

Orthogonal Extensions in Structural Operational Semantics

MohammadReza Mousavi and Michel A. Reniers
Department of Computer Science,
Eindhoven University of Technology,
NL-5600 MB Eindhoven, The Netherlands

Abstract

In this paper, we give novel and more liberal notions of operational and equational conservativity for language extensions. We motivate these notions by showing their practical application in existing formalisms. Based on our notions, we formulate and prove meta-theorems that establish conservative extensions for languages defined using Structural Operational Semantics (SOS).

Key words: Formal Semantics, Structural Operational Semantics (SOS), Conservative Extension, Operational Conservativity, Equational Conservativity, Orthogonality.

1 Introduction

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 intuition behind the old subset, or said otherwise, the extensions are *conservative*. In the context of languages with Structural Operational Semantics (SOS) [17], this topic has been touched upon in [12, 13] and studied in depth in [1, 5, 8, 16, 21, 23]. This research has resulted in meta-theorems proving sufficient conditions for an extension to be *operationally* and/or *equationally conservative*. In the remainder, we mostly refer to [8] which gives the most detailed account of the problem and subsumes almost all previous results. We do not treat multi-sorted and variable binding signatures, addressed in [8, 16], in this paper.

So far, *operational conservativity* has only allowed for extensions that consistently deny the addition of any new behavior to the old syntax. One can imagine that an extension which grants a new behavior consistently to the old syntax can also be considered safe or “conservative”. This phenomenon occurs quite often in practice. For example, designers of many timed extensions of existing formalisms (e.g., the timed process algebras of [2, 3, 14, 20]) have decided to add timed behavior homogeneously to the terms from the old syntax. Unfortunately, it turns out that the existing definitions and their corresponding meta-theorems come short of any formal result about such extensions.

In this paper, we present a more liberal notion of operational conservativity, called *orthogonality*, which caters for both possibilities (i.e., denying some types of behavior from the old syntax while granting some other types). We show that our notion is useful in the aforementioned cases where the old notions cannot be used. We formulate orthogonality meta-theorems for languages with Structural Operational Semantics and prove them correct.

In [21], *equational extensions* are considered in the setting where a new set of axioms is added to an existing set. Then, the extension is called *equationally conservative* if it induces exactly the same derivable ground (i.e., closed) equalities on the old syntax as the original equational theory. In this paper, we remove the requirement for including the old set of axioms in the extended equational theory. We refer to such extensions as *equationally conservative ground-extensions*. This relaxation is motivated by the fact that in many extensions, such as those of [2, 18, 20], for some axioms, only all closed derivable equalities on the old syntax are kept and the axioms

themselves are removed. This may be due to two reasons: the old axioms do not hold with respect to the newly introduced operators or they (or all their closed instantiations) are derivable from the new axioms. For example, some of the old axioms of [2, 18, 20] (and also Section 3 of this paper) are not sound in the extended language when they are instantiated by terms from the extended syntax. For the relaxed notion of equational conservativity, we present similar meta-theorems as for the old notions in [1, 21, 23]. Operational conservativity is usually considered as a means for equational conservativity and we show that our notion of orthogonality leads to equational conservativity in the same way as operational conservativity does (no matter which notion of equational conservativity is chosen, the traditional notion or the relaxed one).

The rest of this paper is structured as follows. Section 2 gives the basic definitions about Structural Operational Semantics, Transition System Specification (TSS) and equational theory. Section 3 presents the notions of operational and equational conservativity and gives sufficient conditions for proving operational conservativity. Orthogonality and related notions are defined in Section 4. Subsequently, Section 5 defines sufficient conditions for orthogonality. In the same section, we also present theorems establishing the link between orthogonality and equational conservativity. Finally, Section 6 summarizes the results and presents future directions. In each section, we provide abstract and concrete examples from the area of process algebra to motivate the definitions and illustrate the results.

2 Preliminaries

2.1 Structural Operational Semantics

Structural Operational Semantics [17] is a logical way of defining operational semantics which has found lots of applications in different areas of computer science. A semantic specification in the style of SOS, called a *Transition System Specification (TSS)*, consists of a number of deduction rules which specify the possibility of a transition (in the conclusion of the rules) in terms of the (im)possibility of other transitions (in the premises). Predicates on states are other possible ingredients of TSS's which can both be defined in the conclusion of the rules and used in the premises. Predicates can always be coded as transitions with dummy right-hand sides (cf. [22]) and thus, we do not complicate the presentation with their formal treatment. We show how to treat predicates in our settings by a concrete example in Section 5.2. Next, we formalize the rest of the concepts mentioned above.

Definition 1 (Term and Substitution) We assume that the set of *process terms*, denoted by $T(\Sigma)$ with typical members t, t', t_i, \dots , is inductively defined on a given set of *variables* $V = \{x, y, \dots\}$ and a signature Σ . A *signature* contains a number of *function symbols* (composition operators: f, g, \dots) with fixed *arities*. Function symbols with arity 0 are called *constants*. *Closed terms*, denoted by $C(\Sigma)$ with typical members p, q, p_i, \dots , are terms that do not contain variables. The set of variables appearing in term t is denoted by $\text{vars}(t)$.

A *(closed) substitution* σ replaces variables in a term with other (closed) terms. Terms *generated* by a set of terms S , denoted by $G(S)$, is the set of all terms $t' = \sigma(t)$, for some $t \in S$ and some σ such that $\forall_{x \in V} \sigma(x) \in S$. A set of terms S *covers* Σ -terms, if $C(\Sigma) \subseteq G(S)$.

A transition system specification, defined below, is a logical way of defining a transition relation on (closed) terms.

Definition 2 (Transition System Specification (TSS)) A *transition system specification* is a tuple (Σ, L, D) where Σ is a signature, L is a set of labels (with typical members l, l', l_0, \dots) and D is a set of deduction rules. For all $l \in L$, and $t, t' \in T(\Sigma)$ we define that $(t, l, t') \in \rightarrow$ and $(t, l) \notin \rightarrow$ are formulae (positive and negative, respectively). To avoid any confusion, note that $\dots \in \rightarrow$ and $\dots \notin \rightarrow$ are used as a syntactic notation and are not intended to denote the set-theoretic membership at this point. A deduction rule $dr \in D$, is defined as a tuple (H, c) where H is a set

of formulae and c is a positive formula. The formula c is called the *conclusion* and the formulae from H are called *premises*. A deduction rule with label l in its conclusion is called an l -rule.

Formulae $(t, l, t') \in \rightarrow$ and $(t, l) \notin \rightarrow$ are denoted by the more intuitive notations $t \xrightarrow{l} t'$ and $t \not\xrightarrow{l}$, respectively. We refer to t as the source of both formulae and to t' as the target of the first one. A deduction rule (H, c) is denoted by $\frac{H}{c}$ in the remainder.

The notion of closed and the concept of substitution are lifted from terms to formulae in the natural way.

Different interpretations of the transition relation, i.e., the set of closed (positive) formulae, induced by a TSS are given in the literature. In [10], an extensive overview of alternative interpretations is provided. We formulate and prove our main results in such a general way that they remain independent from the chosen interpretation and can be adopted for several existing ones. In cases where we need an explicit transition relation, we assume that this transition relation is uniquely defined by the corresponding TSS using one of the interpretations given in [10]. In such cases, we use the notation $tss \vDash \phi$ to denote that a closed positive formula ϕ is in the transition relation induced by tss .¹ If we need to go further and examine the proof of a formula in a TSS, we use the following notion of *provable transition rules*.

Definition 3 (Provable Transition Rules) A deduction rule $\frac{N}{c}$ is called a *transition rule* if c is a closed positive formula and N is a set of closed negative formulae.

A transition rule $\frac{N}{c}$ is provable with respect to tss , denoted by $tss \vdash \frac{N}{c}$ if and only if there exists a well-founded upwardly branching proof tree with nodes labelled by formulae such that:

- the root of the proof tree is labelled by c ;
- if the label of a node q , denoted by ψ , is a positive formula and $\{\psi_i \mid i \in I\}$ is the set of labels of the nodes directly above q , then there is a deduction rule $\frac{\{\chi_i \mid i \in I\}}{\chi}$ in tss (N.B. χ_i can be a positive or a negative formula) and a substitution σ such that $\sigma(\chi) = \psi$ and for all $i \in I$, $\sigma(\chi_i) = \psi_i$;
- for all negative formulae χ , $\chi \in N$ if and only if χ is a leaf of the proof tree.

This technique will enable us to apply our results to various existing interpretations (following [8]). Particularly, if $tss \vdash \frac{N}{\phi}$ and tss induces a unique transition relation using one of the existing interpretations, it always follows that $tss \vDash \phi$.

One criterium that guarantees the existence and uniqueness of a transition relation associated with a TSS is the following concept of (strict) stratification, which we use for other purposes in this paper, as well.

Definition 4 (Stratification [12]) A TSS tss is stratified by a function \mathcal{S} from closed positive formulae to an ordinal if and only if for all deduction rules in tss of the following form:

$$\frac{\{t_i \xrightarrow{l_i} t'_i \mid i \in I\} \quad \{t_j \xrightarrow{l_j} t''_j \mid j \in J\}}{t \xrightarrow{l} t'}$$

and for all closed substitutions σ , $\forall_{i \in I} \mathcal{S}(\sigma(t_i \xrightarrow{l_i} t'_i)) \leq \mathcal{S}(\sigma(t \xrightarrow{l} t'))$ and $\forall_{j \in J} \mathcal{S}(\sigma(t_j \xrightarrow{l_j} t''_j)) < \mathcal{S}(\sigma(t \xrightarrow{l} t'))$, for all terms t'' . If the measure decreases also from the conclusion to the positive premises, then tss is *strictly stratified* by \mathcal{S} .

¹In this paper, we only consider transition relation and behavioral equalities on closed terms. In [19], the same concepts are extended to open terms. Following the approach of [19], our results can be extended to open terms, too.

The following example illustrates the concepts defined in this section.

Example 1 (Minimal Process Algebra (*MPA*)) Consider the following deduction rules defined on a signature with a constant δ , a family of unary operators $a\cdot$ (for all $a \in A$, where A is a given set of atomic actions) and a binary operator $\cdot + \cdot$. The labels of transitions are $a \in A$.

$$\text{(a)} \frac{}{a \cdot x \xrightarrow{a} x} \quad (a \in A) \quad \text{(c0)} \frac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'} \quad \text{(c1)} \frac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$$

This TSS (called tss_m in the remainder) is supposed to define a transition relation for the Minimal Process Algebra (*MPA*) of [2], simplified here by removing the concept of termination, which we use as our running example in the remainder. Deduction rules of *MPA* are (strictly) stratified using a measure of size on the terms in the source of formulae and it defines a unique transition relation by all possible interpretations. The following transitions are among those included in this relation: $tss_m \models (a \cdot \delta) + \delta \xrightarrow{a} \delta$ and $tss_m \models a \cdot (\delta + a \cdot \delta) \xrightarrow{a} \delta + a \cdot \delta$ which are all provable in tss_m using an empty set of negative premises.

2.2 Equational Theory

Equational theories play a central role in process algebras. They capture the basic intuition behind the algebra, and the models of the algebra are expected to respect this intuition (e.g., the models induced by operational semantics). To establish a reasonable link between the operational model and the equational theory of the algebra, a notion of behavioral equality is needed. This notion captures when two syntactic terms show the “same behavior” and thus they should belong to the same equivalence class. There is a spectrum of notions of behavioral equality in the literature [9, 11]. As we show in this paper, our results are valid for a wide range of notions in this spectrum.

Getting back to the equational side of the algebra, the notion of behavioral equivalence should ideally coincide with the closed derivations of the equational theory. One side of this coincidence is captured by the soundness theorem which states that all closed derivations of the equational theory are indeed valid with respect to the particular notion of behavioral equality. The other side of the coincidence, called completeness, phrases that all induced behavioral equalities are derivable from the equational theory, as well. These concepts are formalized in the remainder.

Definition 5 (Equational Theory) An *equational theory* or *axiomatization* (Σ, E) is a set of equalities E on a signature Σ of the form $t = t'$, where $t, t' \in T(\Sigma)$. A closed instance $p = p'$, for some $p, p' \in C(\Sigma)$, is derivable from E , denoted by $E \vdash p = p'$ if and only if it is in the smallest congruence relation induced by the equalities of E .

An equational theory (Σ, E) is *sound* with respect to a TSS tss (also on signature Σ) and a particular notion of behavioral equality \sim if and only if for all $p, p' \in C(\Sigma)$, if $E \vdash p = p'$, then it holds that $tss \vdash p \sim p'$. It is *complete* if the implication holds in the other direction.

An equational theory E on Σ *eliminates* 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 $E \vdash p = p'$.

The following example illustrates the idea of equational theory.

Example 2 (*MPA*: Equational Theory) Consider the Minimal Process Algebra of Example 1. The following is an axiomatization of *MPA* [2].

$$x + y = y + x \quad x + (y + z) = (x + y) + z \quad x + x = x \quad x + \delta = x$$

It is well-known that this axiomatization is sound and complete with respect to tss_m given in Example 1 and strong bisimilarity as the notion of behavioral equivalence (see, for example, [15]). The following are examples of derivable equalities from the above axiomatization: $(a \cdot \delta) + \delta = a \cdot \delta$ and $(a \cdot \delta) + a \cdot \delta = a \cdot \delta$.

3 Operational and Equational Conservativity

In this section, we define different concepts regarding language extensions. To extend a language defined by a TSS, one may have to combine an existing signature with a new one. However, not all signatures can be combined into one as the arities of the function symbols may clash. To prevent this, we define two signatures to be *consistent* when they agree on the arity of the shared function symbols. In the remainder, we always assume that extended and extending TSS's are consistent. The following definition formalizes the concept of operational extension.

Definition 6 (Extension of a TSS) Consider TSS's $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$. The extension of tss_0 with tss_1 , denoted by $tss_0 \cup tss_1$, is defined as $(\Sigma_0 \cup \Sigma_1, L_0 \cup L_1, D_0 \cup D_1)$.

Next, we define when an extension of a TSS is called operationally conservative.

Definition 7 (Operational Conservativity [21]) Consider TSS's $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$. If $\forall_{p \in C(\Sigma_0)} \forall_{p' \in C(\Sigma_0 \cup \Sigma_1)} \forall_{l \in L_0 \cup L_1} tss_0 \cup tss_1 \models p \xrightarrow{l} p' \Leftrightarrow tss_0 \models p \xrightarrow{l} p'$, then $tss_0 \cup tss_1$ is an *operationally conservative extension* of tss_0 .

Note that in the above definition, the labels and the targets of the transitions are taken from the extended TSS and thus, any new transition of the old syntax, even with a new label or a new target is prohibited. The following example illustrates the idea of extending TSS's and the concept of operational conservativity.

Example 3 (Timed-MPA: Operational Semantics) Consider the following deduction rules (divided into three parts) which are defined on a signature with two constants δ and $\underline{\delta}$, a unary function symbol $\underline{\sigma}_-$, two families of unary function symbols \underline{a}_- and $\underline{\sigma}_-$ (for all $a \in A$) and a binary function symbol $_ + _$. The set of labels of the TSS is $A \cup \{1\}$ (for $1 \notin A$).

$$\begin{aligned}
 (1) \quad & \text{(ua)} \frac{}{\underline{a}.x \xrightarrow{a} x} \quad \text{(td)} \frac{}{\underline{\sigma}.x \xrightarrow{1} x} \\
 (2) \quad & \text{(tc0)} \frac{x \xrightarrow{1} x' \quad y \xrightarrow{1} y'}{x + y \xrightarrow{1} x' + y'} \quad \text{(tc1)} \frac{x \xrightarrow{1} x' \quad y \xrightarrow{1} y'}{x + y \xrightarrow{1} x'} \quad \text{(tc2)} \frac{y \xrightarrow{1} y' \quad x \xrightarrow{1} y'}{x + y \xrightarrow{1} y'} \\
 (3) \quad & \text{(ta)} \frac{}{a.x \xrightarrow{1} a.x} \quad \text{(d)} \frac{}{\delta \xrightarrow{1} \delta}
 \end{aligned}$$

The above TSS, which we call tss_t defines the aspect of timing in terms of new time-transitions $\xrightarrow{1}$ and it is added in [2] to tss_m in Example 1 to define a relative-discrete-time extension of MPA. The intuition behind the new underlined function symbols (\underline{a}_- and $\underline{\sigma}_-$) is that they are not delayable in time and should take their (respectively action and time) transitions immediately. Addition of the first and/or the second parts of the above TSS (each or both) to tss_m results in an operationally conservative extension of the latter as the newly added transitions will be restricted to the new syntax. (Note that in the first and second parts, there is no rule about timed transition of constants in old syntax.) We prove this claim formally as an instance of a meta-theorem in the rest of this section. However, the addition of part (3) violates the conservativity of the extension as it adds time-transitions ($\xrightarrow{1}$) to the behavior of terms from the old syntax. For example, in combination with part (2), it allows for transitions such as $tss_m \cup tss_t \models a.\delta \xrightarrow{1} a.\delta$ and $tss_m \cup tss_t \models (a.\delta) + \delta \xrightarrow{1} (a.\delta) + \delta$, all of which are prohibited by the original TSS and thus are considered harmful from the operational conservativity point of view.

Next, we formulate sufficient conditions to prove operational conservativity. But before that, we need a few auxiliary definitions.

Definition 8 (Source Dependency) All variables appearing in the source of the conclusion of a deduction rule are called *source dependent*. A variable of a deduction rule is *source dependent* if it appears in a target of a premise of which all the variables of the source are source dependent. A premise is *source dependent* when all the variables appearing in it are source dependent. A rule is *source dependent* when all its variables are. A TSS is *source dependent* when all its rules are.

Definition 9 (Reduced Rules) For a deduction rule $d = (H, c)$, the reduced rule with respect to a signature Σ is defined by $\rho(d, \Sigma) \doteq (H', c)$ where H' is the set of all premises from H which have a Σ -term as a source.

Theorem 1 (Operational Conservativity Meta-Theorem [8]) Given two TSS's $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$, $tss_0 \cup tss_1$ is an operationally conservative extension of tss_0 if:

1. tss_0 is source dependent;
2. for all $d \in D_1$ at least one of the following holds:
 - (a) the source of the conclusion has a function symbol in $\Sigma_0 \setminus \Sigma_1$, or
 - (b) $\rho(d, \Sigma_0)$ has a source-dependent positive premise $t \xrightarrow{l} t'$ such that $l \notin \Sigma_0$ or $t' \notin T(\Sigma_0)$.

Proof. See [8] or the proof of Theorem 4 in this paper. Theorem 1 can be instantiated as a special case of Theorem 4. □

The above meta-theorem is more general than other similar ones presented in [1, 21, 23].

Example 4 (Timed-*MPA*: Operational Conservativity, Revisited) The addition of parts (1) and (2) of tss_t in Example 3 to the tss_m of Example 1 results in an operationally conservative extension following Theorem 1: All three deduction rules of tss_m are source dependent; Rules **(ua)** and **(td)** both have a new function symbol in the source of their conclusion (i.e., $\underline{a}.$ and $\underline{\sigma}.$, respectively) and hence, satisfy condition 2a; Rules **(tc0)**, **(tc1)** and **(tc2)** all have a source-dependent positive premise with a timed-*MPA* label (1) and hence, satisfy condition 2b. Note that the reduced version of each of the deduction rules **(tc0)**, **(tc1)** and **(tc2)** is that deduction rule itself.

The following definition formalizes the concept of equationally conservative ground-extensions.

Definition 10 (Equational Conservativity) An equational theory E_1 on signature Σ_1 is an *equationally conservative ground-extension* of E_0 on Σ_0 if and only if $\Sigma_0 \subseteq \Sigma_1$ and for all $p, p' \in C(\Sigma_0)$, $E_0 \vdash p = p' \Leftrightarrow E_1 \vdash p = p'$.

It is worth mentioning that the above definition is more liberal than the notion of equational conservativity in [21] in that there, it is required that the same axioms are included in the extended equational theory (i.e., $E_0 \subseteq E_1$). In practice, some process algebras do not keep the same axioms when extending the formalism while they make sure that the ground instantiations of the old axioms with old terms indeed remain derivable (see for example, [2, 18, 20] and Example 5 in the remainder). Hence, we believe that the restriction imposed by [21] unnecessarily limits the applicability of the theory. If, for any reason, one chooses the more restricted notion of [21], the theorems concerning equational conservativity in this paper remain valid.

Example 5 (Timed-*MPA*: Equational Theory) Consider the TSS resulting from extending tss_m of Example 1 with tss_t of Example 3. The following are a set of sound and complete axioms (w.r.t. strong bisimilarity) for this TSS:

$$\begin{array}{ccccccc}
x + y = y + x & x + (y + z) = (x + y) + z & x + x = x & \delta = \underline{\sigma}.\delta \\
x + \underline{\delta} = x & (\underline{\sigma}.x) + \underline{\sigma}.y = \underline{\sigma}.(x + y) & a.x = (\underline{a}.x) + \underline{\sigma}.a.x & (a.x) + \delta = a.x
\end{array}$$

The above axiomatization underscores the fact we mentioned before. Namely, the axioms of the old system do not hold in the new system (e.g., $(\underline{a}.x) + \delta \neq \underline{a}.x$ as an instance of $x + \delta = x$) but all closed instantiations of the old axioms by the old syntax are derivable from the new set of axioms.

It can be checked that the above axiomatization of timed-*MPA* is indeed an equationally conservative ground-extension of the axiomatization of *MPA* in the sense of Definition 10. Thus, if one considers operational conservativity as a means to equational conservativity, this example already suggests the need for an extension of Definition 7. In other words, we believe that the transitions added by the extension are quite innocent and harmless to the intuition behind the original semantics, for they are added uniformly to the old syntax without changing the old behavior or violating previously valid equalities. In the next section, we formalize our idea of orthogonal extensions which caters for extensions of the above type.

4 Orthogonality

4.1 Orthogonal Extension

In this section, we define the notion of orthogonality and an instance of this notion, called *granting extensions*, which can be checked syntactically.

Definition 11 (Orthogonal Extension) Consider TSS's $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$ and a behavioral notion of equality \sim . The TSS $tss_0 \cup tss_1$ is a \sim -orthogonal extension of tss_0 when

1. $\forall_{p,p' \in C(\Sigma_0)} \forall_{l \in L_0} tss_0 \models p \xrightarrow{l} p' \Leftrightarrow tss_0 \cup tss_1 \models p \xrightarrow{l} p'$, and
2. $\forall_{p,p' \in C(\Sigma_0)} tss_0 \models p \sim p' \Leftrightarrow tss_0 \cup tss_1 \models p \sim p'$.

Our results in this paper are valid for most notions of behavioral equivalence in the literature (to be named explicitly in the remainder). The notion of *operational conservativity up to ϕ -equivalence* of [21, 5] can be seen as a variant of orthogonality which only has the second condition. To our knowledge, beyond operational conservativity results (e.g., [21]), no systematic study of these notions (i.e., orthogonality and operational conservativity up-to, including meta-theorems guaranteeing them) has been carried out.

Corollary 1 An operationally conservative extension is an orthogonal extension.

4.2 Granting Extension

Corollary 1 addresses operational conservativity as an extreme case of orthogonality which denies all new transitions from the old syntax; the other extreme is an extension which grants all new behavior to the old syntax. However, for such an extension to be orthogonal, these transitions should be made to equivalent terms from the old syntax. In particular, if we allow for self-transitions, we are able to prove orthogonality with respect to many notions of behavioral equivalence. The following definitions and the subsequent theorem substantiate these concepts.

Definition 12 (Granting Extension) Consider TSS's $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$ with disjoint labels. We call $tss_0 \cup tss_1$ a *granting extension* of tss_0 when

1. $\forall_{p,p' \in C(\Sigma_0)} \forall_{l \in L_0} tss_0 \models p \xrightarrow{l} p' \Leftrightarrow tss_0 \cup tss_1 \models p \xrightarrow{l} p'$, and
2. $\forall_{p \in C(\Sigma_0)} \forall_{p' \in C(\Sigma_0 \cup \Sigma_1)} \forall_{l \in L_1} tss_0 \cup tss_1 \models p \xrightarrow{l} p' \Leftrightarrow p = p'$.

The above definition states that granting extensions keep the old transitions on the old terms intact and only add self-transitions with all of the new labels to old terms. The above definition does not make any statement about the transitions on the new terms, i.e., terms from $T(\Sigma_0 \cup \Sigma_1) \setminus T(\Sigma_0)$. We are doubtful whether any meaningful relaxation of Definition 12 would be at all possible that allows for anything coarser than syntactic equality on the old terms involved in (the left- or the right-hand side of) the new transitions and still can be captured by simple syntactic checks (which is our aim in this paper, even if one confines him/herself to \sim -orthogonality for a particular \sim). This suggests that to formulate syntactic criteria for proving orthogonality, we have to resort to one of the two extremes (operational conservativity or granting extensions). Admitting that these two extremes need to be combined in some way to reach a reasonable balance, we define the sufficient criteria for this mixture to be orthogonal in the next section. Next, we show that granting extensions are indeed \sim -orthogonal extension for most notions of behavioral equivalence \sim .

Theorem 2 Consider two TSS's tss_0 and tss_1 where tss_1 is a granting extension of tss_0 . Let \sim be any of the following notions of behavioral equivalence (cf. [11, 9] and the proof presented next, for details about these notions):

1. trace equivalence $=_T$,
2. failures equivalence $=_F$,
3. ready equivalence $=_R$,
4. failure trace equivalence $=_{FT}$,
5. ready trace equivalence $=_{RT}$,
6. simulation equivalence \Leftrightarrow ,
7. ready simulation equivalence $=_{RS}$,
8. weak bisimilarity \Leftrightarrow_w ,
9. branching bisimilarity \Leftrightarrow_b ,
10. rooted branching bisimilarity \Leftrightarrow_{rb} ,
11. rooted weak bisimilarity \Leftrightarrow_{rw} ,
12. bisimulation equivalence \Leftrightarrow ,

then tss_1 is a \sim -orthogonal extension of tss_0 .

Proof. Let $tss_x \doteq (\Sigma_x, L_x, D_x)$ and let $L' \doteq L_1 \setminus L_0$. Copying Definition 12, we have:

1. $\forall_{p,p' \in C(\Sigma_0)} \forall_{l \in L_0} tss_0 \models p \xrightarrow{l} p' \Leftrightarrow tss_1 \models p \xrightarrow{l} p'$, and
2. $\forall_{p \in C(\Sigma_0)} \forall_{p' \in C(\Sigma_1)} \forall_{l' \in L'} tss_1 \models p \xrightarrow{l'} p' \Leftrightarrow p = p'$.

The first item above is the same as the first item in Definition 11 of orthogonality. Hence, we have to prove the statement

$$\forall_{p,q \in C(\Sigma_0)} tss_0 \models p \sim q \Leftrightarrow tss_1 \models p \sim q$$

for all of the following notions of behavioral equivalence \sim mentioned in the theorem.

1. trace equivalence $=_T$: We start with the following auxiliary definitions

Definition 13 Let L^* be the set of all traces that can be generated from labels L (including the empty trace ϵ). Given a trace $\sigma \in L^*$ and a set of labels L' the *granting extension of σ with L'* , denoted by $\sigma \uparrow L'$, is the smallest set of traces that satisfies $\forall \sigma_0, \sigma_1 \in (L \cup L')^*$ and $\forall l \in L'$:

- (a) $\sigma \in \sigma \uparrow L'$;
- (b) $\sigma_0 \sigma_1 \in \sigma \uparrow L' \Rightarrow \sigma_0(l)\sigma_1 \in \sigma \uparrow L'$.

where juxtaposition denotes concatenation. For a set of traces TR , $TR \uparrow L' \doteq \bigcup_{\sigma \in TR} \sigma \uparrow L'$. Similarly, for a trace σ on labels L , the trace $\sigma \downarrow L'$ is defined inductively by:

$$\epsilon \downarrow L' \doteq \epsilon, \quad (l)\sigma \downarrow L' \doteq \begin{cases} \sigma \downarrow L' & l \in L', \\ (l)(\sigma \downarrow L') & l \notin L'. \end{cases}$$

For a set of traces TR , $TR \downarrow L' \doteq \{\sigma \downarrow L' \mid \sigma \in TR\}$.

Corollary 2 Consider sets of traces TR and TR' both defined on a set of labels L .

- (a) If $TR = TR'$ then for an arbitrary L' , $TR \downarrow L' = TR' \downarrow L'$ and $TR \uparrow L' = TR' \uparrow L'$;
- (b) For a set L' disjoint from L , $(TR \uparrow L') \downarrow L' = TR$.

Lemma 1 Consider sets of traces TR and TR' both defined on a set of labels L and a set L' disjoint from L . $TR = TR'$ if and only if $TR \uparrow L' = TR' \uparrow L'$.

Proof. The implication from left to right follows trivially from the definition of $TR \uparrow L'$ (see the first item of Corollary 2). Then, it remains to prove $TR = TR'$ assuming the left-to-right implication and $TR \uparrow L' = TR' \uparrow L'$. It follows from the first item of Corollary 2 that $(TR \uparrow L') \downarrow L' = (TR' \uparrow L') \downarrow L'$ and from the second item of the same corollary, $TR = TR'$. \square

Definition 14 Given a transition relation $\rightarrow \subseteq C(\Sigma) \times L \times C(\Sigma)$, the *reflexive and transitive closure of \rightarrow* , denoted by $\rightarrow^* \subseteq C(\Sigma) \times L^* \times C(\Sigma)$ is defined as the smallest relation satisfying the following constraints: $\forall p, p', p'' \in C(\Sigma)$

- (a) $p \xrightarrow{\epsilon}^* p$;
- (b) $p \xrightarrow{l} p' \Rightarrow p \xrightarrow{(l)}^* p'$;
- (c) $p \xrightarrow{l} p' \wedge p' \xrightarrow{\sigma}^* p'' \Rightarrow p \xrightarrow{(l)\sigma}^* p''$

Let $tss = (\Sigma, L, D)$ be a TSS. The set of traces in tss originating from $p \in C(\Sigma)$, denoted by $TR(tss, p)$ is the smallest set satisfying for all $p' \in C(\Sigma)$ and $\sigma \in L^*$: if $tss \models p \xrightarrow{\sigma}^* p'$ then $\sigma \in TR(tss, p)$. The processes p and q are trace equivalent w.r.t. TSS tss , notation $tss \models p =_T q$, iff $TR(tss, p) = TR(tss, q)$.

Corollary 3 Let tss_0 and tss_1 be the TSS's defined above. For all $p \in C(\Sigma_0)$, $TR(tss_1, p) = TR(tss_0, p) \uparrow L'$.

We now have $tss_0 \models p =_T q$ iff, by definition, $TR(tss_0, p) = TR(tss_0, q)$ iff, by Lemma 1, $TR(tss_0, p) \uparrow L' = TR(tss_0, p) \uparrow L'$ iff, by Corollary 3, $TR(tss_1, p) = TR(tss_1, q)$ iff, by definition, $tss_1 \models p =_T q$.

2. Failures equivalence $=_F$:

Definition 15 Let $tss = (\Sigma, L, D)$ be a TSS. A pair $(\sigma, X) \in L^* \times \mathcal{P}(L)$ is a failure pair of $p \in C(\Sigma)$ originating from tss if $tss \models p \xrightarrow{\sigma} *p'$ for some p' such that for all $l \in X$, $tss_0 \models p \not\overset{l}{\rightarrow}$. The set of failure pairs in tss originating from $p \in C(\Sigma)$, denoted by $FP(tss, p)$ is the set containing all failure pairs of p originating from tss . The processes p and q are failures equivalent w.r.t. TSS tss , notation $tss \models p =_F q$, iff $FP(tss, p) = FP(tss, q)$.

For a set of failure pairs FP and a set of labels L , we define $FP \downarrow L \doteq \{(\sigma \downarrow L, X) \mid (\sigma, X) \in FP\}$ and $FP \uparrow L \doteq \{(\sigma', X) \mid \sigma' \in \sigma \uparrow L \wedge (\sigma, X) \in FP\}$.

Corollary 4 Consider sets of failure pairs FP and FP' both defined on a set of labels L .

- (a) If $FP = FP'$ then for an arbitrary L' , $FP \downarrow L' = FP' \downarrow L'$ and $FP \uparrow L' = FP' \uparrow L'$;
- (b) For a set L' disjoint from L , $(FP \uparrow L') \downarrow L' = FP$.

Lemma 2 Consider sets of failure pairs FP and FP' both defined on a set of labels L and a set L' disjoint from L . $FP = FP'$ if and only if $FP \uparrow L' = FP' \uparrow L'$.

Proof. The implication from left to right follows trivially from the definition of $FP \uparrow L'$ (see the first item of Corollary 4). Then, it remains to prove $FP = FP'$ assuming the left-to-right implication and $FP \uparrow L' = FP' \uparrow L'$. It follows from the first item of Corollary 4 that $(FP \uparrow L') \downarrow L' = (FP' \uparrow L') \downarrow L'$ and from the second item of the same corollary, $FP = FP'$. \square

Consider a failure pair $(\sigma, X) \in FP(tss_0, p)$. This means that $\sigma \in TR(tss_0, p)$ and for some p' such that $tss_0 \models p \xrightarrow{\sigma} *p'$ and for all $l \in X$, $tss_0 \models p \not\overset{l}{\rightarrow}$. Then, for all $\sigma' \in \sigma \uparrow L'$, (σ', X) is a failure pair originating from p in tss_1 since $\sigma' \in TR(tss_1, p)$ and the blocked transitions from X remain blocked since $L \cap L' = \emptyset$ and $X \subseteq L$ and hence $X \cap L' = \emptyset$, i.e., the added L' -transitions do not change the status of X . Conversely, if a pair $(\sigma, X) \in FP(tss_1, p)$ then $(\sigma \downarrow L', X)$ is a failure pair of p in tss_0 since first, $\sigma \downarrow L' \in TR(tss_0, p)$ (see the previous item) and second, X may not contain any label from L' since all transitions with a label from L' are enabled in tss_1 .

Corollary 5 Let tss_0 and tss_1 be the TSS's defined above. For all $p \in C(\Sigma_0)$, $FP(tss_1, p) = FP(tss_0, p) \uparrow L'$.

We now have $tss_0 \models p =_F q$ iff, by definition, $FP(tss_0, p) = FP(tss_0, q)$ iff, by Lemma 2, $FP(tss_0, p) \uparrow L' = FP(tss_0, q) \uparrow L'$ iff, by Corollary 5, $FP(tss_1, p) = FP(tss_1, q)$ iff, by definition, $tss_1 \models p =_F q$.

3. Ready equivalence: Similar to the previous two items; the only difference is that L' is always a member of ready sets in tss_1 and should be removed from the ready sets when projecting from the ready traces in tss_1 to the ready traces in tss_0 .

Definition 16 Let $tss = (\Sigma, L, D)$ be a TSS. A pair $(\sigma, X) \in L^* \times \mathcal{P}(L)$ is a ready pair of $p \in C(\Sigma)$ originating from tss if $tss \models p \xrightarrow{\sigma} *p'$ for some p' such that $X = \{l \in L \mid p' \overset{l}{\rightarrow}\}$. The set of ready pairs in tss originating from $p \in C(\Sigma)$, denoted by $R(tss, p)$ is the set containing all ready pairs of p originating from tss . The processes p and q are ready equivalent w.r.t. TSS tss , notation $tss \models p =_R q$, iff $R(tss, p) = R(tss, q)$.

For a set of ready pairs R and a set of labels L , we define $R \downarrow L \doteq R \downarrow L = \{(\sigma \downarrow L, X \setminus L) \mid (\sigma, X) \in R\}$ and $R \uparrow L \doteq \{(\sigma', X \cup L) \mid \sigma' \in \sigma \uparrow L \wedge (\sigma, X) \in R\}$.

Corollary 6 Consider sets of ready pairs R and R' both defined on a set of labels L .

- (a) If $R = R'$ then for an arbitrary L' , $R \downarrow L' = R' \downarrow L'$ and $R \uparrow L' = R' \uparrow L'$;
- (b) For a set L' disjoint from L , $(R \uparrow L') \downarrow L' = R$.

Lemma 3 Consider sets of ready pairs R and R' both defined on a set of labels L and a set L' disjoint from L . $R = R'$ if and only if $R \uparrow L' = R' \uparrow L'$.

Proof. The implication from left to right follows trivially from the definition of $R \uparrow L'$ (see the first item of Corollary 6). Then, it remains to prove $R = R'$ assuming the left-to-right implication and $R \uparrow L' = R' \uparrow L'$. It follows from the first item of Corollary 6 that $(R \uparrow L') \downarrow L' = (R' \uparrow L') \downarrow L'$ and from the second item of the same corollary, $R = R'$. \square

Consider a ready pair $(\sigma, X) \in R(tss_0, p)$. This means that $\sigma \in TR(tss_0, p)$ and for some p' such that $tss_0 \models p \xrightarrow{\sigma} p'$ and $X = \{l \in L \mid p' \xrightarrow{l}\}$. Then, for all $\sigma' \in \sigma \uparrow L'$, $(\sigma', X \cup L') \in R(tss_1, p)$ since $\sigma' \in TR(tss_1, p)$ and the transitions from X remain enabled and the transitions from L' are also enabled since tss_1 is a granting extension of tss_0 . Conversely, if a pair $(\sigma, X) \in R(tss_1, p)$ then $(\sigma \downarrow L', X \setminus L') \in R(tss_0, p)$.

Corollary 7 Let tss_0 and tss_1 be the TSS's defined above. For all $p \in C(\Sigma_0)$, $R(tss_1, p) = R(tss_0, p) \uparrow L'$.

We now have $tss_0 \models p =_{\text{R}} q$ iff, by definition, $R(tss_0, p) = R(tss_0, q)$ iff, by Lemma 3, $R(tss_0, p) \uparrow L' = R(tss_0, p) \uparrow L'$ iff, by Corollary 7, $R(tss_1, p) = R(tss_1, q)$ iff, by definition, $tss_1 \models p =_{\text{R}} q$.

4. Failure trace equivalence $=_{\text{FT}}$: Similar to the above item; elements of L' cannot be present in the refusal sets in the failure traces of tss_0 and hence one can repeat the above reasoning to prove the coincidence of failure traces.

Definition 17 Let $tss = (\Sigma, L, D)$ be a TSS. The *refusal relation* $\dashv\vdash \subseteq C(\Sigma) \times \mathcal{P}(L) \times C(\Sigma)$ is defined as $p \dashv\vdash q$ iff $p = q$ and $p \xrightarrow{l}$ for all $l \in X$. The *failure trace relation* $\succ \subseteq C(\Sigma) \times (L \cup \mathcal{P}(L))^* \times C(\Sigma)$ is defined as the reflexive transitive closure of the transition relation \rightarrow and the refusal relation $\dashv\vdash$. $\sigma \in (L \cup \mathcal{P}(L))^*$ is a *failure trace* of a process p w.r.t. tss if $p \xrightarrow{\sigma}$. Let $FT(tss, p)$ denote the failure traces of p w.r.t. tss . The processes p and q are failure trace equivalent w.r.t. tss , notation $tss \models p =_{\text{FT}} q$, iff $FT(tss, p) = FT(tss, q)$.

For a failure trace σ on labels L , the failure trace $\sigma \downarrow L'$ is defined inductively by:

$$\begin{aligned} \epsilon \downarrow L' &\doteq \epsilon, \\ (l)\sigma \downarrow L' &\doteq \begin{cases} \sigma \downarrow L' & l \in L', \\ (l)(\sigma \downarrow L') & l \notin L', \end{cases} \\ (X)\sigma \downarrow L' &\doteq (X \setminus L')(\sigma \downarrow L'). \end{aligned}$$

For a set of failure traces FT , $FT \downarrow L' \doteq \{\sigma \downarrow L' \mid \sigma \in FT\}$. Given a failure trace σ and a set of labels L' the *granting extension of σ with L'* , denoted by $\sigma \uparrow L'$, is the smallest set of traces that satisfies $\forall \sigma_0, \sigma_1$ and $\forall l \in L'$:

- (a) $\sigma \in \sigma \uparrow L'$;
- (b) $(\sigma_0)(\sigma_1) \in \sigma \uparrow L' \Rightarrow (\sigma_0)(l)(\sigma_1) \in \sigma \uparrow L'$.

For a set of failure traces FT , $FT \uparrow L' \doteq \bigcup_{\sigma \in FT} \sigma \uparrow L'$.

Corollary 8 Consider sets of failure traces FT and FT' both defined on a set of labels L .

- (a) If $FT = FT'$ then for an arbitrary L' , $FT \downarrow L' = FT' \downarrow L'$ and $FT \uparrow L' = FT' \uparrow L'$;

(b) For a set L' disjoint from L , $(FT \uparrow L') \downarrow L' = FT$.

Lemma 4 Consider sets of failure traces FT and FT' both defined on a set of labels L and a set L' disjoint from L . $FT = FT'$ if and only if $FT \uparrow L' = FT' \uparrow L'$.

Proof. The implication from left to right follows trivially from the definition of $FT \uparrow L'$ (see the first item of Corollary 8). Then, it remains to prove $FT = FT'$ assuming the left-to-right implication and $FT \uparrow L' = FT' \uparrow L'$. It follows from the first item of Corollary 8 that $(FT \uparrow L') \downarrow L' = (FT' \uparrow L') \downarrow L'$ and from the second item of the same corollary, $FT = FT'$. \square

Corollary 9 Let tss_0 and tss_1 be the TSS's defined above. For all $p \in C(\Sigma_0)$, $FT(tss_1, p) = FT(tss_0, p) \uparrow L'$.

We now have $tss_0 \models p =_{\text{FT}} q$ iff, by definition, $FT(tss_0, p) = FT(tss_0, q)$ iff, by Lemma 4, $FT(tss_0, p) \uparrow L' = FT(tss_0, p) \uparrow L'$ iff, by Corollary 9, $FT(tss_1, p) = FT(tss_1, q)$ iff, by definition, $tss_1 \models p =_{\text{FT}} q$.

5. Ready trace equivalence: Again similar to item 1 but the same observation as in item 3 should be noted along the ready traces.

Definition 18 Let $tss = (\Sigma, L, D)$ be a TSS. The *ready trace relation* $\rightsquigarrow_{\subseteq} C(\Sigma) \times (L \cup \mathcal{P}(L))^* \times C(\Sigma)$ is defined recursively as follows: for $p, q, r \in C(\Sigma)$, $l \in L$ and $X \subseteq L$

- $p \xrightarrow{\epsilon} p$;
- $p \xrightarrow{l} q$ implies $p \xrightarrow{(l)} q$;
- $p \xrightarrow{X} p$ in case $X = \{l \in L \mid p \xrightarrow{l}\}$;
- if $p \xrightarrow{\sigma} q$ and $q \xrightarrow{\sigma'} r$, then $p \xrightarrow{\sigma\sigma'} r$.

The sequence $\sigma \in (L \cup \mathcal{P}(L))^*$ is a *ready trace* of a process p w.r.t. tss if $p \xrightarrow{\sigma}$. Let $RT(tss, p)$ denote the ready traces of p w.r.t. tss . The processes p and q are ready trace equivalent w.r.t. tss , notation $tss \models p =_{\text{RT}} q$, iff $RT(tss, p) = RT(tss, q)$.

For a ready trace σ on labels L , the ready trace $\sigma \downarrow L'$ is defined inductively by:

$$\begin{aligned} \epsilon \downarrow L' &\doteq \epsilon, \\ (l)\sigma \downarrow L' &\doteq \begin{cases} \sigma \downarrow L' & l \in L', \\ (l)(\sigma \downarrow L') & l \notin L', \end{cases} \\ (X)\sigma \downarrow L' &\doteq (X \setminus L')(\sigma \downarrow L'). \end{aligned}$$

For a set of ready traces RT , $RT \downarrow L' \doteq \{\sigma \downarrow L' \mid \sigma \in RT\}$. Given a ready trace σ and a set of labels L' the *granting extension of σ with L'* , denoted by $\sigma \uparrow L'$, is the smallest set of traces that satisfies $\forall_{\sigma_0, \sigma_1}$ and $\forall_{l \in L'}$:

- (a) $\sigma \in (\sigma \sqcup L') \uparrow L'$;
- (b) $(\sigma_0)(\sigma_1) \in \sigma \uparrow L' \Rightarrow (\sigma_0)(l)(\sigma_1) \in \sigma \uparrow L'$;

where \sqcup is defined inductively by:

$$\epsilon \sqcup L \doteq \epsilon, \quad ((l)\sigma) \sqcup L \doteq (l)(\sigma \sqcup L), \quad ((X)\sigma) \sqcup L \doteq (X \cup L)(\sigma \sqcup L).$$

For a set of ready traces RT , $RT \uparrow L' \doteq \bigcup_{\sigma \in RT} \sigma \uparrow L'$.

Corollary 10 Consider sets of ready traces RT and RT' both defined on a set of labels L .

- (a) If $RT = RT'$ then for an arbitrary L' , $RT \downarrow L' = RT' \downarrow L'$ and $RT \uparrow L' = RT' \uparrow L'$;
- (b) For a set L' disjoint from L , $(RT \uparrow L') \downarrow L' = RT$.

Lemma 5 Consider sets of ready traces RT and RT' both defined on a set of labels L and a set L' disjoint from L . $RT = RT'$ if and only if $RT \uparrow L' = RT' \uparrow L'$.

Proof. The implication from left to right follows trivially from the definition of $RT \uparrow L'$ (see the first item of Corollary 10). Then, it remains to prove $RT = RT'$ assuming the left-to-right implication and $RT \uparrow L' = RT' \uparrow L'$. It follows from the first item of Corollary 10 that $(RT \uparrow L') \downarrow L' = (RT' \uparrow L') \downarrow L'$ and from the second item of the same corollary, $RT = RT'$. \square

Corollary 11 Let tss_0 and tss_1 be the TSS's defined above. For all $p \in C(\Sigma_0)$, $RT(tss_1, p) = RT(tss_0, p) \uparrow L'$.

We now have $tss_0 \models p =_{RT} q$ iff, by definition, $RT(tss_0, p) = RT(tss_0, q)$ iff, by Lemma 5, $RT(tss_0, p) \uparrow L' = RT(tss_0, p) \uparrow L'$ iff, by Corollary 11, $RT(tss_1, p) = RT(tss_1, q)$ iff, by definition, $tss_1 \models p =_{RT} q$.

6. Simulation equivalence \rightleftharpoons :

Definition 19 Let $tss = (\Sigma, L, D)$ be a TSS. A *simulation* w.r.t. tss is a binary relation R on processes, satisfying, for $l \in L$: if pRq and $tss \models p \xrightarrow{l} p'$, then $q \xrightarrow{l} q'$ and $p'Rq'$ for some q' . The processes p and q are simulation equivalent or similar w.r.t. TSS tss , notation $tss \models p \rightleftharpoons q$, iff there exist a simulation R such that pRq and a simulation R' such that $qR'p$.

Suppose that $tss_0 \models p \rightleftharpoons q$. Then there exist simulations R and R' such that pRq and $qR'p$. It is very easy to check that $R \cup Id$ and $R' \cup Id$ with $Id = \{(p, p) \mid p \in C(\Sigma_0)\}$ are simulations with respect to tss_1 and hence, $tss_1 \models p \rightleftharpoons q$.

For the inclusion in the other direction, suppose that, $tss_1 \models p \rightleftharpoons q$. Then, there exist simulations R and R' w.r.t. tss_1 such that pRq and $qR'p$. One can easily verify that $R \cap (C(\Sigma_0) \times C(\Sigma_0))$ and $R' \cap (C(\Sigma_0) \times C(\Sigma_0))$ are both simulations w.r.t. tss_0 and hence $tss_0 \models p \rightleftharpoons q$.

7. Ready simulation equivalence $=_{RS}$:

Definition 20 Let $tss = (\Sigma, L, D)$ be a TSS. A *ready simulation* w.r.t. tss is a binary relation R on processes, satisfying, for $l \in L$: (1) if pRq and $tss \models p \xrightarrow{l} p'$, then $q \xrightarrow{l} q'$ and $p'Rq'$ for some q' , and (2) if pRq , then $p \xrightarrow{l'}$ iff $q \xrightarrow{l'}$. The processes p and q are ready simulation equivalent or ready similar w.r.t. TSS tss , notation $tss \models p =_{RS} q$, iff there exist a ready simulation R such that pRq and a ready simulation R' such that $qR'p$.

Suppose that $tss_0 \models p =_{RS} q$. Then there exist ready simulations R and R' such that pRq and $qR'p$. It is very easy to check that $R \cup Id$ and $R' \cup Id$ with $Id = \{(p, p) \mid p \in C(\Sigma_0)\}$ are ready simulations with respect to tss_1 and hence, $tss_1 \models p =_{RS} q$.

For the inclusion in the other direction, suppose that, $tss_1 \models p =_{RS} q$. Then, there exist ready simulations R and R' w.r.t. tss_1 such that pRq and $qR'p$. One can easily verify that $R \cap (C(\Sigma_0) \times C(\Sigma_0))$ and $R' \cap (C(\Sigma_0) \times C(\Sigma_0))$ are both ready simulations w.r.t. tss_0 and hence $tss_0 \models p =_{RS} q$.

8. Weak bisimulation equivalence \leftrightarrow_w :

Definition 21 Let $tss = (\Sigma, L, D)$ be a TSS. A *weak bisimulation* w.r.t. tss is a symmetric binary relation R on processes, satisfying, for $l \in L$: if pRq and $tss \models p \xrightarrow{l} p'$, then either $l = \tau$ and $p'Rq$, or $q \Longrightarrow q_1 \xrightarrow{l} q_2 \Longrightarrow q'$ and $p'Rq'$ for some q_1, q_2, q' . Here \Longrightarrow denotes the reflexive transitive closure of $\xrightarrow{\tau}$. The processes p and q are weakly bisimulation equivalent or weakly bisimilar w.r.t. TSS tss , notation $tss \models p \leftrightarrow_w q$, iff there exist a weak bisimulation R such that pRq .

Suppose that for $p, q \in C(\Sigma_0)$, $tss_0 \models p \leftrightarrow_w q$. Then there exists a weak bisimulation R such that pRq . We claim that $R \cup Id$ (for Id defined in the previous item) is a weak bisimulation w.r.t. tss_1 . It is trivial to see that the pairs from Id respect the requirements for a weak bisimulation. So it remains to verify this for pairs pRq . In tss_1 , all transitions with a label from L_0 are accounted for by R since tss_1 is a granting extension of tss_0 which means that $tss_0 \models p \xrightarrow{l} p'$ iff $tss_1 \models p \xrightarrow{l} p'$. For the new transitions, $p \xrightarrow{l} p'$ for some $l \in L'$, we have that $p = p'$. Again, since the extension is granting we also have $q \xrightarrow{l} q$. Thus we have $q \Longrightarrow q \xrightarrow{l} q \Longrightarrow q$ and pRq .

Suppose that $tss_1 \models p \leftrightarrow_w q$. Then there exists a weak bisimulation R such that pRq . We claim that $R' = R \cap (C(\Sigma_0) \times C(\Sigma_0))$ is a weak bisimulation w.r.t. tss_0 . The crucial observation is that since the extension is granting, no transitions from a $C(\Sigma_0)$ term to $C(\Sigma_1)$ term are possible. Then, obviously every pair from R' satisfies the requirements of a weak bisimulation.

9. Branching bisimulation equivalence \leftrightarrow_b :

Definition 22 Let $tss = (\Sigma, L, D)$ be a TSS. A *branching bisimulation* w.r.t. tss is a symmetric binary relation R on processes, satisfying, for $l \in L$: if pRq and $tss \models p \xrightarrow{l} p'$, then either $l = \tau$ and $p'Rq$, or $q \Longrightarrow q_1 \xrightarrow{l} q_2 \Longrightarrow q'$ and $pRq_1, p'Rq_2$ and $p'Rq'$ for some q_1, q_2, q' . The processes p and q are branching bisimulation equivalent or branching bisimilar w.r.t. TSS tss , notation $tss \models p \leftrightarrow_b q$, iff there exist a branching bisimulation R such that pRq .

The proof is similar to the proof of the previous item.

10. Rooted weak bisimulation equivalence \leftrightarrow_{rw} :

Definition 23 Let $tss = (\Sigma, L, D)$ be a TSS. The processes p and q are rooted weak bisimulation equivalent or rooted weak bisimilar w.r.t. TSS tss , notation $tss \models p \leftrightarrow_{rw} q$, iff there exist a weak bisimulation R such that pRq and whenever $p \xrightarrow{l} p'$ there exists q' such that $q \xrightarrow{l} q'$ and $p'Rq'$ and whenever $q \xrightarrow{l} q'$ there exists p' such that $p \xrightarrow{l} p'$ and $p'Rq'$.

Apart from the proof given in item 8, it remains to show that for two terms $p, q \in C(\Sigma_0)$, the root condition holds w.r.t. tss_0 if and only if it holds for tss_1 . For all labels l from L_0 this follows immediately from the fact that tss_1 is a granting extension of tss_0 . For labels l' from L' , the only relevant (new) transitions in tss_1 are $p \xrightarrow{l'} p$ and $q \xrightarrow{l'} q$. Obviously, they pose no problem.

11. Rooted branching bisimulation equivalence \leftrightarrow_{rb} :

Definition 24 Let $tss = (\Sigma, L, D)$ be a TSS. The processes p and q are rooted branching bisimulation equivalent or rooted branching bisimilar w.r.t. TSS tss , notation $tss \models p \leftrightarrow_{rb} q$, iff there exist a branching bisimulation R such that pRq and whenever $p \xrightarrow{l} p'$ there exists q' such that $q \xrightarrow{l} q'$ and $p'Rq'$ and whenever $q \xrightarrow{l} q'$ there exists p' such that $p \xrightarrow{l} p'$ and $p'Rq'$.

The proof is similar to the proof of the previous item.

12. Bisimulation equivalence \Leftrightarrow :

Definition 25 Let $tss = (\Sigma, L, D)$ be a TSS. A *bisimulation* w.r.t. tss is a binary relation R on processes, satisfying, for $l \in L$: (1) if pRq and $tss \models p \xrightarrow{l} p'$, then $q \xrightarrow{l} q'$ and $p'Rq'$ for some q' , if pRq and $tss \models q \xrightarrow{l} q'$, then $p \xrightarrow{l} p'$ and $p'Rq'$ for some p' . The processes p and q are bisimulation equivalent w.r.t. TSS tss , notation $tss \models p \Leftrightarrow q$, iff there exist a bisimulation R such that pRq .

Suppose that $tss_0 \models p \Leftrightarrow q$. Then there exist a bisimulation R such that pRq . It is very easy to check that $R \cup Id$ is a bisimulation with respect to tss_1 and hence, $tss_1 \models p \Leftrightarrow q$.

For the inclusion in the other direction, suppose that, $tss_1 \models p \Leftrightarrow q$. Then, there exists a bisimulation R w.r.t. tss_1 such that pRq . One can easily verify that $R \cap (C(\Sigma_0) \times C(\Sigma_0))$ is a bisimulation w.r.t. tss_0 and hence $tss_0 \models p \Leftrightarrow q$.

□

Using the result of Theorem 2, henceforth, we refer to the concept of *orthogonality* and by that we mean \sim -orthogonality with respect to any of the notions of behavioral equivalence named above.

Unfortunately, not all notions of behavioral equivalence are preserved under granting extensions, i.e., granting extensions are not \sim -orthogonal for *all* notions of behavioral equivalence \sim . The only two counterexamples that we encountered so far in the literature are the notions of *completed-trace equivalence* and *complete simulation equivalence*. Next, we give a counterexample for completed-trace equivalence. The same example is also a counterexample for complete simulation equivalence.

Example 6 Consider the following deduction rule added to the semantics of *MPA* (tss_m of Example 1) which we refer to as MPA_ω .

$$\frac{}{a^\omega \xrightarrow{a} a^\omega}$$

Let $=_{CT}$ denote completed trace equivalence. It does not hold that $a^\omega + a.0 =_{CT} a^\omega$ since $a^\omega + a.0$ has a completed trace a while a^ω has no completed trace. (Note that the set of traces of these two process are equal, i.e., $a^\omega + a.0$ and a^ω are trace equivalent.)

Consider the granting extension of MPA_ω with the following deduction rule.

$$\frac{}{x \xrightarrow{1} x}$$

For MPA_ω extended with the above rule, the two processes become completed-trace equivalent, since both do not have any completed trace anymore.

5 Orthogonality Meta-Theorems

In this section, we seek sufficient conditions for establishing orthogonality and equational conservativity.

5.1 Granting Meta-Theorems

We start with defining sufficient conditions to prove an extension to be granting. Hence, we need to define when a deduction rule proves (only) self-transitions. We use unification as a means to this end.

Definition 26 (Unification) A term t is *unifiable* with t' using σ , denoted by $t \approx_\sigma t'$ if and only if $\sigma(t) = \sigma(t')$. The set of unifiers of t and t' is defined by $U(t, t') = \{\sigma \mid t \approx_\sigma t'\}$. The set of unifiers of a set of pairs is defined as the intersection of the sets of unifiers of each pair. The set of unifiers of an empty set is defined to include all substitutions. The set of unifiers of a positive formula $t \xrightarrow{l} t'$ is defined as the set of unifiers of t and t' . Unification also naturally extends to a set of positive formulae, again, using intersection.

Next, we characterize the set of rules that induce self-transitions. This is done by only allowing for unifiable (positive) formulae in the premises and the conclusion of a rule and further, by forcing the unification of the conclusion to follow from that of the premises.

Definition 27 (Source-Preserving Rules) A deduction rule $\frac{H}{c}$ without negative premises is *source preserving* if $U(H) \neq \emptyset$ and $U(H) \subseteq U(c)$. A TSS is *source preserving* if all its deduction rules are. For a source-preserving TSS, the set of *unified-conclusions* contains conclusions of the deduction rules with their unifiers applied to them.

The following lemma captures the intuition behind source-preserving rules.

Lemma 6 Suppose that $tss = (\Sigma, D, L)$ is source preserving, then $\forall l \in L \forall p, p' \in C(\Sigma) \ tss \vDash p \xrightarrow{l} p' \Rightarrow p = p'$.

Proof. Source-preserving rules do not contain negative premises and hence the two notations $tss \vDash p \xrightarrow{l} p'$ and $tss \vdash \frac{}{p \xrightarrow{l} p'}$ coincide. Hence, we proceed with an induction on the depth of the proof for $tss \vdash \frac{}{p \xrightarrow{l} p'}$.

If the proof tree has depth one, then the transition is due to a rule of the following form

$$\frac{}{t \xrightarrow{l} t'}$$

and a substitution σ such that $\sigma(t) = p$ and $\sigma(t') = p'$. But since tss is source preserving, it holds that $U(\emptyset) \subseteq U(t \xrightarrow{l} t')$ and since $\sigma \in U(\emptyset)$, it follows that $\sigma \in U(t \xrightarrow{l} t')$ and hence, $p = \sigma(t) = \sigma(t') = p'$.

For the induction step, suppose that the last deduction rule in the proof tree of depth n has the following form

$$\frac{H}{t \xrightarrow{l} t'}$$

and there exists a substitution σ such that $\sigma(t) = p$ and $\sigma(t') = p'$ and for all $h \in H$, $\sigma(h)$ is provable, trivially with a proof of depth less than n . Following the induction hypothesis, for all $h \in H$, $\sigma \in U(h)$ and thus, $\sigma \in U(H)$. It then follows from $U(H) \subseteq U(t \xrightarrow{l} t')$ that $\sigma \in U(t \xrightarrow{l} t')$ and we conclude that $p = \sigma(t) = \sigma(t') = p'$. \square

As illustrated above, source-preserving rules are safe for the purpose of proving self-transitions. However, there might be other rules in the extending TSS that can be harmful in that they may prove other types of transition for old terms. This may be prevented by forcing the other (non source-preserving) rules to have negative or non-unifiable positive premises addressing the old syntax. The following definition gives sufficient conditions for an extension to be granting.

Definition 28 (Granting Criteria) Consider a TSS $tss = (\Sigma, L, D)$ stratified by \mathcal{S} . It *grants* L_0 transitions on Σ_0 -terms, if $tss = tss_0 \cup tss_1$ (with $tss_x = (\Sigma_x, L_x, D_x)$ for $x \in \{0, 1\}$) such that:

1. tss_0 is strictly stratified by \mathcal{S} , it is source dependent and for all $l \in L_0$, the set containing sources of unified-conclusions of l -rules covers Σ_0 -terms, and

2. for all deduction rules $d \in D_1$ at least one of the following holds:

- (a) d has a function symbol from $\Sigma_1 \setminus \Sigma_0$ in the source of its conclusion, or
- (b) $\rho(d, \Sigma_0)$ has a source-dependent negative premise with a label in L_1 , or
- (c) $\rho(d, \Sigma_0)$ has a source-dependent positive premise $t \xrightarrow{l} t'$ with $l \in L_1$ and $U(t, t') = \emptyset$.

The first condition in the above definition is dedicated to proving self-transitions from the syntax of Σ_0 , and the the second one takes care of preventing Σ_0 -terms from performing other types of transitions while allowing other terms to do so.

Theorem 3 (Granting Meta-theorem) Consider TSS's $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$. If tss_0 is source dependent, tss_1 grants L_1 transitions on Σ_0 -terms and $L_0 \cap L_1 = \emptyset$ then $tss_0 \cup tss_1$ is a granting extension of tss_0 .

Proof. Since tss_1 grants L_1 transitions over Σ_0 -terms, there exists a decomposition of tss_1 into $tss_A \cup tss_B$ such that $tss_A = (\Sigma_0, L_1, D_A)$ and $tss_B = (\Sigma_B, L_B, D_B)$ and

- 1. tss_A is strictly stratified by \mathcal{S} , it is source preserving and for all $l \in L_1$, the set containing sources of unified-conclusions of l -rules covers Σ_0 -terms, and
- 2. for all deduction rules in $d \in D_B$ at least one of the following holds:
 - (a) d has a function symbol from $\Sigma_B \setminus \Sigma_0$ in the source of its conclusion, or
 - (b) $\rho(d, \Sigma_0)$ has a source-dependent negative premise with a label in L_1 , or
 - (c) $\rho(d, \Sigma_0)$ has a source-dependent positive premise $t \xrightarrow{l} t'$ with $l \in L_1$ and $U(t, t') = \emptyset$.

To show that $tss_0 \cup tss_1$ is a granting extension of tss_0 , we have to prove the following two items:

- 1. $\forall_{p, p' \in C(\Sigma_0)} \forall_{l \in L_0} tss_0 \vDash p \xrightarrow{l} p' \Leftrightarrow tss_0 \cup tss_1 \vDash p \xrightarrow{l} p'$. To show this, we prove that the sets of provable transition rules with conclusion $p \xrightarrow{l} p'$ w.r.t. tss_0 and w.r.t. $tss_0 \cup tss_1$ coincide.²

First, if a transition rule $\frac{N}{p \xrightarrow{l} p'}$ is provable w.r.t. tss_0 , then the same proof tree can be used w.r.t. $tss_0 \cup tss_1$ to provide us with $tss_0 \cup tss_1 \vdash \frac{N}{p \xrightarrow{l} p'}$.

Second, to prove that $tss_0 \cup tss_1 \vdash \frac{N}{p \xrightarrow{l} p'} \Rightarrow tss_0 \vdash \frac{N}{p \xrightarrow{l} p'}$, we prove the following stronger claim.

Claim. If for some $p \in C(\Sigma_0)$, $p' \in C(\Sigma_0 \cup \Sigma_1)$, $l \in L_0$, $tss_0 \cup tss_1 \vdash \frac{N}{p \xrightarrow{l} p'}$ then $p' \in C(\Sigma_0)$ and $tss_0 \vdash \frac{N}{p \xrightarrow{l} p'}$ (thus, N is built upon Σ_0 -terms and L_0 labels).

Proof. Take an arbitrary term $p \in C(\Sigma_0)$, we prove by an induction on the depth of the proof tree for $tss_0 \cup tss_1 \vdash \frac{N}{p \xrightarrow{l} p'}$ that $p' \in C(\Sigma_0)$ and $tss_0 \vdash \frac{N}{p \xrightarrow{l} p'}$. If the proof tree has depth one, then it is due to a deduction rule $d = (H, c)$ in tss_0 with no positive premises (rules in tss_1 have a disjoint set of labels and thus cannot provide a proof for a transition with label $l \in L_0$) and a substitution σ . Since d is source dependent and has no positive premise, variables in H are all among the variables in the source of c . Thus, applying σ to

²This is similar to the technique used in [8] for proving equality of the induced transition relations of two TSS's. Here, however, this technique is used to show the equality of partitions of the induced transition relation.

H and c yields terms from Σ_0 . Hence, using deduction rule d and substitution σ , we have a proof for $tss_0 \vdash \frac{N}{p \xrightarrow{l} p'}$ and this concludes the induction basis.

For the induction step, suppose that $tss_0 \cup tss_1 \vdash \frac{N}{p \xrightarrow{l} p'}$ has a proof of depth n . Then, the last deduction rule applied in the proof tree is in tss_0 (again, due to the disjointness of labels). Hence, the induction hypothesis applies to the positive premises (if any). Since tss_0 is source dependent, we can define a measure of source distance on premises as follows.

The source-dependency graph of a deduction rule is constructed by taking the variables that appear in the deduction rule as nodes and by putting an edge between two variables if one appears in the source and the other in the target of a premise. The distance of a variable in the source-dependency graph is the length of the shortest backward chain in this graph starting from the variable and ending in a variable in the source of the conclusion. The distance of a premise is the maximal distance of the variables of its source.

We proceed by an induction on the distance of premises in the source-dependency graph. For the induction basis, all the variables in the source of the premise should be among the variables in the source of the conclusion, hence the induction hypothesis on the depth of the proof applies and thus, the target of the premise (after substitution) should also be in $C(\Sigma_0)$. Similarly, for the induction step, all the variables in the source of the premise under consideration should already be valuated by terms in $C(\Sigma_0)$ and again by applying the induction hypothesis on the depth of the proof, we get that the variables in target are also valuated by terms in $C(\Sigma_0)$. Hence, the premises are all provable from the same set of negative premises in tss_0 and this concludes the proof of the lemma. \square

2. $\forall_{p \in C(\Sigma_0)} \forall_{p' \in C(\Sigma_0 \cup \Sigma_1)} \forall_{l \in L_1} tss_0 \cup tss_1 \vdash p \xrightarrow{l} p' \Leftrightarrow p = p'$. We first prove the implication from right to left, which will be used in the proof of the implication in the other direction.

Consider a formula $p \xrightarrow{l} p'$. Since tss_A is source preserving and for all $l \in L_1$, the set of sources of unified-conclusions of l -rules cover Σ_0 -terms. Hence, there should exist an l -rule in D_A of the following form:

$$\frac{H}{t \xrightarrow{l} t'}$$

and a unifier σ of t and t' and a substitution σ' such that $\sigma'(\sigma(t)) = p = \sigma'(\sigma(t'))$.

We proceed by an induction on the strict stratification measure $\mathcal{S}(p \xrightarrow{l} p)$. For the induction basis, consider $p \xrightarrow{l} p$ that has a minimal stratification measure. This means that deduction rule d has no premises, since otherwise, the premises would also be unifiable using $\sigma \circ \sigma'$ and have a strictly smaller measure. For a rule with an empty set of premises, any substitution is a unifier for the conclusion and hence using d and substitution $\sigma \circ \sigma'$, we have a proof for $p \xrightarrow{l} p$. For the induction step, since the unifier of the conclusion is a unifier for all the premises, by applying $\sigma \circ \sigma'$ on the premises and applying the induction hypothesis we construct a proof for $p \xrightarrow{l} p$ (see the proof of Lemma 6 for details).

For the implication in the other direction, namely $tss_0 \cup tss_1 \vdash p \xrightarrow{l} p' \Rightarrow p = p'$, we prove the following claim.

Claim. For all set of negative premises N such that $\forall_{l \in L_1} p \not\xrightarrow{l} \notin N, \forall_{l' \in L_1}$

$$tss_0 \cup tss_1 \vdash \frac{N}{p \xrightarrow{l'} p'} \Rightarrow p = p' \wedge N = \emptyset.$$

In the above claim, we require that $\forall l \in L_1 \ p \xrightarrow{l} \notin N$ since in the proof of the implication from right to left, we have shown that these negative formulae can always be contradicted by constructing a proof for $p \xrightarrow{l} p$.

Proof. To prove the claim, we use an induction on the proof depth for $tss_0 \cup tss_1 \vdash \frac{N}{p \xrightarrow{l} p'}$.

For the induction basis, since $l' \in L_1$, there exist a deduction rule $d \in D_1$ with no positive premises, of the following form

$$\frac{H}{t \xrightarrow{l'} t'}$$

and a substitution σ such that $\sigma(t) = p$ and $\sigma(t') = p'$. Suppose that d is in D_B then it either has a function symbol not in $\Sigma_B \setminus \Sigma_0$ in the source of its conclusion or $\rho(d, \Sigma_0)$ has a negative premise $t_j \xrightarrow{l_j}$ with $l_j \in L_1$. In the former case, the source of rule d cannot match p . In the latter case, $\text{vars}(t_j) \subseteq \text{vars}(t)$ (since d is source dependent and has no positive premises) and hence $\sigma(t_j) \in C(\Sigma_0)$, thus, there is a negative formulae $\sigma(t_j) \xrightarrow{l_j} \in N$, with $\sigma(t_j) \in C(\Sigma_0)$ and $l_j \in L_1$, contradicting our hypothesis. From these two contradictions, we conclude that $d \in D_A$. Following the same reasoning as in the other implication, from a rule in D_A with empty premises (since d is source preserving and has no positive premises), we can only prove transition rules of the form $\frac{}{p \xrightarrow{l'} p}$.

For the induction step, there should again be a deduction rule d in D_1 of the following form

$$\frac{H}{t \xrightarrow{l'} t'}$$

and a substitution σ such that $\sigma(t) = p$. If $d \in D_B$ one of the following three should hold:

- (a) $t \notin T(\Sigma_0)$, then $\sigma(t) \notin C(\Sigma_0)$ and this contradicts $\sigma(t) = p$;
- (b) $\rho(d, \Sigma_0)$ has a negative premise $t_j \xrightarrow{l_j}$ with $l_j \in L_1$. Then, using an induction on the chain of source dependencies leading to the premise $t_j \xrightarrow{l_j}$, it follows that $\sigma(t_j) \in C(\Sigma_0)$ and since $\sigma(t_j) \xrightarrow{l_j} \in N$, it contradicts our hypothesis.
- (c) $\rho(d, \Sigma_0)$ has a positive premise $t_i \xrightarrow{l_i} t'_i$ and $U(t_i \xrightarrow{l_i} t'_i) = \emptyset$. Again by an induction on the chain of source dependencies, it follows that $\sigma(t_i) \in C(\Sigma_0)$ and hence, following the induction hypothesis (concerning the proof depth), $\sigma(t_i) = \sigma(t'_i)$, and this contradicts $U(t_i \xrightarrow{l_i} t'_i) = \emptyset$.

Hence, we conclude that $d \in D_A$ and hence, it only has positive premises. We apply the induction hypothesis on all the premises in H and conclude that σ is a unifier for all $h \in H$ and hence it is a unifier for t and t' . Furthermore, it follows from the induction hypothesis that the set of leaves for the proof tree above each and every premise is empty and hence the whole proof tree has no negative premise as a leaf. Hence for this proof, it holds that $N = \emptyset$ and $\sigma(t) = \sigma(t') = p$. \square

This completes the proof. \square

5.2 Decomposing Orthogonality

An operationally conservative extension denies all types of new behavior from the old syntax. In total contrast, a granting extension forces to add all types of behavior to the old syntax. It would be most interesting, if we could reach a compromise between these two types of extensions while preserving the orthogonality. Hence, in this section, we propose a few ways of decomposing orthogonality into operationally conservative and granting extensions.

The first way to decompose orthogonality is by taking two subsets of the extending TSS with disjoint sets of labels and proving that one subset is a conservative extension and the other set is a granting extension of the extended TSS.

Theorem 4 Consider a TSS $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$. If $tss_1 = tss_{10} \cup tss_{11}$ (with $tss_x = (\Sigma_x, L_x, D_x)$ for $x \in \{10, 11\}$) such that:

1. tss_{10} satisfies the operational conservativity criteria of Theorem 1 w.r.t. tss_0 ,
2. tss_{11} satisfies the granting extension criteria of Theorem 3 w.r.t. tss_0 , and
3. $L_{10} \cap L_{11} = \emptyset$,

then $tss_0 \cup tss_1$ is an orthogonal extension of tss_0 .

Proof. Note that we cannot use Theorems 1 and 3 here directly by proving, for example, that $tss_0 \cup tss_1$ is a conservative extension of $tss_0 \cup tss_{11}$ since the hypotheses of Theorem 1 may be violated due to the addition of the new signature Σ_{11} .

Instead, we prove that $tss_0 \cup tss_1$ is a granting extension of $tss_A = (\Sigma_0, L_0 \cup L_{10}, D_0)$. Then $tss_1 \cup tss_0$ will be an orthogonal extension of tss_0 as well since the transition relations induced by tss_0 and tss_A coincide.

To prove that $tss_0 \cup tss_1$ is a granting extension of tss_A , copying Definition 12, we have to prove:

1. $\forall_{p,p' \in C(\Sigma_0)} \forall_{l \in L_0 \cup L_{10}} tss_A \vdash p \xrightarrow{l} p' \Leftrightarrow tss_0 \cup tss_1 \vdash p \xrightarrow{l} p'$, and
2. $\forall_{p \in C(\Sigma_0)} \forall_{p' \in C(\Sigma_0 \cup \Sigma_1)} \forall_{l \in L_{11}} tss_0 \cup tss_1 \vdash p \xrightarrow{l} p' \Leftrightarrow p = p'$.

To prove the first item, we prove the following stronger claim.

Claim. $\forall_{p \in C(\Sigma_0)} \forall_{p' \in C(\Sigma_0 \cup \Sigma_1)} \forall_{l \in L_0 \cup L_{10}} tss_A \vdash \frac{N}{p \xrightarrow{l} p'} \Leftrightarrow tss_0 \cup tss_1 \vDash \frac{N}{p \xrightarrow{l} p'}$

Proof. The implication from left to right holds trivially since any proof structure in tss_A remains valid in $tss_0 \cup tss_1$. For the implication in the other direction, all deduction rules used to deduce $tss_0 \cup tss_1 \vDash \frac{N}{p \xrightarrow{l} p'}$ should be in $D_0 \cup D_{10}$ since rules from D_{11} have disjoint labels from both L_0

and L_{10} . We prove the claim by an induction on the depth of the proof for $tss_0 \cup tss_1 \vDash \frac{N}{p \xrightarrow{l} p'}$.

If the last deduction rule d applied in the proof tree is in tss_0 and if we denote the substitution applied to d by σ , then it follows from source dependency and by an induction on the source distance of the premises that the source of all premises with σ applied to them are in $C(\Sigma_0)$ and since the labels are in L_0 , the induction hypothesis applies and all the positive premises have a proof in tss_A and all the variables in their target is evaluated by σ to terms in $C(\Sigma_0)$. The variables in the target of the conclusion are source dependent and hence, the target of the conclusion with σ applied to it is also in $C(\Sigma_0)$.

If the last deduction rule d applied in the proof tree is in D_{10} and if we denote the substitution applied to d by σ then it follows from the hypotheses of Theorem 1 that one of the following two conditions should hold:

1. source of the conclusion of d mentions a function symbol not in Σ_0 but then it cannot match p , or

2. $\rho(d, \Sigma_0)$ has a premise $t_i \xrightarrow{l_i} t'_i$ such that $l_i \notin L_0$ or $t'_i \notin T(\Sigma_0)$. In this case, by an induction on the source distance of $t_i \xrightarrow{l_i} t'_i$, it follows that $\sigma(t_i) \in C(\Sigma_0)$ and since $d \in D_{10}$, $l \in L_{10}$, thus, the induction hypothesis applies and $tss_A \vdash \frac{N}{\sigma(t_i) \xrightarrow{l_i} \sigma(t'_i)}$, but since $\sigma(t'_i) \notin T(\Sigma_0)$ this transition is not provable in tss_A .

Hence, both cases lead to a contradiction and a rule in D_{10} cannot be used in constructing a proof for $\frac{N}{p \xrightarrow{l} p'}$ and this concludes the proof of our claim. \square

For the second item, we have to prove that $\forall_{p \in C(\Sigma_0)} \forall_{p' \in C(\Sigma_0 \cup \Sigma_1)} \forall_{l \in L_{11}} tss_0 \cup tss_1 \vdash p \xrightarrow{l} p' \Leftrightarrow p = p'$. To prove this item, we can confine ourselves to rules in D_{11} since rules in $D_0 \cup D_{10}$ have disjoint labels and hence cannot provide a proof for $p \xrightarrow{l} p'$. tss_{11} satisfies the hypotheses of Theorem 3 and hence it grants L_{11} transitions on Σ_0 terms. Using a similar proof as of Theorem 3, we can prove that for Σ_0 -terms as a source, it can only prove self transitions with L_{11} labels, hence, the second item. \square

Note that, in general, the proof obligation for orthogonality cannot be decomposed into the proof of orthogonality of two subsets of an extension. However, we conjecture that if the two subsets have disjoint labels and one has a disjoint label with the original TSS, the orthogonality of the combination can be guaranteed (thus, a generalization of the above theorem).

Another way to decompose orthogonality is to apply conservativity and granting extension theorems in sequence. This is possible by virtue of the following corollary which follows trivially from the definition of orthogonality (Definition 11).

Corollary 12 Orthogonal extension is a preorder.

Proof. For an arbitrary tss , it trivially holds that tss is an orthogonal extension of itself. Consider three TSS's tss_0 , tss_1 and tss_2 with $tss_x \doteq (\Sigma_x, L_x, D_x)$ for $x \in \{0, 1, 2\}$ such that tss_2 is an orthogonal extension of tss_1 and tss_1 is an orthogonal extension of tss_0 . One can easily check that the two conditions for tss_2 to be an orthogonal extension of tss_0 hold:

1. $\forall_{p, p' \in C(\Sigma_0)} \forall_{l \in L_0} tss_2 \vDash p \xrightarrow{l} p' \Leftrightarrow tss_1 \vDash p \xrightarrow{l} p' \Leftrightarrow tss_0 \vDash p \xrightarrow{l} p'$, and
2. $\forall_{p, p' \in C(\Sigma_0)} tss_2 \vDash p \sim p' \Leftrightarrow tss_1 \vDash p \sim p' \Leftrightarrow tss_0 \vDash p \sim p'$.

\square

Using this corollary one can interleave several steps of operationally conservative and granting extensions to define different new aspects and, in the end, have an orthogonal extension of the original language. The following example illustrates applications of the above results.

Example 7 (Timed-MPA: Orthogonality) Consider the tss_m of MPA in Example 1 and tss_t of Example 3. TSS tss_t can be decomposed into the following three parts: $tss_0 \doteq (\{\underline{a}, \delta\}, A, \{(\mathbf{ua}), (\mathbf{td})\})$, $tss_1 \doteq (\{\delta, a, -, +, -\}, \{1\}, \{(\mathbf{tc0}), (\mathbf{ta}), (\mathbf{d})\})$ and $tss_2 \doteq (\{-, +, -\}, \{1\}, \{(\mathbf{tc1}), (\mathbf{tc2})\})$.

It follows from Definition 27 that tss_1 is source preserving since:

1. the conclusions of (\mathbf{ta}) and (\mathbf{d}) are unifiable using any substitution, hence using the unifiers of the empty set of premises, and
2. the conclusion of $(\mathbf{tc0})$ is unifiable using the unifiers of the premises, i.e., those that evaluate x and x' to the same term and y and y' to the same term.

It then follows from Definition 28 that $tss_1 \cup tss_2$ grants time transitions over MPA terms since

1. tss_1 is strictly stratified using a simple measure of size on terms, it is source preserving as shown before, and by applying unifiers to the source of conclusion of **(tc0)**, **(ta)** and **(d)**, i.e., the set $\{x + y, a.x, \delta\}$, we can cover the syntax of *MPA*,
2. in tss_2 , deduction rules **(tc1)** and **(tc2)** have source-dependent negative premises with label 1 (note that **(tc1)** and **(tc2)** are the same as their reduced versions).

From Theorem 3, it follows that the extension of tss_m with $tss_1 \cup tss_2$ is a granting extension, hence an orthogonal extension. Furthermore, the extension of $tss_m \cup tss_1 \cup tss_2$ with tss_0 is conservative, hence orthogonal, following Theorem 1. Since orthogonality is a preorder, we conclude that $tss_m \cup tss_t$ is an orthogonal extension of tss_m .

Alternatively, we could use Theorem 4 to obtain orthogonality by using the above results. Namely, tss_0 satisfies the criteria for operational conservativity of Theorem 1 w.r.t. tss_m , $tss_1 \cup tss_2$ satisfies the granting criteria of Theorem 3 w.r.t. tss_m and finally, the labels of tss_0 and $tss_1 \cup tss_2$ are disjoint. Hence, using Theorem 4, we can conclude that $tss_m \cup tss_t$ is an orthogonal extension of tss_m .

To give an idea how to deal with predicates in our settings, we treat the process algebra timed-*MPA* enriched with a termination predicate and successful termination constants.

Example 8 (Timed-*MPA*: Termination)

$$\begin{array}{l}
(1) \quad \text{(et)} \frac{}{\epsilon \downarrow} \quad \text{(ct0)} \frac{x_0 \downarrow}{x_0 + x_1 \downarrow} \quad \text{(ct0)} \frac{x_1 \downarrow}{x_0 + x_1 \downarrow} \\
(2) \quad \quad \quad \text{(te)} \frac{}{\epsilon \xrightarrow{1} \epsilon} \quad \text{(ue)} \frac{}{\underline{\epsilon} \downarrow}
\end{array}$$

Consider the above TSS which is supposed to be added to the tss_t of Example 3. In order to deal with the termination predicate in our setting, we transform it to the form of a binary transition formula with a new label \downarrow . However, a choice can be made concerning the target of the newly formed formulae. In [22], it is suggested to take different fresh variables as targets of transformed premises and a new dummy constant as targets of premises. Thus, the above example would be transformed to the following TSS, where \surd is the dummy constant. Using the following TSS, we can now apply Theorem 1 and conclude that timed-*MPA* with successful termination is an orthogonal extension of both timed-*MPA* and *MPA*.

$$\begin{array}{l}
(1) \quad \text{(et)} \frac{}{\epsilon \xrightarrow{\downarrow} \surd} \quad \text{(ct0)} \frac{x_0 \xrightarrow{\downarrow} y_0}{x_0 + x_1 \xrightarrow{\downarrow} \surd} \quad \text{(ct0)} \frac{x_1 \xrightarrow{\downarrow} y_1}{x_0 + x_1 \xrightarrow{\downarrow} \surd} \\
(2) \quad \quad \quad \text{(te)} \frac{}{\epsilon \xrightarrow{1} \epsilon} \quad \text{(ue)} \frac{}{\underline{\epsilon} \xrightarrow{\downarrow} \surd}
\end{array}$$

The above choice of targets for the target of transformed formulae is motivated by the desire to make the transformed TSS conform to a certain rule format, i.e., PANTH format of [22]. However, for proving orthogonality, we do not necessarily need the conformance to the PANTH format and one can take other targets for the targets. For example, a natural choice would be to take the source terms as the target and this way, we end up with the following TSS.

$$\begin{array}{l}
(1) \quad \text{(et)} \frac{}{\epsilon \xrightarrow{\downarrow} \epsilon} \quad \text{(ct0)} \frac{x_0 \xrightarrow{\downarrow} x_0}{x_0 + x_1 \xrightarrow{\downarrow} x_0 + x_1} \quad \text{(ct0)} \frac{x_1 \xrightarrow{\downarrow} x_1}{x_0 + x_1 \xrightarrow{\downarrow} x_0 + x_1} \\
(2) \quad \quad \quad \text{(te)} \frac{}{\epsilon \xrightarrow{1} \epsilon} \quad \text{(ue)} \frac{}{\underline{\epsilon} \xrightarrow{\downarrow} \underline{\epsilon}}
\end{array}$$

Actually, the above transformation may be preferable in the case of proving a granting extension since it discharges all obligation concerning unification.

5.3 Orthogonality and Equational Conservativity

The following theorems establish the link between orthogonality and equational conservativity. They are very similar to those in [22, 23] about the relation between operational and equational conservativity. The first theorem states that a sound axiomatization of an operationally conservative extension cannot induce new equalities on the old syntax.

Theorem 5 (Equational Conservativity Theorem) Consider TSS's $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$ where tss_1 is an orthogonal extension of tss_0 . Also let E_0 be a sound and complete axiomatization of tss_0 and E_1 be a sound axiomatization of tss_1 . If $\forall_{p,p' \in C(\Sigma_0)} E_0 \vdash p = p' \Rightarrow E_1 \vdash p = p'$ then E_1 is an equationally conservative ground-extension of E_0 .

Proof. We have to prove that $\forall_{p,p' \in C(\Sigma_0)} E_0 \vdash p = p' \Leftrightarrow E_1 \vdash p = p'$. The implication from left to right is given by the hypotheses, thus it remains to prove that $E_1 \vdash p = p' \Rightarrow E_0 \vdash p = p'$. Since E_1 is sound, it follows that $tss_1 \vdash p \sim p'$ and since tss_1 is an orthogonal extension of tss_0 , it should hold that $tss_0 \vdash p \sim p'$. Then it follows from the completeness of E_0 that $E_0 \vdash p = p'$. \square

The next theorem enables us to use orthogonality as a means to equational conservativity.

Theorem 6 Consider TSS's $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$ where tss_1 is an orthogonal extension of tss_0 . Also let E_0 and E_1 be sound and complete axiomatizations of tss_0 and tss_1 , respectively. Then, E_1 is an equationally conservative ground-extension of E_0 .

Proof. We have to prove that $\forall_{p,p' \in C(\Sigma_0)} E_0 \vdash p = p' \Leftrightarrow E_1 \vdash p = p'$. From $E_0 \vdash p = p'$, by soundness of E_0 , $tss_0 \vdash p \sim p'$. Since tss_1 is an orthogonal extension of tss_0 , we obtain $tss_1 \vdash p \sim p'$. From the hypothesis that E_1 is a complete axiomatization of \sim w.r.t. tss_1 , we have $E_1 \vdash p = p'$. Similarly, from $E_1 \vdash p = p'$, since E_1 is sound, we have $tss_1 \vdash p \sim p'$. Due to orthogonality, $tss_0 \vdash p \sim p'$. The completeness of E_0 then gives $E_0 \vdash p = p'$. \square

Using the above theorem, we can obtain the equational conservativity result for timed-*MPA*.

Example 9 (Timed-*MPA*: Equational Conservativity) Consider the equational theories for *MPA* and timed-*MPA* presented in Example 2 and 5. Assuming the soundness of completeness of both axiomatizations which we claimed without a proof, and the orthogonality of the extension of *MPA* to timed-*MPA* which we proved in Example 7, we can deduce using Theorem 6 that the equational theory of Example 5 is an equationally conservative ground-extension of that of Example 2.

Finally, the last theorem establishes sufficient conditions for a sound equationally conservative ground-extension to be a complete equational theory for the extended language.

Theorem 7 (Elimination Theorem) Consider TSS's $tss_0 = (\Sigma_0, L_0, D_0)$ and $tss_1 = (\Sigma_1, L_1, D_1)$ where tss_1 is an orthogonal extension of tss_0 . Also let E_0 and E_1 be sound axiomatizations of tss_0 and tss_1 , respectively. If E_0 is also complete on tss_0 , E_1 is an equationally conservative ground-extension of E_0 and E_1 eliminates terms from $\Sigma_1 \setminus \Sigma_0$, then E_1 is complete for tss_1 .

Proof. Consider two closed terms $p, p' \in C(\Sigma_1)$ such that $tss_1 \vdash p \sim p'$. Since E_1 eliminates terms from $\Sigma_1 \setminus \Sigma_0$, there exist two terms $q, q' \in C(\Sigma_0)$ such that $E_1 \vdash p = q$ and $E_1 \vdash p' = q'$. It follows from soundness of E_1 that $tss_1 \vdash p \sim q$ and $tss_1 \vdash p' \sim q'$ and since \sim is an equivalence relation, it follows that $tss_1 \vdash q \sim q'$. But tss_1 is an orthogonal extension of tss_0 and hence, we have $tss_0 \vdash q \sim q'$. E_0 is a complete axiomatization of tss_0 and thus, $E_0 \vdash q = q'$ and it follows from the equational conservativity of E_1 with respect to E_0 that $E_1 \vdash q = q'$. From $p = q$, $q = q'$ and $p' = q'$, we conclude that $E_1 \vdash p = p'$. \square

A typical line of reasoning starts with taking an orthogonal extension and a sound axiomatization thereof, and proving equational conservativity using Theorem 5. Then, by proving an elimination result for newly introduced operators, one can get completeness of the axiomatization following Theorem 7.

6 Conclusions

In this paper, we defined a more relaxed notion of operational conservativity, called *orthogonality* which allows for non-destructive extension of the behavior of the old language. We gave a meta-theorem providing sufficient conditions (which are indeed more relaxed than the sufficient conditions for the traditional notion). Also, we presented the slightly more general notion of *equational conservativity for ground-extensions* and established the link between these two notions. The concepts and results were illustrated by extending the Minimal Process Algebra of [2] to timed setting.

Extending the theory presented in this paper with the concept of variable binding is a straightforward extension along the lines of [8]. The second enhancement of our work concerns operational extensions that require a translation of labels (using a kind of abstraction function). Finally, investigating the possibility of other realizations of orthogonality is an interesting subject for future research.

References

- [1] L. Aceto, W. J. Fokkink, and C. Verhoef. Structural operational semantics. In *Handbook of Process Algebra*, Chapter 3, pages 197–292. Elsevier Science, 2001.
- [2] J. C. M. Baeten. Embedding untimed into timed process algebra: the case for explicit termination. *Mathematical Structures in Computer Science*, 13(4):589–618, 2003.
- [3] J. C. M. Baeten and J. A. Bergstra. Discrete time process algebra. *Formal Aspects of Computing*, 8(2):188–208, 1996.
- [4] J. C. M. Baeten and C. A. Middelburg. *Process Algebra with Timing*, Springer-Verlag, 2002.
- [5] J. C. M. Baeten and C. Verhoef. Concrete Process Algebra. In *Handbook of Logic in Computer Science*, volume 4, *Semantic Modelling*, pages 149–268. Oxford University Press, 1995.
- [6] J. C. M. Baeten and W. P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.
- [7] R. Bol and J. F. Groote. The meaning of negative premises in transition system specifications. *Journal of the ACM*, 43(5):863–914, Sept. 1996.
- [8] W. J. Fokkink and C. Verhoef. A conservative look at operational semantics with variable binding. *Information and Computation*, 146(1):24–54, 1998.
- [9] R. J. van Glabbeek. The linear time - branching time spectrum II. In E. Best, editor, *International Conference on Concurrency Theory (CONCUR'93)*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer-Verlag, Berlin, Germany, 1993.
- [10] R. J. van Glabbeek. The meaning of negative premises in transition system specifications II. *Journal of Logic and Algebraic Programming*, 60-61:229–258, 2004.
- [11] R. J. van Glabbeek. The linear time - branching time spectrum I. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra, Chapter 1*, pages 3–100. Elsevier Science, Dordrecht, The Netherlands, 2001.

- [12] J. F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118(2):263–299, 1993.
- [13] J. F. Groote and F. W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.
- [14] G. Leduc and L. Leonard. A timed LOTOS supporting a dense time domain and including new timed operators. In *Proceedings of FORTE'92*, volume C-10 of *IFIP Transactions*, pages 87–102. North-Holland, 1993.
- [15] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
- [16] C.A. Middelburg. An alternative formulation of operational conservativity with binding terms. *Journal of Logic and Algebraic Programming*, 55(1/2):1–19, 2003.
- [17] G. D. Plotkin. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming*, 60:17–139, 2004.
- [18] M. A. Reniers, J. F. Groote, M. B. van der Zwaag, and J. van Wamel. Completeness of timed μCRL . *Fundamenta Informaticae*, 50(3-4):361–402, 2002.
- [19] A. Rensink. Bisimilarity of open terms. *Information and Computation*, 156:345–385, 2000.
- [20] J. J. Vereijken. *Discrete Time Process Algebra*. PhD thesis, Department of Mathematics and Computer Science, Eindhoven University of Technology, 1997.
- [21] C. Verhoef. A general conservative extension theorem in process algebra. In E.-R. Olderog, editor, *Proceedings of PROCOMET'94*, volume A-56 of *IFIP Transactions*, pages 274–302. Elsevier Science Publishers, 1994.
- [22] C. Verhoef. A congruence theorem for structured operational semantics with predicates and negative premises. *Nordic Journal of Computing*, 2(2):274–302, 1995.
- [23] C. Verhoef, L. Aceto, and W. Fokkink. Conservative extension in structural operational semantics. *Bulletin of the EATCS*, 69:110–132, 1999.