Syntactic Formats for Free

An Abstract Approach to Process Equivalence

Bartek Klin   Paweł Sobociński

BRICS[⋆]
University of Aarhus, Denmark

Abstract. A framework of Plotkin and Turi's, originally aimed at providing an abstract notion of bisimulation, is modified to cover other operational equivalences and preorders. Combined with bialgebraic methods, it yields a technique for the derivation of syntactic formats for transition system specifications which guarantee operational preorders to be precongruences. The technique is applied to the trace preorder, the completed trace preorder and the failures preorder. In the latter two cases, new syntactic formats ensuring precongruence properties are introduced.

1 Introduction

Structural operational semantics [18, 2] is one of the most fundamental frameworks for providing a precise interpretation of programming and specification languages. It is usually presented as a labelled transition system (LTS), in which states (sometimes called processes) are closed terms over some syntactic signature, and transitions are labelled with elements of some fixed set of actions. The transition relation is in turn specified by a set of derivation rules.

Many operational equivalences and preorders have been defined on processes. Among these are: bisimulation equivalence [17], simulation preorder, trace preorder, completed trace preorder, failures preorder [13, 21] and many others (for a comprehensive list see [10]). In the case of processes without internal actions, all of the above have been given modal characterisations [10].

Reasoning about operational equivalences and preorders is significantly easier when they are congruences (resp. precongruences). This facilitates compositional reasoning and full substitutivity. In general, operational equivalences (preorders) are not necessarily congruences (resp. precongruences) on processes defined by operational rules. Proofs of such congruence results for given transition system specifications can be quite demanding.

One way to ensure congruential properties is to impose syntactic restrictions (syntactic formats) on operational rules. Many such formats have been developed. For bisimulation equivalence, the examples are: de Simone format [23],

⋆ Basic Research in Computer Science (`www.brics.dk`), funded by the Danish National Research Foundation.

GSOS [8], and ntyft/ntyxt [11]. For trace equivalence, examples include [27, 5], while decorated trace preorders have been provided with formats in [6]. For an overview of the subject see [2].

The search for an abstract theory of bisimulation and 'well-behaved' operational semantics has led to development of final coalgebra semantics [22], and bialgebraic semantics [25, 26] of processes. In these frameworks, the notion of a transition system is parametrised by a notion of behaviour. Bisimulation is modelled abstractly as a span of coalgebra morphisms.

In [25, 26] it was shown how to define operational rules on an abstract level, guaranteeing bisimulation equivalence (defined abstractly, using spans of coalgebra morphisms) to be a congruence. At the core of this so-called abstract GSOS is the modelling of a transition system specification as a natural transformation

$$\lambda : \Sigma(\mathrm{id} \times B) \to BT$$

where Σ is the syntactic endofunctor, T is the monad freely generated from Σ, and B is some behaviour endofunctor. In the special case of the behaviour endofunctor $\mathcal{P}_{\mathrm{f}}(A \times -)$, the abstract operational rules specialise to GSOS rules.

The abstract framework which defines bisimulation as a span of coalgebra morphisms is not sufficient for certain purposes [19]. Recently, another abstract notion of bisimulation, based on topologies (or complete boolean algebras) of tests, has been proposed [20, 24].

In this paper we show that the latter abstract definition of bisimulation can be modified in a structured manner, to yield other known equivalences and preorders. We illustrate this approach on trace, completed trace and failures preorders. This constitutes a systematic approach to operational preorders, such as those based on testing scenarios [10], modal logics [10], and quantales [1].

Although the framework is general, in this paper we shall concentrate on the category of sets and functions, **Set**. We define the *test-suite fibration* with total category $\mathbf{Set}^*$ having as objects pairs consisting of a set X and a test suite (a subset of $\mathcal{P}X$) over X. We lift the abstract-GSOS framework to $\mathbf{Set}^*$ by describing how to lift the syntactic functor Σ and the behaviour functor B. By changing how B lifts to $\mathbf{Set}^*$ we alter the specialisation preorder of certain test suites in $\mathbf{Set}^*$. In particular, taking liftings which strongly resemble fragments of the Hennessy-Milner logic [12] causes the specialisation preorder to vary between known operational preorders. The abstract framework guarantees precongruence properties. The only hurdle is proving that a particular transition system specification (natural transformation) λ lifts to a natural transformation in $\mathbf{Set}^*$:

$$\lambda : \Sigma^*(\mathrm{id} \times B^*) \to B^*T^*.$$

The consideration of which properties λ must satisfy in order to lift provides us with syntactic sub-formats of GSOS which guarantee precongruence properties for various operational preorders.

In this paper, we illustrate this approach by presenting precongruence formats for the trace preorder, the completed trace preorder and the failures preorder.

The format derived for the trace preorder coincides with the well known de Simone format [27]. The format derived for the completed trace preorder is, to the best of our knowledge, the first such format published. The format derived for the failures preorder is incomparable with the analogous format given in [6].

The structure of the paper is as follows. After §2 of preliminaries, we present the three obtained syntactic formats in §3, together with some examples and counterexamples from literature. The remaining sections are devoted to proving that the presented formats are indeed precongruence formats w.r.t. their respective preorders, and at the same time to illustrating the method of deriving such formats from a given operational preorder. In §4, we recall the basics of bialgebraic semantics. In §5, we present an abstract approach to operational preorders based on the notion of a test suite. In §6, this approach is merged with the bialgebraic framework to yield a general way of checking whether a given operational preorder is a congruence for a given transition system specification. Finally, in §7, we show that the formats presented in §3 ensure the respective precongruence results. We conclude in §8 by showing possible directions of future work. Due to lack of space, most proofs are left to the full version of this paper [15].

Acknowledgements. Most of the contents of §5 and §6 is a modified version of the framework developed (and, unfortunately, not yet published) by Gordon Plotkin [20] and Daniele Turi [24]. Thanks also goes to Mikkel Nygaard for reading the paper and providing us with many valuable comments. The first author is also grateful to Daniele Turi for introducing him to the subject and for inspiration, and to Gordon Plotkin for discussions and encouragement.

2 Preliminaries

A *labelled transition system* (LTS) is a set P of *processes*, a set A of *actions*, and a *transition relation* $\longrightarrow \subseteq P \times A \times P$. As usual, we write $p \xrightarrow{a} p'$ instead of $\langle p, a, p' \rangle \in \longrightarrow$. An LTS is *finitely branching*, if for every process p there are only finitely many transitions $p \xrightarrow{a} p'$.

Given a set of actions A, three sets of *modal formulae* $\mathcal{F}_{\mathsf{Tr}}$, $\mathcal{F}_{\mathsf{CTr}}$, and $\mathcal{F}_{\mathsf{Fl}}$ are given by the following BNF grammars:

$$\begin{array}{ccc} \mathcal{F}_{\mathsf{Tr}} & \mathcal{F}_{\mathsf{CTr}} & \mathcal{F}_{\mathsf{Fl}} \\ \phi ::= \top \mid \langle a \rangle\, \phi & \phi ::= \top \mid \langle a \rangle\, \phi \mid \tilde{A} & \phi ::= \top \mid \langle a \rangle\, \phi \mid \tilde{Q} \end{array}$$

where a ranges over A, and Q ranges over subsets of A. Formulae in $\mathcal{F}_{\mathsf{Tr}}$ are called *traces* over A. Formulae in $\mathcal{F}_{\mathsf{CTr}}$ ended with $\tilde{A}$ are called *completed traces*, and formulae in $\mathcal{F}_{\mathsf{Fl}}$ — *failures*.

Given an LTS, the satisfaction relation $\models$ between processes and modal formulae is defined inductively as follows:

$$\begin{array}{lcl} p \models \top & & \\ p \models \langle a \rangle\, \phi & \iff & p' \models \phi \text{ for some } p' \text{ such that } p \xrightarrow{a} p' \\ p \models \tilde{Q} & \iff & \text{there is no } a \in Q,\ p' \in P \text{ such that } p \xrightarrow{a} p' \end{array}$$

Then three operational preorders on the set of processes are defined: the *trace preorder* $\sqsubseteq_{\mathsf{Tr}}$, the *completed trace preorder* $\sqsubseteq_{\mathsf{CTr}}$, and the *failures preorder* $\sqsubseteq_{\mathsf{Fl}}$:

$$p \sqsubseteq_W p' \iff (\forall \phi \in \mathcal{F}_W . p \models \phi \Longrightarrow p' \models \phi)$$

where $W \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}\}$.

In the context of structural operational semantics, processes are usually closed terms over some signature. A *signature* Σ is a set (also denoted Σ) of *language constructs*, together with an *arity function* $ar : \Sigma \to \mathbb{N}$. For a given set X of *variables*, ΣX is the set of expressions of the form $\mathtt{f}(x_1, \ldots, x_{ar(\mathtt{f})})$, where $\mathtt{f} \in \Sigma$ and $x_i \in X$. Given a signature Σ and a set X, the set $T_\Sigma X$ of *terms* over Σ with variables X is (isomorphic to) the least fixpoint of the operator

$$\Phi Y = X + \Sigma Y$$

where $+$ denotes disjoint union of sets. When describing terms from $T_\Sigma X$ the injections $\iota_1 : X \to T_\Sigma X$ and $\iota_2 : \Sigma T_\Sigma X \to T_\Sigma X$ will often be omitted, i.e., we will write $\mathtt{f}(x, y)$ rather than $\iota_2(\mathtt{f}(\iota_1(x), \iota_1(y)))$. Also the subscript in $T_\Sigma X$ will be omitted if Σ is irrelevant or clear from the context. Elements of $T\emptyset$ are called *closed terms* over Σ.

For a term $t \in TX$ and a function $\sigma : X \to Y$, $t[\sigma]$ will denote the term in TY resulting from t by simultaneously replacing every $x \in X$ with $\sigma(x)$.

In the following, we assume a fixed, infinite set of variables Ξ, ranged over by $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{y}_1, \mathtt{y}_2, \ldots$. Terms with variables from Ξ will be typeset $\mathtt{t}$, $\mathtt{t}'$, etc.

Let us fix an arbitrary set of labels A. For a signature Σ, a *positive Σ-literal* is an expression $\mathtt{t} \xrightarrow{a} \mathtt{t}'$, and a *negative Σ-literal* is an expression $\mathtt{t} \overset{a}{\not\longrightarrow}$, where $\mathtt{t}, \mathtt{t}' \in T\Xi$ and $a \in A$. A *transition rule* ρ over Σ is an expression $\frac{H}{\alpha}$, where H is a set of Σ-literals and α is a positive Σ-literal. Elements of H are called *premises* of ρ, and α — the *conclusion* of ρ. The left-hand side and the right-hand side of the conclusion of ρ are called the *source* and the *target* of ρ, respectively. A *transition system specification* over Σ is a set of transition rules over Σ.

Similarly, a *Σ-semiliteral* is either a negative Σ-literal, or an expression $\mathtt{t} \xrightarrow{a}$, where $\mathtt{t} \in T\Xi$ and $a \in A$. A positive literal $\mathtt{t} \xrightarrow{a} \mathtt{t}'$ *completes* the semiliteral $\mathtt{t} \xrightarrow{a}$, and we say that a negative literal completes itself.

In the following definition assume a fixed signature Σ, and a *finite* set A.

Format 1 (GSOS) A transition system specification R is in GSOS [8] format if every rule $\rho \in R$ is of the form

$$\frac{\left\{ \mathtt{x}_i \xrightarrow{a_{ij}} \mathtt{y}_{ij} \;:\; i \leq n,\; j \leq m_i \right\} \cup \left\{ \mathtt{x}_i \overset{b_{ik}}{\not\longrightarrow} \;:\; i \leq n, k \leq n_i \right\}}{\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{c} \mathtt{t}}$$

with $\mathtt{f} \in \Sigma$ and $n = ar(\mathtt{f})$, such that $\mathtt{x}_i \in \Xi$ and $\mathtt{y}_{ij} \in \Xi$ are all distinct and are the only variables that occur in ρ. In the following, we will consider only *image finite* GSOS specifications, i.e. those with finitely many rules for each construct $\mathtt{f} \in \Sigma$ and action $c \in A$.

Given a transition system specification R in GSOS format, one defines a notion of a provable positive literal in a straightforward way. The set of all provable literals forms a finitely branching LTS with closed terms over Σ as processes, and with positive closed literals as transitions (for details, see [2]).

An operational preorder $\sqsubseteq$ is a *precongruence* with respect to a transition system specification R, if in the LTS induced by R, for each $\mathtt{f} \in \Sigma$ with arity n, if $t_1 \sqsubseteq t'_1, \ldots, t_n \sqsubseteq t'_n$, then $\mathtt{f}(t_1, \ldots, t_n) \sqsubseteq \mathtt{f}(t'_1, \ldots, t'_n)$.

The examples in §3 are based on basic process algebra **BPA**. Assuming a finite set A of actions, its syntax Σ is defined by the BNF grammar $t ::= \mathtt{0} \mid \alpha t \mid t{+}t$ and the transition system specification **BPA** over Σ is a collection of rules

$$\frac{}{\alpha\mathtt{x} \xrightarrow{\alpha} \mathtt{x}} \qquad \frac{\mathtt{x} \xrightarrow{\alpha} \mathtt{x}'}{\mathtt{x} + \mathtt{y} \xrightarrow{\alpha} \mathtt{x}'} \qquad \frac{\mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}{\mathtt{x} + \mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}$$

where α ranges over A. When presenting terms over the above syntax, the trailing $\mathtt{0}$'s will be omitted. It is easy to see that **BPA** is in the GSOS format.

3 Precongruence Formats

In this section we introduce the syntactic formats derived using the framework described in the latter parts of the paper. The precongruence properties of these formats are formally stated in §7.

Format 2 (Tr-format) A set of GSOS rules R is in Tr-*format*, if for each $\rho \in R$, all premises of ρ are positive, and no variable occurs more than once in the left-hand sides of premises and in the target of ρ.

It is easy to see that this format coincides with the well-known de Simone format [23]. The fact that this syntactic format ensures the trace preorder to be a precongruence was first proved in [27].

We proceed to define an analogous syntactic format for the completed trace preorder.

Definition 1 (CTr-testing set). A CTr-*testing set* P over a set of variables $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$ is a set of semiliterals of the form

$$P = \left\{ \mathtt{x}_i \xrightarrow{a_i}\!\!\!\!\!\!/\;\; : i \in I \right\} \cup \left\{ \mathtt{x}_i \xrightarrow{a} \; : i \in J, a \in A \right\}$$

where $I, J \subseteq \{1, \ldots, n\}$.

Format 3 (CTr-format) A set of GSOS rules R is in CTr-*format*, if

1. For each rule $\rho \in R$:
   - if ρ has a negative premise $\mathtt{x} \xrightarrow{a}\!\!\!\!\!\!/\;\;$, than for every label $b \in A$, ρ has also the negative premise $\mathtt{x} \xrightarrow{b}\!\!\!\!\!\!/\;\;$,

  - no variable occurs more than once in the target of ρ,
  - no variable occurs simultaneously in the left-hand side of a premise and in the target of ρ,
  - no variable occurs simultaneously in the left-hand side of a positive premise and in the left-hand side of any other premise of ρ.
2. For each construct $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$ of the language, there exists a sequence $P_1, \ldots, P_k$ of CTr-testing sets over $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$, such that
  - For every (possibly renamed) rule $\rho \in R$ with source $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$ there exists a sequence $p_1, \ldots p_k$ of semiliterals from $P_1, \ldots, P_k$ respectively, such that for every $i \in \{1, \ldots, k\}$ there exists a premise r of ρ such that r completes p_i.
  - For every sequence $p_1, \ldots, p_k$ of semiliterals from $P_1, \ldots, P_k$ respectively, there exists a (possibly renamed) rule $\rho \in R$ with source $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$ such that for each premise r of ρ there exists an $i \in \{1, \ldots, k\}$ such that r completes p_i.

Proposition 2. **BPA** is in CTr-format. □

The following example is taken from [2]. Assume $A = \{a, b\}$, and extend **BPA** with an operational rule for the so-called *encapsulation operator* $\partial_{\{b\}}$:

$$\frac{\mathtt{x} \xrightarrow{a} \mathtt{y}}{\partial_{\{b\}}(\mathtt{x}) \xrightarrow{a} \partial_{\{b\}}(\mathtt{y})}$$

Then it is easy to check that $aa + ab \sim_{\mathsf{CTr}} a(a + b)$ but that $\partial_{\{b\}}(aa + ab) \not\sqsubseteq_{\mathsf{CTr}} \partial_{\{b\}}(a(a + b))$.

Another example of an operational construct that is not well behaved with respect to completed traces is the *synchronous composition*, as shown in [27]. Here, we add the rules

$$\frac{\mathtt{x} \xrightarrow{\alpha} \mathtt{x}' \quad \mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}{\mathtt{x} \times \mathtt{y} \xrightarrow{\alpha} \mathtt{x}' \times \mathtt{y}'}$$

where α ranges over $A = \{a, b\}$. Here it is easy to see that $aa \times (aa + ab) \not\sqsubseteq_{\mathsf{CTr}} aa \times a(a + b)$.

These two examples have led the authors of [2] to speculate that one cannot hope for a general syntactic congruence format for completed trace equivalence.

Proposition 3. The semantics for the encapsulation operator ∂ and the synchronous composition $\times$ are not in CTr-format. □

For a non-trivial example of a transition system specification in CTr-format, extend **BPA** with *sequential composition*, defined by rules

$$\frac{\mathtt{x} \xrightarrow{\alpha} \mathtt{x}'}{\mathtt{x};\mathtt{y} \xrightarrow{\alpha} \mathtt{x}';\mathtt{y}} \qquad \frac{\mathtt{x} \stackrel{a}{\not\rightarrow} \text{ for all } a \in A \quad \mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}{\mathtt{x};\mathtt{y} \xrightarrow{\alpha} \mathtt{y}'}$$

where α ranges over A.

Proposition 4. **BPA** extended with sequential composition is in CTr-format. $\square$

We proceed to define a precongruence syntactic format for the failures preorder.

Definition 5 (Fl-testing set). An Fl-*testing set* P over a set of variables $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$ is a set of semiliterals of the form

$$P = \left\{ \mathtt{x}_i \overset{a_i}{\not\longrightarrow} \ : \ i \in I \right\} \cup \left\{ \mathtt{x}_i \xrightarrow{b_{ij}} \ : \ 1 \leq i \leq n, 1 \leq j \leq m_i \right\}$$

(where $I \subseteq \{1, \ldots, n\}$, $m_i \in \mathbb{N}$), such that for any labels $a, b \in A$, if $\mathtt{x}_i \xrightarrow{a} \in P$ and $\mathtt{x}_i \overset{b}{\not\longrightarrow} \in P$ then $\mathtt{x}_i \xrightarrow{b} \in P$.

Format 4 (Failures Format) A set of GSOS rules R is in Fl-*format*, if

1. For each rule $\rho \in R$:
   - no variable occurs more than once in the target of ρ,
   - no variable occurs simultaneously in the left-hand side of a premise and in the target of ρ,
   - no variable occurs simultaneously in the left-hand side of a positive premise and in the left-hand side of any other premise of ρ.
2. For each construct $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)$ of the language, and for each set of labels $Q \subseteq A$, there exists a sequence $P_1, \ldots, P_k$ of Fl-testing sets over $\{\mathtt{x}_1, \ldots, \mathtt{x}_n\}$, such that
   - For every (possibly renamed) rule $\rho \in R$ with the conclusion of the form $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{a} \mathtt{t}$ with $a \in Q$ and an arbitrary $\mathtt{t}$, there exists a sequence $p_1, \ldots, p_k$ of semiliterals from $P_1, \ldots, P_k$ respectively, such that for every $i \in \{1, \ldots, k\}$ there exists a premise r of ρ such that r completes p_i.
   - For every sequence $p_1, \ldots, p_k$ of semiliterals from $P_1, \ldots, P_k$ respectively, there exist a label $a \in Q$, a term $\mathtt{t}$, and a (possibly renamed) rule $\rho \in R$ with the conclusion $\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n) \xrightarrow{a} \mathtt{t}$ such that for each premise r of ρ there exists an $i \in \{1, \ldots, k\}$ such that r completes p_i.

Proposition 6. **BPA** is in Fl-format. $\square$

In [7] it was shown that the failures preorder is not a precongruence for **BPA** extended with sequential composition.

Proposition 7. If A contains at least two different labels a, b, then **BPA** extended with sequential composition is not in Fl-format. $\square$

The Fl-format excludes many examples of transition system specifications that behave well with respect to the failures preorder. Many of these examples are covered by the 'failure trace format' introduced in [6]. However, the latter format excludes also some examples covered by FI-format. Indeed, assume $a, b \in A$ and extend **BPA** with two unary constructs $\mathtt{g}, \mathtt{h}$ and rules (where α ranges over A)

$$\frac{\mathtt{x} \xrightarrow{\alpha} \mathtt{x}'}{\mathtt{g(x)} \xrightarrow{\alpha} \mathtt{h(x')}} \qquad \frac{\mathtt{x} \overset{a}{\not\rightarrow}}{\mathtt{h(x)} \xrightarrow{b} \mathtt{0}}$$

Proposition 8. **BPA** extended with $\mathtt{g}$ and $\mathtt{h}$ as above, is in FI-format. □

However, the rules above are not in the 'failure trace format' proposed in [6]. This means that FI-format is incomparable with that format.

4 An Abstract Approach

In this section we shall recall the foundations needed for the framework described in §5 and §6. First, we briefly recall how LTS can be described as coalgebras for a specific behaviour endofunctor and briefly recall final coalgebra semantics. We then proceed to recall several notions from the abstract approach to operational semantics of Plotkin and Turi [26].

In the following, $\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$ will denote the (covariant) powerset functor. The (covariant) finite powerset functor $\mathcal{P}_\mathrm{f} : \mathbf{Set} \to \mathbf{Set}$ takes a set to the set of its *finite* subsets. The reader is referred to [16] for any unexplained categorical notation used henceforward.

There is a bijection between the set of finitely branching LTS over a fixed set of actions A and the coalgebras of the functor $\mathcal{P}_\mathrm{f}(A \times -)$. Indeed, given an LTS $\langle P, A, \longrightarrow \rangle$ let

$$h : P \to \mathcal{P}_\mathrm{f}(A \times P)$$

be defined by $h(p) = \{ \langle a, p' \rangle \, : \, p \xrightarrow{a} p' \}$.

The functor $\mathcal{P}_\mathrm{f}(A \times -)$ has a final coalgebra $\varphi : S \to \mathcal{P}_\mathrm{f}(A \times S)$. The carrier S of this coalgebra may be described as the set of synchronisation trees with edges having labels from A, quotiented by bisimulation [4, 25].

In the following we specialise the framework of [26] to the category **Set** and behaviour functor $\mathcal{P}_\mathrm{f}(A \times -)$. Any syntactic signature Σ determines a so-called syntactic endofunctor $\Sigma : \mathbf{Set} \to \mathbf{Set}$ which acts on sets by sending

$$\Sigma X = \coprod_{\mathtt{f} \in \Sigma} X^{ar(\mathtt{f})} \tag{1}$$

and the action on functions is the obvious one. The functor Σ freely generates a monad $\langle T, \mu, \eta \rangle : \mathbf{Set} \to \mathbf{Set}$. It turns out that TX is (isomorphic to) the set of all terms over Σ with variables from X.

Theorem 9 ([26]). There is a correspondence between sets of rules in the GSOS format (Format 1) and natural transformations

$$\lambda : \Sigma(\mathrm{id} \times \mathcal{P}_\mathrm{f}(A \times -)) \to \mathcal{P}_\mathrm{f}(A \times T-)$$

Moreover, the correspondence is 1-1 up to equivalence of sets of rules. □

Assume a natural transformation $\lambda : \Sigma(\mathrm{id} \times B) \to BT$. A λ-*model* is a pair

$$\Sigma X \xrightarrow{h} X \xrightarrow{g} BX$$

such that $g \circ h = Bh^\sharp \circ \lambda_X \circ \Sigma \langle \mathrm{id}, g \rangle$, ($h^\sharp : TX \to X$ is the inductive extension of h). A λ-model morphism between $\Sigma X \xrightarrow{h} X \xrightarrow{g} BX$ and $\Sigma X' \xrightarrow{h'} X' \xrightarrow{g'} BX'$ is a morphism $f : X \to X'$ which is simultaneously a Σ-algebra morphism and a B-coalgebra morphism, ie. $h' \circ \Sigma f = g \circ h$ and $g' \circ f = Bf \circ g$. Let λ-**Mod** denote the category of λ-models and λ-model morphisms.

Theorem 10 ([26]). Suppose that **C** is a category, Σ is an endofunctor which freely generates a monad T and B is an endofunctor which cofreely generates a comonad D. Then the following hold:

1. λ-**Mod** has an initial and final object,
2. the carrier and algebra part of the initial λ-model is the initial Σ-algebra,
3. the carrier and coalgebra part of the final λ-model is the final B-coalgebra,
4. the coalgebra part of the initial λ-model is the so-called *intended operational model* of λ. □

In particular, if $\mathbf{C} = \mathbf{Set}$ and $B = \mathcal{P}_\mathrm{f}(A \times -)$, then the intended operational model of λ is the LTS generated by the GSOS rules associated to λ.

5 Process Equivalences from Fibred Functors

In this section, we introduce the central concept of a test suite fibration. This is a modification of the yet unpublished framework [20, 24] due to Plotkin and Turi. In that approach, the test suites (Definition 11) are necessarily *topologies*, that is, they satisfy certain closure properties. We relax this definition and require only that a test suite contains the largest test. This modification allows us to consider operational preorders and equivalences different from bisimulation. Also, the original framework was developed largely for **Cppo**-enriched categories, here we deal primarily with **Set**.

We define $2 = \{\mathtt{tt}, \mathtt{ff}\}$. Given a function $f : X \to Y$ and subsets $V \subseteq X$, $V' \subseteq Y$, we shall use $f(V)$ to denote the set $\{\, y \in Y \;:\; \exists x \in V.\, fx = y \,\}$ and similarly $f^{-1}(V')$ to denote $\{\, x \in X \;:\; fx \in V' \,\}$. Given a set $\tau \subseteq \mathcal{P}X$, the *specialisation preorder* of τ is defined by

$$x \leq_\tau x' \quad \text{iff} \quad \forall V \in \tau.\, x \in V \Rightarrow x' \in V$$

For an introduction to fibrations and related terminology, the reader is referred to the first chapter of [14].

Definition 11 (Test suite). A *test* on a set X is a function $V : X \to 2$. We say that an element x *passes a test* V iff $Vx = \mathtt{tt}$. A *test suite* on X is a collection of tests on X which includes the maximal test, that is, the function constant at $\mathtt{tt}$. Let X^* denote the poset of test suites on X ordered by inclusion.

We can define a functor $(-)^* : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Pos}$ which sends a set to the poset of test suites X^* and sends a function $f : X \to Y$ to $f^* : Y^* \to X^*$ defined by

$$f^*\tau' = \{\, V' \circ f \;:\; V' \in \tau' \,\} .$$

Intuitively, we think of tests on X as subsets of X. Then f^* is the pre-image operation, taking each test on Y to the test on X which maps to Y via f.

Definition 12 (Test suite fibration). A *fibration of test suites* for $(-)^*$ is the fibration obtained using the Grothendieck construction, ie. the total category $\mathbf{Set}^*$ has

- objects: pairs $\langle X, \tau \rangle$ where $X \in \mathbf{Set}$ and $\tau \in X^*$, τ is a test suite.
- arrows: $\langle X, \tau \rangle \xrightarrow{f} \langle X', \tau' \rangle$ iff $f : X \to X'$ and $f^*\tau' \subseteq \tau$.

It is then standard that the obvious forgetful functor $U : \mathbf{Set}^* \to \mathbf{Set}$ taking $\langle X, \tau \rangle$ to X is a fibration.

It will be useful to define various operations on test suites τ. Letting $\nabla : 2 + 2 \to 2$ be the codiagonal and $\wedge : 2 \times 2 \to 2$ be logical-and, we let

$$\begin{aligned}
\tau \oplus \tau' &= \{\, \nabla \circ (V + V') \;:\; V \in \tau, V' \in \tau' \,\} \\
\tau \otimes \tau' &= \{\, \wedge \circ (V \times V') \;:\; V \in \tau, V' \in \tau' \,\} \\
\tau \bowtie \tau' &= \{\, V \circ \pi_1 \;:\; V \in \tau \,\} \cup \{\, V' \circ \pi_2 \;:\; V' \in \tau' \,\} .
\end{aligned}$$

It is easy to check that given two test suites, families $\tau \oplus \tau'$, $\tau \otimes \tau'$ and $\tau \bowtie \tau'$ are test suites. Intuitively, given test suites τ and τ' on X and Y, $\tau \oplus \tau'$ is the test suite on $X + Y$ obtained by taking (disjoint) unions of tests from τ on X and τ' on Y, $\tau \otimes \tau'$ is the test suite on $X \times Y$ consisting of tests built by performing a test from τ on X *and simultaneously* performing a test from τ' on Y and accepting when both tests accept; finally, $\tau \bowtie \tau'$ is the test on $X \times Y$ which consists of *either* a test from τ on X *or* a test from τ' on Y.

Proposition 13. The category $\mathbf{Set}^*$ has coproducts and products:

$$\begin{aligned}
\langle X, \tau \rangle + \langle Y, \tau' \rangle &= \langle X + Y, \tau \oplus \tau' \rangle \\
\langle X, \tau \rangle \times \langle Y, \tau' \rangle &= \langle X \times Y, \tau \bowtie \tau' \rangle
\end{aligned}$$

Let $B : \mathbf{Set} \to \mathbf{Set}$ be some behaviour endofunctor. A *lifting* of B to $\mathbf{Set}^*$ is an endofunctor $B^* : \mathbf{Set}^* \to \mathbf{Set}^*$ such that, for some $B_X : X^* \to (BX)^*$ we have $B^* \langle X, \tau \rangle = \langle BX, B_X \tau \rangle$ and $B^* f = Bf$. It turns out that there are many possible choices for B_X giving different liftings of B to $\mathbf{Set}^*$. One systematic way to construct such liftings is via families of functions from $B2$ to 2. Intuitively, such functions correspond to modalities like those in the Hennessy-Milner logic. In the original framework due to Plotkin and Turi [20, 24] the canonical choice of *all* functions from $B2$ to 2 is taken.

For any X, let $\mathrm{Cl}_X : \mathcal{P}\mathcal{P}X \to X^*$ denote a closure operator. We shall only demand that for all $f : X \to Y$ and $Z \subseteq \mathcal{P}Y$ we have $\mathrm{Cl}_X\, f^*Z = f^*\, \mathrm{Cl}_Y\, Z$ (with the obvious extension of the domain of f^* from Y^* to $\mathcal{P}\mathcal{P}Y$). Intuitively, a closure operator corresponds to a set of propositional connectives.

Given an arbitrary family W of functions $B2 \to 2$, we define an operator $B^W_X : X^* \to (BX)^*$ as follows:

$$B^W_X(\tau) = \mathrm{Cl}_{BX}\, \{\, w \circ BV \;:\; w \in W \text{ and } V \in \tau \,\}\,.$$

We are now in a position to construct a lifting of B to $\mathbf{Set}^*$. Indeed, we let $B^W \langle X, \tau\rangle = \left\langle BX, B^W_X \tau \right\rangle$ and $B^W f = Bf$. It turns out that this defines an endofunctor B^W on $\mathbf{Set}^*$.

Theorem 14. Suppose that $B : \mathbf{Set} \to \mathbf{Set}$ has a final coalgebra $\varphi : S \to BS$. Then $\varphi : \left\langle S, M^W \right\rangle \to \left\langle BS, B^W_S M^W \right\rangle$ is a final B^W coalgebra where M^W is the least fixpoint of the operator $\Phi(\tau) = \varphi^* B^W_S \tau$ on S^*. $\square$

Suppose that $B : \mathbf{Set} \to \mathbf{Set}$ lifts to a functor $B^W : \mathbf{Set}^* \to \mathbf{Set}^*$ with B^W defined as before.

Theorem 15. Take any coalgebra $h : X \to BX$, and let $k : X \to S$ be the unique coalgebra morphism from h to the final B-coalgebra. Then k^*M (where $\langle S, M\rangle$ is the carrier of the final B^W-coalgebra) is the least test suite τ on X such that $h : \langle X, \tau\rangle \to \left\langle BX, B^W_X \tau \right\rangle$ is a morphism in $\mathbf{Set}^*$. $\square$

From now on we shall assume a finite set of labels A and confine our attention to the endofunctor $BX = \mathcal{P}_{\mathrm{f}}(A \times X)$ on $\mathbf{Set}$.

Assuming $a \in A$ and $Q \subseteq A$, let $w_{\langle a\rangle}, w_{rQ} : B2 \to 2$ denote the functions

$$w_{\langle a\rangle} X = \begin{cases} \mathtt{tt} & \text{if } \langle a, \mathtt{tt}\rangle \in X \\ \mathtt{ff} & \text{otherwise,} \end{cases} \qquad w_{rQ} X = \begin{cases} \mathtt{tt} & \text{if } \forall a \in Q\, \forall v \in 2.\, \langle a, v\rangle \notin X \\ \mathtt{ff} & \text{otherwise.} \end{cases}$$

We shall now define three subsets of maps $B2 \to 2$:

$$\mathsf{Tr} = \left\{\, w_{\langle a\rangle} \;:\; a \in A \,\right\} \qquad \mathsf{CTr} = \mathsf{Tr} \cup \{w_{rA}\} \qquad \mathsf{Fl} = \mathsf{Tr} \cup \{\, w_{rQ} \;:\; Q \subseteq A \,\}$$

The set Tr together with the closure operator $\mathrm{Cl}^{\top}_X(\tau) = \tau \cup \{X\}$, determines B^{Tr}_X for any X and therefore determines a lifting of B to $B^{\mathsf{Tr}} : \mathbf{Set}^* \to \mathbf{Set}^*$. Similarly, CTr with $\mathrm{Cl}^{\top}$ and and Fl with $\mathrm{Cl}^{\top}$ determine liftings B^{CTr} and B^{Fl} respectively.

The following Theorems 16-18 show that the specialisation preorders in the final B^{Tr}, B^{CTr} and B^{Fl}-coalgebras coincide with the trace, the completed trace and the failures preorders. We use these facts to prove Theorem 19 which states that given any $h : X \to \mathcal{P}_{\mathrm{f}}(A \times X)$, the specialisation preorders on certain test suites on X coincide with these operational preorders.

Theorems 16-18. In the final B^W-coalgebra, the specialization preorder coincides with $\sqsubseteq_W$, where $W \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}\}$.

Theorem 19. Suppose that $h : X \to \mathcal{P}_{\mathrm{f}}(A \times X)$ is a coalgebra (LTS), $\varphi : S \to BS$ is the final B-coalgebra and that $k : X \to S$ is the unique morphism given by finality. Then $x \leq_{k^* M^W} x'$ if and only if $x \sqsubseteq_W x'$, where $W \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}\}$ and $\langle S, M^W \rangle$ is the carrier of the final B^W-coalgebra. □

6 Application: Congruence Formats from Bialgebras

To lift the bialgebraic framework to the total category **Set***, we need a way to lift the syntactic and the behaviour functors together with the natural transformation λ. Various ways to lift the behaviour B were shown in the previous section, now we proceed to show a lifting of the syntactic functor.

Given a syntactic endofunctor Σ on **Set** defined as in Equation (1), define an endofunctor Σ^* on **Set***: $\Sigma^* \langle X, \tau \rangle = \langle \Sigma X, \Sigma_X \tau \rangle$, where

$$\Sigma_X \tau = \mathrm{Cl}^{\cup} \Big(\bigoplus_{\mathtt{f} \in \Sigma} \tau^{\otimes ar(\mathtt{f})} \Big)$$

where $\mathrm{Cl}^{\cup}$ is closure under arbitrary unions, and $\tau^{\otimes n}$ denotes $\overbrace{\tau \otimes \tau \otimes \cdots \otimes \tau}^{n \text{ times}}$. On arrows, given $f : \langle X, \tau \rangle \to \langle X', \tau' \rangle$, we define simply $\Sigma^* f = \Sigma f$. It turns out that Σ^* defined this way is an endofunctor on **Set***.

Theorem 20. Suppose that an endofunctor F lifts to a endofunctor F^*, and has an initial algebra $\psi : FN \to N$. Then $\psi : \langle FN, F_N P \rangle \to \langle N, P \rangle$ is the initial F^* algebra where P is the greatest fixpoint of the operator $\Psi(\tau) = (\psi^{-1})^* F_N \tau$.

Corollary 21. For any syntactic endofunctor Σ, the functor Σ^* freely generates a monad T^* that lifts the monad T freely generated by Σ. □

A similar corollary about a behaviour B^W cofreely generating a comonad D^W can be drawn from Theorem 14. These two corollaries allow us to apply Theorem 10 for the category **Set*** and endofunctors Σ^* and B^W.

The following theorem is a crucial property of the endofunctor Σ^*. Indeed, varying the definition of Σ^* in our framework would lead to definition of various precongruence formats, but only as long as the following property holds.

Theorem 22. For any Σ^*-algebra $h : \langle \Sigma X, \Sigma_X \tau \rangle \to \langle X, \tau \rangle$, the specialisation preorder $\leq_\tau$ is a precongruence on $h : \Sigma X \to X$. □

We now have the technology needed to prove the main result of this section.

Consider a natural transformation $\lambda : \Sigma(\mathrm{id} \times B) \to BT$. By Theorem 10, the coalgebraic part of the initial λ-model has $N = T\emptyset$ as its carrier, and it is the intended operational model of λ. If $B = \mathcal{P}_{\mathrm{f}}(A \times -)$, then the intended operational model is the LTS generated by GSOS rules associated to λ. Let $k : N \to S$ be the final coalgebra morphism from the intended operational model to the final B-coalgebra. Assume B lifts to some B^* as before, and let $\langle S, M\rangle$ be the carrier of the final B^*-coalgebra.

Theorem 23. If λ lifts to a natural transformation in the total category:

$$\lambda : \Sigma^*(\mathrm{id} \times B^*) \to B^*T^*.$$

then the specialisation preorder on k^*M is a precongruence on N.

Proof. (Sketch) In diagram *(i)* below, the left column is the initial λ-model while the right column is the final λ-model; the λ-model morphism k is the unique morphism making both squares commutative.

$$\begin{tikzcd}
\Sigma^* \langle N, P\rangle \arrow[r, "\Sigma^* k"] \arrow[d, "\psi"'] & \Sigma^* \langle S, M\rangle \arrow[d, "\delta"] \\
\langle N, P\rangle \arrow[r, "k"] \arrow[d, "\epsilon"'] & \langle S, M\rangle \arrow[d, "\varphi"] \\
B^* \langle N, P\rangle \arrow[r, "B^* k"'] & B^* \langle S, M\rangle
\end{tikzcd}
\qquad
\begin{tikzcd}
\Sigma^* \langle N, k^*M\rangle \arrow[r, "\Sigma^* k"] \arrow[d, "\psi"'] & \Sigma^* \langle S, M\rangle \arrow[d, "\delta"] \\
\langle N, k^*M\rangle \arrow[r, "k"] \arrow[d, "\epsilon"'] & \langle S, M\rangle \arrow[d, "\varphi"] \\
B^* \langle N, k^*M\rangle \arrow[r, "B^* k"'] & B^* \langle S, M\rangle
\end{tikzcd}$$

(i) *(ii)*

Our goal is to show that *(ii)* above a diagram in $\mathbf{Set}^*$. If all the morphisms are defined then its commutativity follows from the commutativity of *(i)*. By Theorem 15, $\epsilon : \langle N, k^*M\rangle \to B^* \langle N, k^*M\rangle$ is a B^*-coalgebra.

Now $\psi^*k^*M = (\Sigma k)^*\delta^*M \subseteq (\Sigma k)^*(\Sigma_S M) \subseteq \Sigma_X(k^*M)$ where we use the fact that δ is a morphism in $\mathbf{Set}^*$ and the fact that Σ^* is a functor. Thus $\psi : \langle N, k^*M\rangle \to B^* \langle N, k^*M\rangle$ is a Σ^*-algebra and by Theorem 22 the specialisation preorder of k^*M is a precongruence. □

7 Precongruence Formats for (almost) Free

In this section we consider a syntactic endofunctor Σ with a freely generated monad T, the behaviour functor $BX = \mathcal{P}_{\mathrm{f}}(A \times X)$, and a set R of GSOS rules with the corresponding natural transformation $\lambda : \Sigma(\mathrm{id} \times B) \to BT$. The purpose is to describe syntactic conditions on R that would ensure that λ lifts to a natural transformation $\lambda : \Sigma^*(\mathrm{id} \times B^W) \to B^W T^*$, where $W \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}\}$. As a consequence of Theorem 23, such syntactic conditions ensure that the respective operational preorders are precongruences.

Theorems 24-26. For $W \in \{\mathsf{Tr}, \mathsf{CTr}, \mathsf{Fl}\}$, if R is in W-format (see Form. 2-4), then $\lambda : \Sigma^*(Id \times B^W) \to B^W T^*$ is a natural transformation in $\mathbf{Set}^*$. $\square$

8 Conclusions

We have presented an abstract coalgebraic approach to the description of various operational preorders, via a fibration of test suites. In Theorems 16-18 we illustrated this approach on the trace preorder, the completed trace preorder and the failures preorder. Combined with bialgebraic methods, this framework allows the derivation of syntactic subformats of GSOS which guarantee that the above operational preorders are precongruences. Theorem 23 is a guideline in the search for such formats, and Theorems 24-26 are applications of the framework.

The generality and abstractness of Theorem 23 prompted us to coin the expression 'precongruence format for free'. However, it must be stressed that to derive a format for a given operational preorder remains a non-trivial task. Indeed, the proofs of Theorems 24-26 are quite long and technical. The expression 'for free' reflects the fact that Theorem 23 lets us prove precongruence properties without considering the global behaviour (e.g. traces) of processes. Instead, one considers only simple test constructions, corresponding intuitively to single modalities.

Related abstract approaches to operational preorders and equivalences include those based on modal characterisations [10] and quantales [1]. In the latter framework, no syntactic issues have been addressed. In the former, some general precongruence formats have been obtained by attempting to decompose modal formulae according to given operational rules [7]. This technique bears some resemblance to our approach, and the precise connections have to be investigated.

There are several possible directions of future work. Firstly, the approach presented here can be extended to deal with other operational preorders and equivalences described in literature. Secondly, one can move from the GSOS format (and its subformats) to the more general (safe) ntree format [9], which can also be formalised in the bialgebraic framework [26]. Thirdly, the abstract framework of test suites seems to be general enough to cover other notions of process behaviour (e.g. involving store), or even other underlying categories (e.g. complete partial orders instead of sets). It may prove interesting to formalise various operational preorders in such cases and to find precongruence formats for them.

References

1. S. Abramsky and S. Vickers. Quantales, observational logic and process semantics. *Math. Struct. in Comp. Sci.*, 3:161–227, 1993.
2. L. Aceto, W. Fokkink, and C. Verhoef. Structural operational semantics. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra.* Elsevier, 1999.

3. P. Aczel and N. Mendler. A final coalgebra theorem. In D. H. Pitt et al., editors, *Proc. CTCS'89*, volume 389 of *LNCS*, pages 357–365, 1989.
4. M. Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 114:299–315, 1993.
5. B. Bloom. When is partial trace equivalence adequate? *Formal Aspects of Computing*, 6:25–68, 1994.
6. B. Bloom, W. Fokkink, and R. J. van Glabbeek. Precongruence formats for decorated trace preorders. In *Logic in Computer Science*, pages 107–118, 2000.
7. B. Bloom, W.J Fokkink, and R.J van Glabbeek. Precongruence formats for decorated trace semantics. *ACM Transactions on Computational Logic*. To appear.
8. B. Bloom, S. Istrail, and A. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42:232–268, 1995.
9. W. Fokkink and R. van Glabbeek. Ntyft/ntyxt rules reduce to ntree rules. *Information and Computation*, 126:1–10, 1996.
10. R. J. van Glabbeek. The linear time-branching time spectrum I. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 1999.
11. J. F. Groote. Transition system specifications with negative premises. *Theoret. Comput. Sci.*, 118:263–299, 1993.
12. M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32:137–161, 1985.
13. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
14. B. Jacobs. *Categorical Logic and Type Theory*, volume 141 of *Studies in Logic and the Foundations of Mathematics*. North Holland, Elsevier, 1999.
15. B. Klin and P. Sobociński. Syntactic formats for free: An abstract approach to process equivalence. BRICS Report RS-03-18, Aarhus University, 2003. Available from `http://www.brics.dk/RS/03/18/BRICS-RS-03-18.pdf`.
16. S. Mac Lane. *Categories for the Working Matematician*. Springer-Verlag, 1998.
17. D. M. Park. Concurrency on automata and infinite sequences. In P. Deussen, editor, *Conf. on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*. Springer Verlag, 1981.
18. G. Plotkin. A structural approach to operational semantics. DAIMI Report FN-19, Computer Science Department, Aarhus University, 1981.
19. G. Plotkin. Bialgebraic semantics and recursion (extended abstract). In A. Corradini, M. Lenisa, and U. Montanari, editors, *Electronic Notes in Theoretical Computer Science*, volume 44. Elsevier Science Publishers, 2001.
20. G. Plotkin. Bialgebraic semantics and recursion. Invited talk, Workshop on Coalgebraic Methods in Computer Science, Genova, 2001.
21. A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, 1997.
22. J. Rutten and D. Turi. Initial algebra and final coalgebra semantics for concurrency. In J. de Bakker et al., editors, *Proc. of the REX workshop A Decade of Concurrency – Reflections and Perspectives*, *LNCS* vol. 803, pp. 530–582. Springer-Verlag, 1994.
23. R. de Simone. Higher-level synchronising devices in Meije-SCCS. *Theoret. Comput. Sci.*, 37:245–267, 1985.
24. D. Turi. Fibrations and bisimulation. Unpublished notes.
25. D. Turi. *Functorial Operational Semantics and its Denotational Dual*. PhD thesis, Vrije Universiteit, Amsterdam, 1996.
26. D. Turi and G. Plotkin. Towards a mathematical operational semantics. In *Proceedings 12th Ann. IEEE Symp. on Logic in Computer Science, LICS'97, Warsaw, Poland, 29 June – 2 July 1997*, pages 280–291. IEEE Computer Society Press, 1997.
27. Frits W. Vaandrager. On the relationship between process algebra and input/output automata. In *Logic in Computer Science*, pages 387–398, 1991.