s \in F$, $t_i \in \mathbb{T}(\Sigma)_{s_i}$, $1 \leqslant i \leqslant n$.

Definition 2.4 *A* Σ-algebra *consists of an* S*-sorted family of non-empty carrier sets* $\{\mathcal{A}_s\}_{s\in S}$, *also denoted* $\mathcal{A}$; *and a total function* $f^{\mathcal{A}} : \mathcal{A}_{s_1} \times \ldots \times \mathcal{A}_{s_n} \to \mathcal{A}_s$ *for each* $f s_1 \ldots s_n \to s \in F$.

Definition 2.5 *Let* $\mathcal{A}$ *and* $\mathcal{B}$ *be two* Σ*-algebras. A* Σ-homomorphism $h : \mathcal{A} \to \mathcal{B}$ *is a family of maps* $\{h_s : \mathcal{A}_s \to \mathcal{B}_s\}_{s\in S}$ *such that for all* $f : s_1 \ldots s_n \to s \in F$ *and* $a_1 \in \mathcal{A}_{s_1}, \ldots, a_n \in \mathcal{A}_{s_n}$, $h_s(f^{\mathcal{A}}(a_1, \ldots, a_n)) = f^{\mathcal{B}}(h_{s_1}(a_1), \ldots, h_{s_n}(a_n))$.

Definition 2.6 *A* Σ-congruence *on a* Σ*-algebra* $\mathcal{A}$ *is an* S*-sorted equivalence relation* $\equiv$ *which is compatible with all function symbols, i.e.* $\equiv = \{\equiv_s\}_{s\in S}$, *and for all* $s \in S$, $\equiv_s \subseteq \mathcal{A}_s \times \mathcal{A}_s$ *is reflexive, symmetric and transitive, and for any* $f : s_1 \ldots s_n \to s \in F$ *and for all* $a_i, b_i \in \mathcal{A}_{s_i}$ *for* $1 \leqslant i \leqslant n$, $a_i \equiv_{s_i} b_i$ $(1 \leqslant i \leqslant n) \Rightarrow f^{\mathcal{A}}(a_1, \ldots, a_n) \equiv_s f^{\mathcal{A}}(b_1, \ldots, b_n)$.

Both $\mathbb{T}(\Sigma)$ and $\mathbf{T}(\Sigma)$ form Σ-algebras and it can be shown that there is a unique homomorphism denoted $i_{\mathcal{A}}$ from the (ground) term algebra $\mathbf{T}(\Sigma)$ to any Σ-algebra $\mathcal{A}$. For each Σ-algebra, there exists a congruence over $\mathbf{T}(\Sigma)$, defined as $t \equiv_{\mathcal{A}} t'$ whenever $i_{\mathcal{A}}(t) = i_{\mathcal{A}}(t')$ for $t, t' \in \mathbf{T}(\Sigma)$. This congruence is said to be induced by $\mathcal{A}$.

A specific kind of sorted set and signature is required to represent the terms that appear in the rules of the format. Let P be a distinguished sort which is understood to the sort of processes.

Definition 2.7 *A signature* $\Sigma = (S \cup \{\mathsf{P}\}, F)$ *is* suitable *if* S *does not contain* P *and is non-empty, and for any function symbol* $f \in F$ *such that* $f : s_1 \ldots s_n \to s$, *whenever* $s \neq \mathsf{P}$ *then for all* $1 \leqslant i \leqslant n$, $s_i \neq \mathsf{P}$.

Hence only process terms can be built out of other process terms. This is reasonable because of the asymmetry in the way in which processes and labels are treated. For convenience, assume that functions with range sort P take the non-P arguments first and then the arguments of sort P. For the rest of this section $\Sigma = (S \cup \{\mathsf{P}\}, F)$ is a sensible, suitable signature.

Definition 2.8 *An* extended transition system specification (eTSS) *is a pair* $\mathcal{E} = (\Sigma, R)$ *and* R *a set of* $rules$ *of the form*

$$\frac{\{p_i \xrightarrow{\lambda_i} p_i' \mid i \in I\}}{p \xrightarrow{\lambda} p'}$$

where I *is an index set,* $p_i, p_i', p, p' \in \mathbb{T}(\Sigma)_{\mathsf{P}}$, *and* $\lambda_i, \lambda \in \mathbb{T}(\Sigma)_S$ *for* $i \in I$.

$p \xrightarrow{\lambda} p'$ *with* $\lambda \in \mathbb{T}(\Sigma)_S$ *and* $p, p' \in \mathbb{T}(\Sigma)$ *is a* transition (labelled with λ). *The elements of* $\{p_i \xrightarrow{\lambda_i} p_i' \mid i \in I\}$ *are the* premises *of* r, *and* $p \xrightarrow{\lambda} p'$ *is the* conclusion *of* r. *The notions of closed, substitution and Var can be extended to transitions in the obvious way.*

A proof *of a transition* ψ *from* $\mathcal{E}$ *is a well-founded, upwardly branching tree of which the nodes are labelled by transitions* $p \xrightarrow{\lambda} p'$ *such that the root is labelled with* ψ, *and if* χ *is the label of a node* π *and* $\{\chi_i \mid i \in I\}$ *is the set of labels of the nodes directly above* π, *then there is a rule* $\{\phi_i \mid i \in I\}/\phi$ *in* R *and a substitution* σ *such that* $\chi = \sigma(\phi)$ *and* $\chi_i = \sigma(\phi_i)$ *for all* $i \in I$.

If a proof of ψ *from* $\mathcal{E}$ *exists,* ψ *is* provable *from* $\mathcal{E}$, *notation* $\mathcal{E} \vdash \psi$.

As this research deals with process algebras, an appropriate definition of labelled transition system and bisimulation is required.

Definition 2.9 *The* (many-sorted) labelled transition system (LTS) $TS(\mathcal{E})$ specified by the eTSS $\mathcal{E}$ *is given by* $TS(\mathcal{E}) = (\mathbf{T}(\Sigma)_{\mathsf{P}}, \mathbf{T}(\Sigma)_S, \to)$ *where* $\to \subseteq \mathbf{T}(\Sigma)_{\mathsf{P}} \times \mathbf{T}(\Sigma)_S \times \mathbf{T}(\Sigma)_{\mathsf{P}}$ *is defined by* $u \xrightarrow{\alpha} u' \iff \mathcal{E} \vdash u \xrightarrow{\alpha} u'$.

Definition 2.10 *Let* $\mathcal{L} = (\mathcal{S}, \mathcal{A}, \to)$ *be an* S*-sorted LTS, and let* $\equiv$ *be an* S*-sorted equivalence relation on* $\mathcal{A}$. *A* strong bisimulation with respect to an equivalence relation $\equiv$ *is a binary relation* $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ *such that* $(s, t) \in \mathcal{R}$ *only if for all* $a \in \mathcal{A}$

1. *whenever* $s \xrightarrow{a} s'$, *then there exists* $t' \in \mathcal{S}$ *and* $b \in \mathcal{A}$ *such that* $t \xrightarrow{b} t'$, $a \equiv b$ *and* $(s', t') \in \mathcal{R}$
2. *whenever* $t \xrightarrow{a} t'$, *then there exists* $s' \in \mathcal{S}$ *and* $b \in \mathcal{A}$ *such that* $s \xrightarrow{b} s'$, $a \equiv b$ *and* $(s', t') \in \mathcal{R}$.

Two states, s and t are strongly bisimilar with respect to $\equiv$, $s \sim_{\equiv} t$, *if there exists a strong bisimulation* $\mathcal{R}$ *with respect to* $\equiv$ *such that* $(s, t) \in \mathcal{R}$.

Note that for this definition, the equivalence is only required over label terms $\mathbf{T}(\Sigma)_S$ and the definition of the equivalence over $\mathbf{T}(\Sigma)_{\mathsf{P}}$ is irrelevant. As mentioned previously, when expressing process algebras in this format, a Σ-algebra will be used to define the semantics of the actual labels and this will induce a congruence over the term algebra $\mathbf{T}(\Sigma)_S$.

The general definition of an eTSS is now made more specific to define the extended *tyft/tyxt* format. The actual details of the definition are not important for what follows, but are included for completeness.

Definition 2.11 *R is in* extended tyft/tyxt format *if it consists of rules of the form*

$$\frac{\{p_i \xrightarrow{\lambda_i} y_i \mid i \in I\}}{f(\eta_1,\ldots,\eta_m,x_1,\ldots,x_n) \xrightarrow{\lambda} p} \qquad \frac{\{p_i \xrightarrow{\lambda_i} y_i \mid i \in I\}}{x \xrightarrow{\lambda} p}$$

with I an index set, $f : s_1 \ldots s_m\mathsf{P}\ldots\mathsf{P} \to \mathsf{P} \in F$, x, x_j $(1 \leqslant j \leqslant n)$ and y_i $(i \in I)$ all different variables from V_P, $p \in \mathbb{T}(\Sigma)_\mathsf{P}$, $\lambda \in \mathbb{T}(\Sigma)_S$,

- *for $1 \leqslant k \leqslant m$, $\eta_k \in \mathbb{T}(\Sigma)_{s_k}$ with $Var_S(\eta_k) \subset V_S - \bigcup_{1\leqslant l\leqslant m, l\neq k} Var_S(\eta_l)$*
- *for $i \in I$, $\lambda_i \in \mathbb{T}(\Sigma)_S$ for $i \in I$ with $Var_S(\lambda_i) \subset V_S - (\bigcup_{l\in I, l\neq i} Var_S(\lambda_l) \cup \bigcup_{1\leqslant k\leqslant m} Var_S(\eta_k))$*
- *for $i \in I$, $p_i \in \mathbb{T}(\Sigma)_\mathsf{P}$ with $Var_S(p_i) \subset V_S - \bigcup_{l\in I} Var_S(\lambda_l)$.*

To relate this material back to the description of a process algebra given in the introduction, the many-sorted signature is used to described the processes and the actions they can perform – this is the syntax component. The rule set of the eTSS gives the operational semantics by which all possible transitions can be determined though building proofs with the rules – this gives rise to a labelled transition system with sorted labels which represent the actions of the processes. Additionally, the conditions for the extended *tyft/tyxt* format have been defined, as well as a definition of bisimulation where matching is done with respect to an equivalence. When expressing a process algebra in the format, the processes are expressed as terms with sort P and the labels/actions are expressed as terms with sorts from S. However, the actual labels of the process algebra will be expressed by an algebra which will induce an congruence over the label terms, and this congruence will be used in the definition of bisimulation.

3 Existing results for the format

This section gives an overview of important definitions and results for this format. The aim is to explain the usage of sums of congruences and the rôle that compatibility and conservativity play since the next section considers results about these properties.

In this section, some terms will be mentioned that are not defined. These terms are *well-founded, free, pure, label-free, label-pure, type-0* and *type-1*. Their definitions are not required here and can be found in [12, 13] together with the proofs for all the results mentioned in this section.

Firstly, some conditions on the congruence over label terms are required for the results. In what follows, $\mathcal{E} = (\Sigma, R)$ indicates an eTSS in extended *tyft/tyxt* format, and $\equiv$ is an S-sorted congruence on $\mathbf{T}(\Sigma)$.

Definition 3.1 *$\equiv$ is* compatible *with $\eta \in \mathbb{T}(\Sigma)_S$ if whenever $\sigma(\eta) \equiv \mu$ for $\mu \in \mathbf{T}(\Sigma)_S$, there exists a substitution σ' such that $\mu = \sigma'(\eta)$ and $\sigma(z) \equiv \sigma'(z)$ for all $z \in Var_S(\eta)$.*

$\equiv$ is compatible *with a rule $r \in R$ if it is compatible with any $\eta \in \mathbb{T}(\Sigma)_S$ that appears as the label on the transition of a premise of r or as an argument to the function on the left hand side of the conclusion of r. $\equiv$ is* compatible *with $\mathcal{E}$ if $\equiv$ is compatible with all rules in R.*

Compatibility is a technical requirement to ensure the major results hold. When trying to create a proof for a matching transition to achieve bisimulation, compatibility ensures a suitable substitution with which to construct the proof.

The first major result states that two terms of sort P constructed from subterms that are related (either by $\equiv$ if they have sort S or by bisimulation up to $\equiv$ if they have sort P) are bisimilar up to $\equiv$. Hence it can be concluded that the bisimulation up to $\equiv$ is itself a congruence.

Theorem 3.1 *Let $\mathcal{E} = (\sigma, R)$ be well-founded and $\equiv$ compatible with $\mathcal{E}$. For all $f \in F$ such that $f : s_1 \ldots s_m\mathsf{P}\ldots\mathsf{P} \to \mathsf{P}$, for all $\mu_k, \nu_k \in \mathbf{T}(\Sigma)_S$ $(1 \leqslant k \leqslant m)$, and for all $u_j, v_j \in \mathbf{T}(\Sigma)_\mathsf{P}$ $(1 \leqslant j \leqslant n)$,*
$\mu_k \equiv \nu_k$ and $u_j \sim_\equiv v_j \Rightarrow$
$f(\mu_1,\ldots,\mu_m,u_1,\ldots,u_n) \sim_\equiv f(\nu_1,\ldots,\nu_m,v_1,\ldots v_n)$.

Next to be considered are the results where two eTSSs are summed together. The notion of the sum of two eTSSs is required to define the notion of an extension. The second summand does not necessarily involve a sensible signature, and so the definitions are asymmetric.

Definition 3.2 *Let $\Sigma_i = (S_i \cup \{\mathsf{P}\}, F_i)$ for $i = 0, 1$ be two suitable signatures such that $f \in F_0 \cap F_1$ implies that $type_0(f) = type_1(f)$. If Σ_0 is sensible, and $\Sigma_0 \oplus \Sigma_1$ is sensible, then the* (asymmetric) sum of Σ_0 and Σ_1 *is the signature $(S_0 \cup S_1 \cup \{\mathsf{P}\}, F_0 \cup F_1)$.*

Let $\mathcal{E}_i = (\Sigma_i, R_i)$ for $i = 0, 1$ be two eTSSs with $\Sigma_0 \oplus \Sigma_1$ defined. The (asymmetric) sum of $\mathcal{E}_0$ and $\mathcal{E}_1$, *$\mathcal{E}_0 \oplus \mathcal{E}_1$, is the eTSS $(\Sigma_0 \oplus \Sigma_1, R_0 \cup R_1)$.*

In the rest of the papers, let $\Sigma_i = (S_i \cup \{\mathsf{P}\}, F_i)$ for $i = 0, 1$ be two signatures with $\Sigma = \Sigma_0 \oplus \Sigma_1$ defined, and let $\mathcal{E}_i = (\Sigma_i, R_i)$ for $i = 0, 1$ be two eTSSs in extended *tyft/tyxt* format with $\mathcal{E} = \mathcal{E}_0 \oplus \mathcal{E}_1$ defined and let $\mathcal{E} = (\Sigma, R)$. Let $\equiv_i$ be congruences over $\mathbf{T}(\Sigma_i)_S$ compatible with $\mathcal{E}_i$ for $i = 0, 1$. A way to combine congruences is required since each of the eTSSs has it own congruence.

Definition 3.3 *The* sum of $\equiv_0$ and $\equiv_1$ *($\equiv_0 \oplus \equiv_1$) is the smallest congruence over $\mathbf{T}(\Sigma_0 \oplus \Sigma_1)$ containing both $\equiv_0$ and $\equiv_1$.*

Definition 3.4 *$\equiv_0 \oplus \equiv_1$ is* conservative with respect to $\equiv_i$ *if $(\equiv_0 \oplus \equiv_1) \cap (\mathbf{T}(\Sigma_i)_S \times \mathbf{T}(\Sigma_i)_S) = \equiv_i$.*

Hence, conservativity of congruences requires that summing two congruences does not identify terms from one of the component term algebras that were not originally identified, although an element from one term algebra may be identified with an element from the other term algebra by summing. This property is used to show that when two terms from one of the term algebras are not bisimilar, it is not possible to make the terms bisimilar by summing congruences.

A notion of a conservative extension up to bisimulation has been defined by Groote and Vaandrager[14]. The following are modifications of this definition which describe the relationship between the semantic equivalence defined on one eTSS, and the semantic equivalence defined on both eTSSs. The first definition describes the situation where the semantic equivalence on the summed eTSS equates few process than that of the first system, and the second definition deals with opposite case. If conditions can be found under which these definitions hold, then these results can be used to compare semantic equivalences. The following two theorems give appropriate conditions.

Definition 3.5 *$\mathcal{E}$ is a* refining extension of *$\mathcal{E}_0$* up to bisimulation with respect to $\equiv$ *if for all* $t_0, u_0 \in \mathbf{T}(\Sigma_0)_{\mathsf{P}}$, $t_0 \sim^{\mathcal{E}}_{\equiv_0 \oplus \equiv_1} u_0 \Rightarrow t_0 \sim^{\mathcal{E}_0}_{\equiv_0} u_0$.

Definition 3.6 *$\mathcal{E}$ is an* abstracting extension of $\mathcal{E}_0$ up to bisimulation with respect to $\equiv$ *if for all* $t_0, u_0 \in \mathbf{T}(\Sigma_0)_{\mathsf{P}}$, $t_0 \sim^{\mathcal{E}_0}_{\equiv_0} u_0 \Rightarrow t_0 \sim^{\mathcal{E}}_{\equiv_0 \oplus \equiv_1} u_0$

Theorem 3.2 *Let $\mathcal{E}_0$ and $\mathcal{E}_1$ be in extended* tyft/tyxt *format. Let $\equiv_0 \oplus \equiv_1$ be conservative with respect to $\equiv_0$. If $\mathcal{E}_0$ is pure and label-pure, and $\mathcal{E}_0 \oplus \mathcal{E}_1$ is type-1, then $\mathcal{E}_0 \oplus \mathcal{E}_1$ is a refining extension.*

Theorem 3.3 *Let $\mathcal{E}_0$ and $\mathcal{E}_1$ be in extended* tyft/tyxt *format. Let $\equiv_0 \oplus \equiv_1$ be compatible with $\mathcal{E}_0 \oplus \mathcal{E}_1$. If $\mathcal{E}_0$ is pure, label-pure and well-founded, $\mathcal{E}_1$ is well-founded, and $\mathcal{E}_0 \oplus \mathcal{E}_1$ is type-0, then $\mathcal{E}_0 \oplus \mathcal{E}_1$ is an abstracting extension.*

These theorems and Theorem 3.1 use compatibility, conservativity and sums of congruence and these have been shown to be necessary by counter-example [12]. These results have been applied in the comparison of multiprocessor bisimulation and pomset bisimulation [12, 7, 16].

It is important to note that although these results have been presented for arbitrary congruences, when applying these results to process algebras, the congruences of interest are those that are induced by the algebra that represents the labels. The next section looks at results to do with these specific congruences.

4 Algebraic results

This section considers sums of congruences, compatibility and conservativity required for the results in the previous section and how these relate to the algebras which represent the labels of the process algebras.

4.1 Sums of congruences and algebras

The results in the previous section considered the sum of two eTSSs $\mathcal{E}_0 \oplus \mathcal{E}_1$, and the associated sum of congruences $\equiv_0 \oplus \equiv_1$. In this section, it is assumed that for each $\mathcal{E}_i = (\Sigma_i, R_i)$ there is an associated Σ_i-algebra, $\mathcal{A}_i$ for $i = 0, 1$ to represent the actual label terms. Then the congruences of interest are $\equiv_{\mathcal{A}_0}$ and $\equiv_{\mathcal{A}_1}$ induced by these algebras over $\mathbf{T}(\Sigma_0)$ and $\mathbf{T}(\Sigma_1)$ respectively. Theorems 3.2 and 3.3 refer to the sum of these congruences $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$. How does this congruence relate to the congruence $\equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1}$ induced on $\mathbf{T}(\Sigma_0 \oplus \Sigma_1)$ by the sum of $\mathcal{A}_0$ and $\mathcal{A}_1$?

This section considers signatures in general, and does not need to consider the distinguished sort P for the reason that since no process terms can appear as arguments to label terms, the label terms form an algebra in their own right (with a different signature consisting the sorts S and those function symbols with range sort S). The definition of the union of two many-sorted sets is straightforward, and this can be used to define the sum of two algebras.

Definition 4.1 *Let S_0 and S_1 be two sets, and let $A_i = \{A_{i,s}\}_{s \in S_i}$ be an S_i-sorted set for $i = 0, 1$. Let $S = S_0 \cup S_1$. Then the S*-sorted union *of A_0 and A_1 is defined as $A = \{A_s\}_{s \in S}$ where*

- *$A_s = A_{0,s}$ if $s \in S_0 - S_1$*
- *$A_s = A_{1,s}$ if $s \in S_1 - S_0$*
- *$A_s = A_{0,s} \cup A_{1,s}$ if $s \in S_0 \cap S_1$*

and is denoted $\biguplus_{i=0,1} A_i$.

Definition 4.2 *For $i = 0, 1$, let $\mathcal{A}_i = \{\mathcal{A}_{i,s}\}_{s \in S_i}$ together with $\{f^{\mathcal{A}_i} \mid f \in F_i\}$ be Σ_i-algebras such that the following conditions hold.*

- *for $s \in S_0 \cap S_1$, $\mathcal{A}_{0,s} = \mathcal{A}_{1,s}$,*
- *for $f \in F_0 \cap F_1$ with $f : s_1 \ldots s_n \rightarrow s$, $f^{\mathcal{A}_0}(a_1, \ldots, a_n) = f^{\mathcal{A}_1}(a_1, \ldots, a_n)$ for all $a_j \in \mathcal{A}_{0,s_j}$ for $1 \leqslant j \leqslant n$.*

Define $\mathcal{A}_0 \oplus \mathcal{A}_1 = \biguplus_{i=0,1} \mathcal{A}_i$ plus the functions $\bigcup_{i=0,1} \{f^{\mathcal{A}_i} \mid f \in F_i\}$.

Note that the second condition relies on the fact that since $f \in F_0 \cap F_1$, $s_i \in S_0 \cap S_1$ for $1 \leqslant i \leqslant n$, hence $f^{\mathcal{A}_0}$ and $f^{\mathcal{A}_1}$ are defined on the same sets.

For the rest of this paper, let $\mathcal{A}_i = \{\mathcal{A}_{i,s}\}_{s \in S_i}$ together with $\{f^{\mathcal{A}_i} \mid f \in F_i\}$ be Σ_i-algebras for $i = 0, 1$ with $\mathcal{A}_0 \oplus \mathcal{A}_1$ defined.

Proposition 4.1 *$\mathcal{A}_0 \oplus \mathcal{A}_1$ is a $(\Sigma_0 \oplus \Sigma_1)$-algebra.*

Proof: $\Sigma_0 \oplus \Sigma_1 = (S_0 \cup S_1, F_0 \cup F_1)$, and $\biguplus_{i=0,1} A_i$ is an $S_0 \cup S_1$-sorted family of non-empty carrier sets. Moreover, it is clear that for each $f \in F_0 \cup F_1$ there is a total function $f^{\mathcal{A}_0 \oplus \mathcal{A}_1}$ with the appropriate argument sorts and result sort, regardless of whether f is in the intersection of F_0 and F_1 or not. ■

A particular class of signatures is of interest, namely those whose sums satisfy the following definition.

Definition 4.3 $\Sigma_0 \oplus \Sigma_1$ *is* sort-similar *if for each* $s \in S_0 \cap S_1$, $f \in F_0 \cup F_1$ *with* $f : s_1 \dots s_n \to s$ *implies* $f \in F_0 \cap F_1$.

This definition goes beyond that of the sum of two signatures where if a function appears in the intersection of the two signatures, then it has the same type in each signature. Here it is necessary that any function that has a shared result sort, must appear in both signatures. A result follows from this definition, which is required for the later results.

Proposition 4.2 *Let* $\Sigma_0 \oplus \Sigma_1$ *be sort-similar. If* $t \in \mathbf{T}(\Sigma_0 \oplus \Sigma_1)$ *has sort* $s \in S_0 \cap S_1$*, then* $t \in \mathbf{T}(\Sigma_0) \cap \mathbf{T}(\Sigma_1)$*. Moreover* $\mathbf{T}(\Sigma_0 \oplus \Sigma_1) = \mathbf{T}(\Sigma_0) \cup \mathbf{T}(\Sigma_1)$.

Proof: For the first part of the result, consider $t \in \mathbf{T}(\Sigma_0 \oplus \Sigma_1)_s$ for $s \in S_0 \cap S_1$. This proceeds by induction on the structure of t. If t is a constant $f : \to s$ then $f \in F_0 \cap F_1$ and clearly $t \in \mathbf{T}(\Sigma_0) \cap \mathbf{T}(\Sigma_1)$. If $t = f(t_1, \dots, t_n)$ for $f : s_1 \dots s_n \to s$, then $f \in F_0 \cap F_1$ and by the induction hypothesis $t_i \in \mathbf{T}(\Sigma_0) \cap \mathbf{T}(\Sigma_1)$ for $1 \leqslant i \leqslant n$, hence $t \in \mathbf{T}(\Sigma_0) \cap \mathbf{T}(\Sigma_1)$.

Since clearly $\mathbf{T}(\Sigma_0) \cup \mathbf{T}(\Sigma_1) \subseteq \mathbf{T}(\Sigma_0 \oplus \Sigma_1)$, it is necessary to show for any $t \in \mathbf{T}(\Sigma_0 \oplus \Sigma_1)$ that either $t \in \mathbf{T}(\Sigma_0)$ or $t \in \mathbf{T}(\Sigma_1)$. This proceeds by induction on the structure of t.

If t is a constant symbol, then clearly $t \in \mathbf{T}(\Sigma_0) \cup \mathbf{T}(\Sigma_1)$. Otherwise consider $t = f(t_1, \dots, t_n)$. Assume that $f \in F_0$ without loss of generality, and that $f : s_1 \dots s_n \to s$, then $s_1 \dots s_n, s \in S_0$. By the induction hypothesis, for $1 \leqslant i \leqslant n$, $t_i \in \mathbf{T}(\Sigma_0) \cup \mathbf{T}(\Sigma_1)$. If any $t_i \in \mathbf{T}(\Sigma_1)$, then since $s_i \in S_0 \cap S_1$, $t_i \in \mathbf{T}(\Sigma_0)$ also. Hence for $1 \leqslant i \leqslant n$, $t_i \in \mathbf{T}(\Sigma_0)$, and $t \in \mathbf{T}(\Sigma_0)$. ■

This result shows how under the condition of sort-similarity, no closed terms can be created from function symbols from both signatures without being in both sets of closed terms.

The sort-similarity condition is not required for the sum of algebras (Definition 4.2) since the second condition works because $f^{\mathcal{A}_0}$ and $f^{\mathcal{A}_1}$ are defined on the same sets; however, it is required for Theorem 4.1 to be proved. Consider the following example.

Example 4.1 *Consider the* $\{s\}$*-sorted signature,* $\Sigma_0 = (\{s\}, \{f\})$ *with* $f : \to s$*, and the* $\{s\}$*-sorted signature,* $\Sigma_1 = (\{s\}, \{g\})$ *with* $g : \to s$*. Clearly the sum of these two signatures is not sort-similar. The same set* $\{a\}$ *will be used for both the* Σ_0*-algebra* $\mathcal{A}_0$ *and* Σ_1*-algebra* $\mathcal{A}_1$*. Let* $f^{\mathcal{A}_0} = a$ *and* $g^{\mathcal{A}_1} = a$*. Then* $f \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} g$ *since* $i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(f) = i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(g)$*, but* f *and* g *are not equated by* $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$.

It may be possible to work without sort-similarity—possibly the definition of $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_0}$ could be modified or the theorem could be changed to have a one-way implication. There are other choices one could make for $\mathcal{A}_0 \oplus \mathcal{A}_1$, such as defining the carrier set for a sort s as the intersection of $\mathcal{A}_{0,s}$ and $\mathcal{A}_{1,s}$, however there does not appear to be any utility in taking these approaches. Sort-similarity is not an unreasonable condition for label terms, as there is no advantage to be gained by having a function symbol in one signature but not the other.

Theorem 4.1 *Let the sum* $\Sigma_0 \oplus \Sigma_1$ *be sort-similar then* $\equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} = \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$.

Proof: First note, that for any $t \in \mathbf{T}(\Sigma_0 \oplus \Sigma_1)$, $t \in \mathbf{T}(\Sigma_0) \cup \mathbf{T}(\Sigma_1)$ by Proposition 4.2. First show that if $t \in \mathbf{T}(\Sigma_i)$, then $i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t) = i_{\mathcal{A}_i}(t)$ for $i = 0, 1$. This proceeds by induction on the structure of t.

If t is a constant symbol, then this is true. If $t = f(t_1, \dots, t_n)$, then assume $t \in \mathbf{T}(\Sigma_0)$ without loss of generality. $i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(f(t_1, \dots, t_n)) = f^{\mathcal{A}_0}(i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t_1), \dots, i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t_n)) = f^{\mathcal{A}_0}(i_{\mathcal{A}_0}(t_1), \dots, i_{\mathcal{A}_0}(t_n))$ by the induction hypothesis and this gives the required result.

Next, let $t, t' \in \mathbf{T}(\Sigma_0 \oplus \Sigma_1)$.

$t \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t' \Rightarrow t \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t'$**:** Let $t \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t'$, then since this is an congruence this means that t and t' have the same sort and hence they must be both in $\mathbf{T}(\Sigma_0)$ or $\mathbf{T}(\Sigma_1)$. Assume without loss of generality that $t, t' \in \mathbf{T}(\Sigma_0)$. By definition $i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t) = i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t')$, hence $i_{\mathcal{A}_0}(t) = i_{\mathcal{A}_0}(t')$. Therefore $t \equiv_{\mathcal{A}_0} t'$ and hence $t \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t'$ as required.

$t \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t' \Rightarrow t \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t'$**:** This proceeds by induction on the definition of $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$.

- $t \equiv_{\mathcal{A}_0} t'$ or $t \equiv_{\mathcal{A}_1} t'$. Assume $t \equiv_{\mathcal{A}_0} t'$, then by definition $i_{\mathcal{A}_0}(t) = i_{\mathcal{A}_0}(t')$ and also $t, t' \in \mathbf{T}(\Sigma_0)$. It can be shown by a simple induction proof that $i_{\mathcal{A}_0}(t) = i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t)$. If t is a constant then this is immediate; otherwise if $t = f(t_1, \dots, t_n)$ then $i_{\mathcal{A}_0}(f(t_1, \dots, t_n) = f^{\mathcal{A}_0}(i_{\mathcal{A}_0}(t_1), \dots, i_{\mathcal{A}}(t_n)) = f^{\mathcal{A}_0 \oplus \mathcal{A}_1}(i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t_1), \dots, i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t_n))$ by the induction hypothesis and the definition of $\mathcal{A}_0 \oplus \mathcal{A}_1$, and this gives the required result.

- Clearly both $t \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t'$ and $t \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t'$ if $t = t'$.

- If $t \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t'$ by a symmetry argument, then $t' \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t$, by a shorter inference, hence $t' \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t$ and therefore $t \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t'$ since $\equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1}$ is symmetric.

- If $t \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t'$ by a transitivity argument, then there exists $t'' \in \mathbf{T}(\Sigma_0 \oplus \Sigma_1)$ with $t \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t''$ and $t'' \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t'$, hence by a shorter inference $t \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t''$ and $t'' \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t'$, therefore $t \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t'$ since $\equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1}$ is transitive.

- If $t \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t'$ by a congruence argument, then $t = f(t_1, \dots, t_n)$, $t' = f'(t'_1, \dots, t'_n)$ and $t_i \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t'_i$ for $1 \leqslant i \leqslant n$, so by a shorter inference $t_i \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t'_i$ for $1 \leqslant i \leqslant n$. Hence $t \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t'$, since $\equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1}$ is a congruence. ■

4.2 Conservativity and compatibility

It can be shown that $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$ is conservative with respect to $\equiv_{\mathcal{A}_0}$ and $\equiv_{\mathcal{A}_1}$.

Proposition 4.3 *Let $\Sigma_0 \oplus \Sigma_1$ be sort-similar then $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$ is conservative with respect to both $\equiv_{\mathcal{A}_0}$ and $\equiv_{\mathcal{A}_1}$.*

Proof: Let $t, t' \in \mathbf{T}(\Sigma_0)$ (without loss of generality) such that $t \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} t'$. Hence $t \equiv_{\mathcal{A}_0 \oplus \mathcal{A}_1} t'$, so $i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t) = i_{\mathcal{A}_0 \oplus \mathcal{A}_1}(t')$. Hence since $t, t' \in \mathbf{T}(\Sigma_0)$, $i_{\mathcal{A}_0}(t) = i_{\mathcal{A}_0}(t')$ and $t \equiv_{\mathcal{A}_0} t'$ as required. ■

Clearly, this does not hold in general since it is possible in the construction of $\equiv_1 \oplus \equiv_2$ from two arbitrary congruences $\equiv_1$ and $\equiv_2$ to obtain two terms from $\mathbf{T}(\Sigma_0)$ that are not identified under $\equiv_1$ to become identified under $\equiv_1 \oplus \equiv_2$ by the fact that they may be related to terms that fall into $\mathbf{T}(\Sigma_0) \cap \mathbf{T}(\Sigma_1)$.

Next, compatibility is considered in the light of sums of algebras. In the results that involve compatibility, it is only required that $\equiv_0 \oplus \equiv_1$ be compatible with $R_0 \cup R_1$. It is not clear that this can be dealt with in a more general manner since compatibility depends on the label terms that appear in the rules. One approach could be to require the congruences to be compatible with any label term, but this could impose very strong restrictions on the Σ-algebras that could be used. However, in the case of sums of algebras it is possible to obtain a result because of the condition of sort-similarity.

Proposition 4.4 *Let $\Sigma_0 \oplus \Sigma_1$ be sort-similar. If $\equiv_{\mathcal{A}_i}$ is compatible with R_i for $i = 0, 1$ then $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$ is compatible with $R_0 \cup R_1$.*

Proof: Since $\equiv_{\mathcal{A}_i}$ is compatible for R_i for $i = 0, 1$, there are two S_i-sorted sets of terms $C_i \subseteq \mathbb{T}(\Sigma)_{S_i}$ for $i = 0, 1$ containing terms that appear as a label on a transition in a premise or as an argument to the function in the right hand side of the conclusion of a rule in R_i. So consider the following sorts:

- $s \in S_0 - S_1$**:** Consider $\eta \in \mathbb{T}(\Sigma_0)_s$. Since $s \notin S_1$, then clearly no terms in $\mathbf{T}(\Sigma_1)$ can be equivalent under $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$ to $\sigma(\eta)$ for any σ. So consider $\sigma(\eta) \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} \mu$ for some $\mu \in \mathbf{T}(\Sigma_0)_s$, then it can be shown that $\sigma(\eta) \equiv_{\mathcal{A}_0} \mu$ and hence there exists a substitution σ' such that $\mu = \sigma'(\eta)$ and $\sigma(z) \equiv_{\mathcal{A}_0} \sigma'(z)$ for all $z \in \text{Var}_S(\eta)$. But since $\equiv_{\mathcal{A}_0}$ is contained in $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$, then $\sigma(z) \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} \sigma'(z)$ for all $z \in \text{Var}_S(\eta)$.
- $s \in S_1 - S_0$**:** this is proved in a similar way to the previous one.
- $s \in S_0 \cap S_1$ Consider $\eta \in \mathbb{T}(\Sigma_0)_s \cap \mathbb{T}(\Sigma_1)_s$. Consider $\sigma(\eta) \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} \mu$ for $\mu \in \mathbf{T}(\Sigma_0)_s \cap \mathbf{T}(\Sigma_1)_s$. It can be shown that $\sigma(\eta) \equiv_{\mathcal{A}_0} \mu$ (and that $\sigma(\eta) \equiv_{\mathcal{A}_1} \mu$). Hence there exists a substitution σ' such that $\mu = \sigma'(\eta)$ and $\sigma(z) \equiv_{\mathcal{A}_0} \sigma'(z)$ for all $z \in \text{Var}_S(\eta)$. But since $\equiv_{\mathcal{A}_0}$ is contained in $\equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1}$, then $\sigma(z) \equiv_{\mathcal{A}_0} \oplus \equiv_{\mathcal{A}_1} \sigma'(z)$ for all $z \in \text{Var}_S(\eta)$. ■

To see why this does not hold in general for two congruences $\equiv_0$ and $\equiv_1$ defined on $\mathbf{T}(\Sigma_0)$ and $\mathbf{T}(\Sigma_1)$ compatible with two sets of rules R_0 and R_1 respectively, it may be that some η appears in the rules of R_0 but not in the rules of R_1. If $\sigma(\eta)$ is also in $\mathbf{T}(\Sigma_1)$ and there exists $\mu \in \mathbf{T}(\Sigma_1)$, such that $\sigma(\eta) \equiv_1 \mu$, it may not be possible to find σ' such that $\mu = \sigma'(\eta)$ and $\sigma(z) \equiv_1 \sigma'(z)$ for all $z \in \text{Var}_S(\eta)$, since η does not appear in the rules of R_1.

4.3 Requirements for compatibility

Compatibility is required for a number of results, specifically the congruence result (Theorem 3.1) and one of the extension results (Theorem 3.3). Conditions on Σ-algebras to ensure compatibility are now considered. The following result requires that the term in question contains no repeated variables. This is not a serious limitation as when there are repeated variables, the only compatible congruence is the identity relation.

Proposition 4.5 *Let $\Sigma = (S, F)$ be a signature, and let $\mathcal{A}$ be a Σ-algebra. Let $\lambda \in \mathbb{T}(\Sigma)$ be a term with no repeated variables, then $\equiv_{\mathcal{A}}$ is compatible with λ if for all function symbols g that appear in λ with $g^{\mathcal{A}} : \mathcal{A}_{s_1} \times \ldots \times \mathcal{A}_{s_n} \to \mathcal{A}_s$,*

- *$Im(g^{\mathcal{A}}) \cap Im(g_1^{\mathcal{A}}) = \emptyset$ for all $g_1^{\mathcal{A}} : \mathcal{A}_{s'_1} \times \ldots \times \mathcal{A}_{s'_m} \to \mathcal{A}_s$,*
- *$g^{\mathcal{A}}$ is injective.*

Proof: Let $\alpha \equiv_{\mathcal{A}} \sigma(\lambda)$ for some $\alpha \in \mathbf{T}(\Sigma)$ and a closed substitution σ. It is necessary to find a substitution σ' such that $\sigma'(\lambda) = \alpha$ and $\sigma(z) \equiv_{\mathcal{A}} \sigma'(z)$ for all $z \in \text{Var}(\lambda)$. The proof will be proceed by induction on the structure of λ. For the base case, assume $\lambda = z$ for $z \in V$ and define σ' as follows

$$\sigma'(z') = \begin{cases} \alpha & \text{if } z' = z \\ \sigma(z') & \text{otherwise.} \end{cases}$$

Then it is clear that the conditions for compatibility are satisfied. Next consider $\lambda = g(\lambda_1, \ldots, \lambda_n)$ where $g : s_1 \ldots s_n \to s$. Hence

$$\begin{aligned} i_{\mathcal{A}}(\alpha) &= i_{\mathcal{A}}(\sigma(g(\lambda_1, \ldots, \lambda_n))) \\ &= i_{\mathcal{A}}(g(\sigma(\lambda_1), \ldots, \sigma(\lambda_n))) \\ &= g^{\mathcal{A}}(i_{\mathcal{A}}(\sigma(\lambda_1)), \ldots, i_{\mathcal{A}}(\sigma(\lambda_n))) \end{aligned}$$

Fact 4.1 $g^{\mathcal{A}}(a_1, \ldots, a_n) = i_{\mathcal{A}}(\alpha) \Rightarrow \alpha = g(\mu_1, \ldots, \mu_n)$ *for some $\mu_1, \ldots, \mu_n$.*

Proof: Suppose not, i.e. $\alpha = g_1(\mu'_1, \ldots, \mu'_m)$ for $g_1 \neq g$ with $g_1 : s'_1 \ldots s'_m \to s$. Then

$$\begin{aligned} g^{\mathcal{A}}(a_1, \ldots, a_n) &= i_{\mathcal{A}}(\alpha) \\ &= i_{\mathcal{A}}(g_1(\mu'_1, \ldots, \mu'_m)) \\ &= g_1^{\mathcal{A}}(i_{\mathcal{A}}(\mu'_1), \ldots, i_{\mathcal{A}}(\mu'_m)). \end{aligned}$$

Contradiction since $Im(g^{\mathcal{A}}) \cap Im(g_1^{\mathcal{A}}) = \emptyset$.

So by this fact, there exist $\mu_1, \ldots \mu_n$ such that $\alpha = g(\mu_1, \ldots, \mu_n)$. Hence

$$\begin{array}{ll} & g^{\mathcal{A}}(i_{\mathcal{A}}(\sigma(\lambda_1)), \ldots, i_{\mathcal{A}}(\sigma(\lambda_n))) \\ = & i_{\mathcal{A}}(g(\mu_1, \ldots, \mu_n)) \\ = & g^{\mathcal{A}}(i_{\mathcal{A}}(\mu_1), \ldots, i_{\mathcal{A}}(\mu_n)) \end{array}$$

Hence $i_{\mathcal{A}}(\sigma(\lambda_i)) = i_{\mathcal{A}}(\mu_i)$ for all $1 \leqslant i \leqslant n$ (since $g_{\mathcal{A}}$ is injective) and $\sigma(\lambda_i) \equiv_{\mathcal{A}} \mu_i$ for all $1 \leqslant i \leqslant n$. By the inductive hypothesis, there are substitutions σ_i such that for each $1 \leqslant i \leqslant n$, $\sigma_i(\lambda_i) = \mu_i$, and $\sigma(z) \equiv_{\mathcal{A}} \sigma_i(z)$ for all $z \in \text{Var}(\lambda_i)$. From these substitutions, construct a substitution that fulfils the compatibility requirements. Let σ' be defined as follows

$$\sigma'(z) \quad = \quad \begin{cases} \sigma_i(z) & \text{if } z \in \text{Var}(\lambda_i) \\ \sigma(z) & \text{otherwise.} \end{cases}$$

This is well-defined since there are no repeated variables. Recall that

$$\begin{array}{lll} \sigma'(\lambda) & = & \sigma'(g(\lambda_1, \ldots, \lambda_n)) \\ & = & g(\sigma'(\lambda_1), \ldots, \sigma'(\lambda_n)) \\ & = & g(\sigma_1(\lambda_1), \ldots, \sigma_n(\lambda_n)) \\ & = & g(\mu_1, \ldots, \mu_n) \\ & = & \alpha. \end{array}$$

Also for $z \in \text{Var}(\lambda)$, $\sigma'(z) = \sigma_i(z)$ for a unique $1 \leqslant i \leqslant n$, and $\sigma_i(z) \equiv_{\mathcal{A}} \sigma(z)$, hence $\sigma'(z) \equiv_{\mathcal{A}} \sigma(z)$. ■

5 Further work and conclusions

The results relating to sums can be summarised as follows. Given the sort-similar sum of two signatures, and two algebras of each signature respectively, then

- the sum of two algebras is an algebra of the sum of the signatures (sort-similarity is not necessary for this),
- the congruence induced by the sum of two algebras is the sum of the congruences induced by each algebra,
- moreover, this congruence is conservative with respect to each of the other two congruences,
- furthermore if given the sum of two eTSSs with the two congruences compatible with each eTSS respectively, then the congruences induced by the sum of the algebras is compatible with the sum of the eTSSs.

The result for compatibility states that the congruence induced over the term algebra is compatible with an open term if any function which appears in the open term is injective and has an image which is disjoint from the image of any other function.

It is not clear whether there is further work possible that relates to the results presented here. However, a question of interest is under what conditions transition equivalence is preserved by sum of eTSSs. It may be the case that reasonable conditions can be found under which transition equivalence is preserved, and this may lead to simpler proofs for the main results for the format.

To conclude, the results here have demonstrated that sort-similarity is a reasonable condition to show results about sums of algebras and congruences, and to ensure that properties such as compatibility and conservativity hold. Also, compatibility can be achieved by reasonable conditions on the algebras representing the labels of the process algebra. These, in turn, demonstrate that the use of these conditions in the major results for the extended *tyft/tyxt* format are reasonable, and hence that these conditions and properties do not prevent the application of the results for the extended *tyft/tyxt* format in the comparison of process algebra semantic equivalences.

References

[1] J.C.M. Baeten and C. Verhoef. A congruence theorem for structured operational semantics with predicates. In E. Best, editor, *CONCUR '93*, number 715 in Lecture Notes in Computer Science, pages 477–492. Springer-Verlag, 1993.

[2] K.L. Bernstein. A congruence theorem for structured operational semantics of higher-order languages. In *LICS 98*, pages 153–164. IEEE Computer Society Press, 1998.

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

[4] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42:232–268, 1995.

[5] R. Bol and J.F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43:863–914, 1996.

[6] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. *Formal Aspects of Computing*, 6(2):165–200, 1994.

[7] I. Castellani. *Bisimulations for concurrency*. PhD thesis, Department of Computer Science, University of Edinburgh, 1988. LFCS report ECS-LFCS-88-51.

[8] P. Darondeau and P. Degano. Causal trees. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *ICALP 88*, number 372 in Lecture Notes in Computer Science, pages 234–248. Springer-Verlag, 1989.

[9] R. de Simone. Higher-level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

[10] G.L. Ferrari and U. Montanari. Parameterized structured operational semantics. *Fundamenta Informatica*, 34:1–31, 1998.

[11] W. Fokkink and C. Verhoef. A conservative look at operational semantics with variable binding. *Information and Computation*, 146:24–54, 1998.

[12] V.C. Galpin. *Equivalence semantics for concurrency: comparison and application*. PhD thesis, Department of Computer Science, University of Edinburgh, 1998. LFCS report ECS-LFCS-98-397. `http://www.dcs.ed.ac.uk/lfcsreps/`.

[13] V.C. Galpin. Comparison of process algebra equivalences using formats. In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *ICALP '99*, number 1644 in Lecture Notes in Computer Science, pages 341–350. Springer-Verlag, 1999.

[14] J.F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100:202–260, 1992.

[15] A. Kiehn. Comparing locality and causality based equivalences. *Acta Informatica*, 31(8):697–718, 1994.

[16] P. Krishnan. Architectural CCS. *Formal Aspects of Computing*, 162:162–187, 1996.

[17] C.A. Middelburg. Variable binding operators in transition system specifications. To appear in *Journal of Logic and Algebraic Programming*.

[18] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 1201–1242. MIT Press, 1994.

[19] G. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1988.

[20] R. J. van Glabbeek. Full abstraction in structural operational semantics. In M. Nivat, C. Rattray, T. Rus, and G. Scollo, editors, *AMAST '93*, Workshops in Computing, pages 77–84. Springer-Verlag, 1993.

[21] C. Verhoef. A congruence theorem of structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2:274–302, 1995.

Acknowledgements

This research was supported in part by a Patrick and Margaret Flanagan Scholarship.