An Intensionally Fully-abstract Sheaf Model for pi

Clovis Eberhart, Tom Hirschowitz, Thomas Seiller

An Intensionally Fully-abstract Sheaf Model for π^*

Clovis Eberhart[1], Tom Hirschowitz[2], and Thomas Seiller[3]

1 **Université Savoie Mont-Blanc, France**
2 **CNRS, Université Savoie Mont-Blanc, France**
3 **Université Paris 7, France**

Abstract

Following previous work on CCS, we propose a compositional model for the pi-calculus in which processes are interpreted as sheaves on certain simple sites. We define an analogue of fair testing equivalence in the model and show that our interpretation is intensionally fully abstract for it. That is, the interpretation preserves and reflects fair testing equivalence; and furthermore, any strategy is fair testing equivalent to the interpretation of some process. The central part of our work is the construction of our sites, whose heart is a combinatorial presentation of pi-calculus traces in the spirit of string diagrams. As in previous work, the sheaf condition is analogous to innocence in Hyland-Ong/Nickau games.

1998 ACM Subject Classification F.3.2 Semantics of Programming Languages

Keywords and phrases concurrency, sheaves, causal models, games

Digital Object Identifier 10.4230/LIPIcs.CALCO.2015.86

1 Introduction

Operational semantics of programming languages standardly model the execution of programs as paths in a certain labelled transition system (LTS). Under this interpretation, different possible interleavings of parallel actions yield different paths. Verification on LTSs thus incurs a well-known state explosion problem. Similarly, causality between various actions, visible in the syntax, is lost in the LTS, thus making, e.g., error diagnostics difficult [17].

Causal models, originally designed for Petri nets [37] and Milner's CCS [42], intend to remedy both problems, but have yet to be applied to full-scale programming languages. They have recently been extended in two different directions: (1) by Crafa et al. [10] to Milner's π-calculus, and (2) by Melliès [32] to Girard's linear logic. The former extension accounts for the subtle interaction of channel creation with synchronisation in π, a significant technical achievement, 30 years after the first causal semantics for CCS. The latter is the first causal model for functional languages (inspired by Hyland-Ong's and Nickau's *games* models for PCF [36, 24]). An important challenge is now the search for a causal model of full-fledged languages with both concurrent and functional features. Winskel and collaborators are currently working in this direction, using extensions of Melliès's approach [39, 43, 8].

In previous work [23, 21, 22], we have proposed a causal model for CCS based on a different approach. We here push this approach further by applying it to the π-calculus.

* The authors acknowledge support from French ANR projets blancs PiCoq ANR 2010 BLAN 0305 01 and Récré ANR-11-BS02-0010.

6th International Conference on Algebra and Coalgebra in Computer Science (CALCO'15).
Editors: Lawrence S. Moss and Paweł Sobociński; pp. 86–100

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1.1 Traces and naive concurrent strategies

In standard causal models, execution traces essentially consist of partially ordered sets of atomic 'events'. Our approach relies on a new notion of trace, which we briefly sketch. There is first a (straightforward) notion of *configuration*, which is essentially a finite hypergraph whose nodes are thought of as *agents*, and whose hyperedges between nodes $x_1, \dots, x_n$ are thought of as communication channels shared by $x_1, \dots, x_n$. There is then a notion of *atomic action* from one configuration to another, thought of as a 'rule of the game'. Examples of atomic actions are: an agent creates a new, private communication channel; an agent forks into two new agents connected to the same channels; an agent sends some channel a over some channel b to some other agent. We finally have a notion of trace which allows several atomic actions to occur, in a way that only retains some minimal causality information between them. We here mean, e.g., information such as: 'such agent outputs on such channel only after having created such other channel'.

The main purpose of our notion of trace is to interpret π-calculus processes as some kind of strategies over them. Most naively, a strategy on some configuration X is a prefix-closed set of 'accepted' traces from X. But what should prefix mean in our setting? Well, we may view traces with initial configuration X and final configuration Y as morphisms $Y \multimap X$. Sequential composition of traces, denoted by $\bullet$, yields an analogue of prefix ordering, defined by $t \leq t \bullet w$. This however fails to suit our needs on three counts.

We start by examining the first two problems. The first, easy one is that there is an obvious notion of isomorphism between traces, under which strategies should be closed. The second problem is more serious: until now, these too naive strategies are not concurrent enough to adequately model CCS or the π-calculus.

▶ **Example 1** (Milner's coffee machines)**.** Consider the CCS processes $P = (a.b + a.c)$ and $Q = a.(b + c)$. The process P has two ways of inputting on a and then, depending on the chosen way, inputs either on b or on c. The process Q inputs on a and then has both possibilities of inputting on b or c. Both processes, however, accept exactly the same traces (in the standard sense), namely $\{\epsilon, a, ab, ac\}$, where ϵ denotes the empty trace.

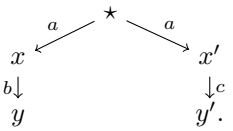
Thus, taking strategies to be prefix-closed sets of traces would prevent us from directly modelling any reasonably fine behavioural equivalence on processes. Inspired by *presheaf models* [26], we remedy both problems at once by passing from prefix-closed sets of traces to presheaves (of finite sets) on traces. Indeed, in the simple case where traces on X form a mere poset $\mathbb{T}(X)$ by prefix ordering, a prefix-closed set of traces is nothing but a contravariant functor from $\mathbb{T}(X)$ to the ordinal 2, viewed as a category. The latter has two objects 0 and 1 and just one non-trivial morphism $0 \to 1$. The idea is that a functor $S \colon \mathbb{T}(X)^{op} \to 2$ maps any trace to 1 when it is accepted, and to 0 otherwise. Furthermore, if $t \leq t'$, i.e., t is a prefix of t', then we have a morphism $t \to t'$ which should be mapped by S to some morphism $S(t') \to S(t)$. If t' is accepted then $S(t') = 1$, so this has to be a morphism $1 \to S(t)$. Because there are no morphisms $1 \to 0$, this entails $S(t) = 1$, hence prefix-closedness of the corresponding strategy.

Now our traces naturally form a proper category $\mathbb{T}(X)$, encompassing both prefix ordering and isomorphisms between traces, so we are led to considering functors $\mathbb{T}(X)^{op} \to 2$. This retains prefix-closedness and solves our first problem: for any $t \cong t'$, functoriality imposes $S(t) \cong S(t')$. Our second problem is then solved by replacing such functors with *presheaves*, i.e., functors $\mathbb{T}(X)^{op} \to \mathsf{Set}$.

▶ **Example 2.** In Example 1, the two ways that P has to accept inputting on a may be

reflected by mapping the trace a to some two-element set. More precisely, P may be modelled by the presheaf S defined on the left and pictured on the right:

- $S(\epsilon) = \{\star\}$,
- $S(a) = \{x, x'\}$,
- $S(ab) = \{y\}$,
- $S(ac) = \{y'\}$,
- S empty otherwise,
- $S(\epsilon \hookrightarrow a) = \{x \mapsto \star, x' \mapsto \star\}$,
- $S(a \hookrightarrow ab) = \{y \mapsto x\}$,
- $S(a \hookrightarrow ac) = \{y' \mapsto x'\}$,



Presheaves thus may 'accept a trace in several ways': the trace a is here accepted in two ways, x and x'. The process Q is of course modelled by identifying x and x'.

As it turns out, we actually only need *finitely many* ways of accepting each trace. Thus, we arrive at a first sensible notion of strategy given by presheaves of finite sets, i.e., functors $\mathbb{T}(X)^{op} \to \mathsf{set}$, where set denotes the category with as objects all finite subsets of $\mathbb{N}$, with all maps between them. We call them *(naive) strategies* in the sequel.

▶ **Notation 3.** *For any $\mathcal{C}$, let $\widehat{\mathcal{C}}$ denote the category of presheaves of finite sets over $\mathcal{C}$.*

1.2 Innocence as a sheaf condition

The third problem evoked above is that functors $\mathbb{T}(X)^{op} \to \mathsf{set}$ allow some undesirable behaviours. Intuitively, in π just as in CCS, agents should not have any control over the routing of messages.

▶ **Example 4.** Consider a configuration X with three agents x, y, and z sharing a communication channel a, and a strategy S accepting (1) the trace where x outputs on a, (2) the trace where y inputs on a, and (3) the trace where z inputs on a. Then, both synchronisations should be accepted by S. However, one easily constructs a naive strategy in which one is refused (see Example 19).

In order to rectify this deficiency, we enrich strategies with 'local' information. The idea is that a strategy should not only accept or refuse traces on the whole configuration X, but also traces on all possible subconfigurations of X. Furthermore, this local information should fit together coherently.

▶ **Example 5.** Consider the configuration X of Example 4. Any strategy on X should now in particular include independent strategies for each of the three agents x, y, and z. Coherence means that in order for a trace to be accepted, it should be enough for it to be 'locally accepted', i.e., at every stage in the trace, each agent should approve what she sees of the next action. E.g., if the next action is a synchronisation $x \to y$ with x outputting and y inputting on some channel a, then all that's required for the synchronisation to be accepted is that x accepts to output and y accepts to input. Consequently, if some other agent z also accepts to input on a at this stage, then the synchronisation $x \to z$ is also accepted.

We call this putative coherence condition *innocence* by analogy with Hyland and Ong's notion [24]. In order to formalise it, we first extend our category of traces $\mathbb{T}(X)$ on X with new objects representing traces on subconfigurations of X. We also add new morphisms, which are about 'locality':

▶ **Example 6.** Consider the configuration X with two unary agents x_1 and x_2. There is a trace t on X in which both agents fork. Consider now the subconfiguration Y of X consisting solely of x_1 and the trace t' on Y in which x_1 merely forks. There is a morphism $t' \to t$ in our new category.

This extended category, $\mathbb{T}_X$, yields an intermediate notion of strategy, given by functors $\mathbb{T}_X^{op} \to \mathsf{set}$. Among the new objects, we have in particular traces on just one agent of X, which are obtained by sequentially composing atomic actions whose final configuration again consists of one agent. We call this particular kind of trace a *view*. Views are the most 'local' kind of objects in $\mathbb{T}_X$. They form a subcategory $\mathbb{V}_X$ of $\mathbb{T}_X$.

▶ **Example 7.** If X merely consists of an agent x linked to n communication channels, consider the atomic action given by x forking into two new agents, say x_1 and x_2. This action, viewed as an object of $\mathbb{T}_X$ has three subobjects which are views: (1) the 'identity' view, in which nothing happens, (2) π_n^l, which represents the left-hand branch (to x_1), (3) and π_n^r, which represents the right-hand branch (to x_2).

The inclusion $\mathbb{V}_X \hookrightarrow \mathbb{T}_X$ induces a simple Grothendieck topology [30] on $\mathbb{T}_X$, which amounts to decreeing that any trace is covered by its views. We finally call any $S \colon \mathbb{T}_X^{op} \to \mathsf{set}$ *innocent* precisely when it is a *sheaf* for this Grothendieck topology. In particular, giving an innocent presheaf on $\mathbb{T}_X$ is equivalent (up to isomorphism) to separately giving an innocent presheaf for each agent of X, which rules out the undesirable behaviour described in Example 4.

Sheaves on $\mathbb{T}_X$ form a category S_X, which is small thanks to our use of set instead of Set. They furthermore map back to naive strategies, i.e., presheaves on $\mathbb{T}(X)$, by forgetting the local information. (This forgetful functor has a left adjoint.) Finally, because the considered topology is particularly simple, sheaves are equivalent to presheaves on views, i.e., $\mathsf{S}_X \simeq \widehat{\mathbb{V}_X}$ (recalling Notation 3). In summary, we have three categories of strategies: *naive* strategies are presheaves on traces $\widehat{\mathbb{T}(X)}$, *innocent* strategies S_X are sheaves on the extended category of traces $\mathbb{T}_X$, and so-called *behaviours* B_X are presheaves on the category of views $\mathbb{V}_X$. The last two are equivalent, and we furthermore have an adjunction $\widehat{\mathbb{T}(X)} \underset{\longleftarrow}{\overset{\longrightarrow}{\bot}} \mathsf{S}_X$.

We use both sides of the equivalence: behaviours directly lead to our compositional interpretation $[\![-]\!] \colon Pi \to \mathsf{S}$ of π-calculus processes, and innocent strategies are used below as the basis for our semantic definition of fair testing equivalence.

1.3 Main result

What should we do in order to demonstrate adequacy of our model? By definition, causal models expose some intensional information. Hence, equality is generally much finer than any reasonable behavioural equivalence, so we should not base our main result on it. On the other hand, causal models are supposed to be 'compositional', i.e., to come equipped with an interpretation of syntactic operations in the model. The natural thing to do is thus to choose some behavioural equivalence from the operational side, use compositionality to transpose it to the model, and prove that the two coincide. More precisely, the considered equivalence induces by quotienting two 'extensional collapses', one syntactic and the other semantic, and we want to prove that the translation $[\![-]\!]$ induces a bijection between both extensional collapses. Following [1], we call this *intensional full abstraction* for the considered equivalence.

We here focus on so-called *testing equivalences* [11, 35, 5, 38], which are defined in two stages. First, one chooses a 'mode of interaction'. That is, one defines what the relevant *tests* are for a given process and specifies how the two should interact. Typically, tests for P are other processes T with the same free communication channels as P, and interaction is just parallel composition $P \mid T$. The second stage amounts to choosing when $P \mid T$ is *successful*. E.g., in may testing equivalence $P \mid T$ is successful just when there exists a transition $(P \mid T) \overset{\heartsuit}{\Rightarrow} P'$ (that is, a $\heartsuit$ transition, possibly surrounded by silent transitions),

where $\heartsuit$ is some action fixed in advance. In must testing equivalence, success is when all maximal (possibly infinite) transition sequences contain at least one $\heartsuit$ transition. In fair testing equivalence (see [7] for some motivation and an adaptation to π), one requires that all silent sequences $(P \mid T) \Rightarrow P'$ extend to some sequence $P' \Rightarrow P'' \xrightarrow{\heartsuit} P'''$ ending with a $\heartsuit$ transition. In this paper, we focus on the latter, i.e., we prove (Theorem 25) that our model is intensionally fully-abstract for fair testing equivalence. However, we show in the long version [12] that our proof applies to a wide range of testing equivalences.

1.4 Contributions

Since this paper follows the same approach as previous work on CCS [23, 21, 22], we should explain in which sense extending the approach to π is more than an easy application.

A first contribution comes from the fact that, in order to even define composition in our category of traces (see our online draft [12] for details), we need to show that traces form the total category of a *fibration* [25] over configurations. In previous work, this was done in an *ad hoc* way. We here introduce a more satisfactory approach based on *factorisation systems* [28, 15].

A second significant contribution is prompted by the interplay between synchronisation and private channels in π, which is notoriously subtle to handle. And indeed, our proof method for CCS fails miserably on π. One reason for this, we think, is that our notion of trace for π, though simple and natural, is not 'modular' enough, in the sense that a trace contains strictly more information than the collection of all 'local' information accessible to agents (i.e., of all of its views, in the above sense). Thus, adapting our proof technique from CCS would have required us to define a much more complex but more modular notion of trace. Instead, we here take a somewhat rougher route, as sketched in Section 4.

Finally, as mentioned above, our proof now applies not only to fair testing equivalence, but also to a whole class of testing equivalences.

1.5 Related work

Beyond the obviously closely related, already mentioned work of Winskel et al., we should mention other causal and interleaving models for π, e.g., [34, 13, 4, 9, 10, 6, 14, 40, 19]. All of these models are based on some LTS for π. Instead, ours is rather based on *reduction rules*. The subtleties usually showing up in LTSs, related to mixing synchronisation and private channels, do resurface in our proof of intensional full abstraction, but not in the definition of our model. Indeed, it merely goes by describing the 'rule of the game' in π, and applying the general framework of *playgrounds* [22].

Another general framework relating operational and denotational descriptions of programs is Kleene coalgebra [3], which is mainly designed for automata theory. Playgrounds may be viewed as adapting ideas from Kleene coalgebra to the process algebraic setting.

We should also mention Laird's games model of (a fragment of) π [27], which accounts for *trace* (a.k.a. *may testing*) equivalence. Standard game models view strategies as *sets* of traces (with well-formedness conditions), so, as we have seen, lend themselves better to modelling trace equivalence. In a non-deterministic, yet not concurrent setting, Harmer and McCusker [18] resort to an explicit action for divergence, which allows them to recover a finer behavioural equivalence. We feel that the presheaf-based approach is more general.

Furthermore, recent work by Tsukada and Ong [41] adapts and extends some ideas of [23, 21] to nondeterministic, simply-typed λ-calculus.

Let us moreover mention less closely related work: Girard's *ludics* [16], Melliès's game semantics in string diagrams [33], Harmer et al.'s categorical combinatorics of innocence [18], McCusker et al.'s graphical foundation for schedules [31]. Finally, Hildebrandt's work [20] also uses sheaves, though for a quite different purpose.

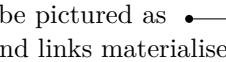
1.6 Plan

We describe our notion of trace at length in Section 2. We then sketch the model produced by the machinery of playgrounds, and state our main result in Section 3. We then conclude in Section 4, with a brief sketch of the proof and some future directions.

2 Traces

In this section, we introduce our notion of trace, which is based on certain combinatorial objects, close in spirit to string diagrams. We first define these string diagrams, and then use them to define traces. Configurations are special, hypergraph-like string diagrams whose vertices represent agents and whose hyperedges represent channels. A perhaps surprising point is that actions are not just a binary relation between configurations, because we not only want to say *when* there is an action from one configuration to another, but also *how* this action is performed. This will be implemented by viewing actions from X to Y as *cospans* $Y \to M \leftarrow X$ in a certain category $\widehat{\mathbb{C}}$, whose objects we call higher-dimensional string diagrams for lack of a better term. The idea is that X and Y respectively are the initial and final configurations, and that M describes how one goes from X to Y. By combining such actions (by pushout), we get a bicategory $\mathbb{D}_v$ of configurations and traces.

2.1 String diagrams

The category $\widehat{\mathbb{C}}$ will be a category of presheaves over a base category, $\mathbb{C}$. Let us motivate the definition of $\mathbb{C}$ by recalling that (directed, multi) graphs may be seen as presheaves over the category with two objects $\star$ and $[1]$, and two non-identity morphisms $s, t \colon \star \to [1]$. Any such presheaf G represents the graph with vertices in $G(\star)$ and edges in $G[1]$, the source and target of any $e \in G[1]$ being respectively $G(s)(e)$ and $G(t)(e)$, or $e \cdot s$ and $e \cdot t$ for short. A way to visualise how such presheaves represent graphs is to compute their *categories of elements* [30]. Recall that the category of elements $\int G$ for a presheaf G over $\mathbb{C}$ has as objects pairs (c, x) with $c \in \mathbb{C}$ and $x \in G(c)$, and as morphisms $(c, x) \to (d, y)$ all morphisms $f \colon c \to d$ in $\mathbb{C}$ such that $y \cdot f = x$. This category admits a canonical functor π_G to $\mathbb{C}$, and G is the colimit of the composite $\int G \xrightarrow{\pi_G} \mathbb{C} \xrightarrow{\mathsf{y}} \hat{\mathbb{C}}$ with the Yoneda embedding. E.g., the category of elements for $\mathsf{y}[1]$ is the poset $(\star, s) \xrightarrow{s} ([1], id_{[1]}) \xleftarrow{t} (\star, t)$, which could be pictured as  , where dots represent vertices, the triangle represents the edge, and links materialise the graph of $G(s)$ and $G(t)$, the convention being that t connects to the apex of the triangle. We thus recover some graphical intuition.

Our string diagrams will also be defined as particular presheaves over some base category $\mathbb{C}$. However, since we'll only be interested in *finite* structures, we restrict ourselves to the category $\widehat{\mathbb{C}}$ of presheaves of finite sets. In the case of graphs, presheaves of finite sets are graphs whose nodes and edges are identified by natural numbers. Such graphs are thus finite. In our case, the base category $\mathbb{C}$ is infinite, so presheaves of finite sets may represent infinite structures. However, our notion of trace will only involve finite ones.

Let us give the formal definition of $\mathbb{C}$ for reference. We advise to skip it on a first reading, as we then attempt to provide some graphical intuition.

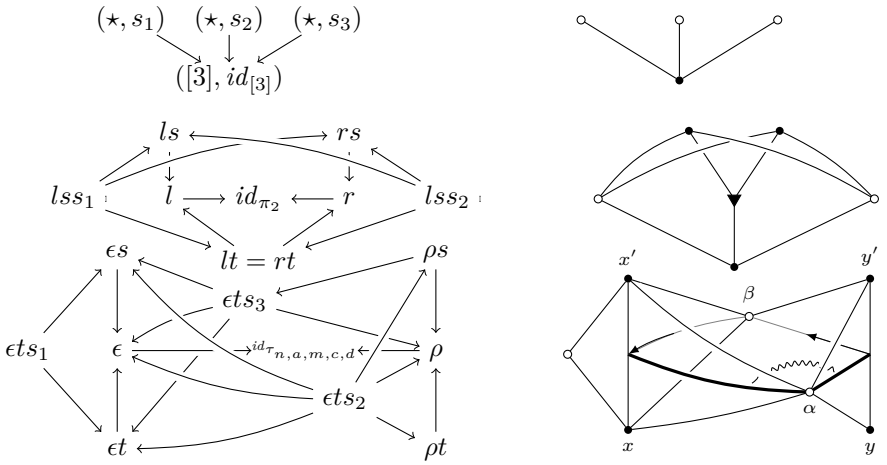


Figure 1 Categories of elements for [3], π_2, and $\tau_{1,1,3,2,3}$, with graphical representation.

▶ **Definition 8.** Let $G_{\mathbb{C}}$ be the graph with, for all n, m, with $a, b \in n$ and $c, d \in m$:

- vertices $\star$, $[n]$, π_n^l, π_n^r, π_n, ν_n, $\heartsuit_n$, τ_n, $\iota_{n,a}$, $o_{n,a,b}$, and $\tau_{n,a,m,c,d}$;
- edges $s_1, ..., s_n : \star \to [n]$, plus, $\forall v \in \{\pi_n^l, \pi_n^r, \heartsuit_n, \tau_n, o_{n,a,b}\}$, edges $s, t : [n] \to v$;
- edges $[n] \xrightarrow{t} \nu_n \xleftarrow{s} [n+1]$ and $[n] \xrightarrow{t} \iota_{n,a} \xleftarrow{s} [n+1]$;
- edges $\pi_n^l \xrightarrow{l} \pi_n \xleftarrow{r} \pi_n^r$ and $\iota_{n,a} \xrightarrow{\rho} \tau_{n,a,m,c,d} \xleftarrow{\epsilon} o_{m,c,d}$.

Let $\mathbb{C}$ be the free category on $G_{\mathbb{C}}$, modulo the equations

$$s \circ s_i = t \circ t_i \qquad l \circ t = r \circ t \qquad \rho \circ t \circ s_a = \epsilon \circ t \circ s_c \qquad \rho \circ s \circ s_{n+1} = \epsilon \circ s \circ s_d.$$

The first equation should be understood in $\mathbb{C}(\star, v)$ for all $n \in \mathbb{N}$, $i \in n$, and $v \in \cup_{a,b \in n}\{\pi_n^l, \pi_n^r, \heartsuit_n, \tau_n, \iota_{n,a}, o_{n,a,b}, \nu_n\}$. (This is rather elliptic: if v has the shape $\iota_{n,a}$ or ν_n, $s \circ s_i$ is really $\star \xrightarrow{s_i} [n+1] \xrightarrow{s} v$.) The second equation should be understood in $\mathbb{C}(\star, \pi_n)$ for all n, and the last two in $\mathbb{C}(\star, \tau_{n,a,m,c,d})$, for all n, m, $a \in n$, and $c, d \in m$.

Our category of string diagrams is the category of finite presheaves $\widehat{\mathbb{C}}$. To explain the design of $\mathbb{C}$, let us compute a few categories of elements. Let us start with an easy one, that of $[3] \in \mathbb{C}$ (we implicitly identify any $c \in \mathbb{C}$ with $\mathsf{y}c$). An easy computation shows that it is the poset pictured in the top left part of Figure 1. We think of it as a configuration with one agent $([3], id_{[3]})$ connected to three channels, and draw it as in the top right part, where the bullet represents the agent, and circles represent channels. In the presheaf, elements over [3] represent ternary agents, while elements over $\star$ represent channels. *Configurations* are finite presheaves empty except perhaps on $\star$ and $[n]$'s. Other objects will represent actions. A *morphism of configurations* is a morphism between presheaves which is injective except perhaps on channels. The intuition for a morphism $X \to Y$ between configurations is thus that X embeds into Y, possibly identifying some channels.

▶ **Definition 9.** Configurations and morphisms between them form a category $\mathbb{D}_h$.

A more difficult category of elements is that of π_2. It is the poset generated by the left-hand graph in the second row of Figure 1 (omitting base objects for conciseness). We think of it as a binary agent (lt) forking into two agents (ls and rs), and draw it as on the right. The graphical convention is that a black triangle stands for the presence of id_{π_2}, l,

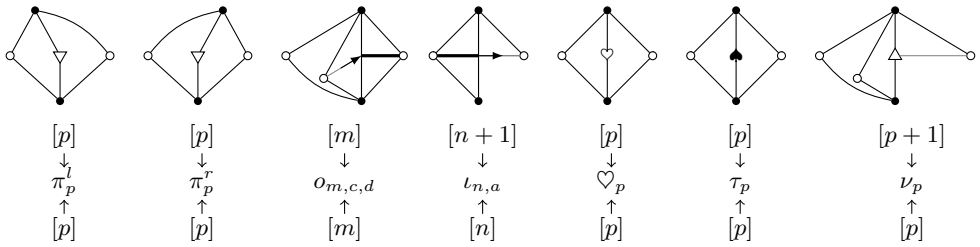


Figure 2 Pictures and corresponding cospans for π_p^l, π_p^r, $o_{m,c,d}$, $\iota_{n,a}$, $\heartsuit_p$, τ_p, and ν_p.

and r. Below, we represent just l as a white triangle with only a left-hand branch, and symmetrically for r. Furthermore, in all our pictures, time flows 'upwards'.

Another category of elements, characteristic of the π-calculus, is the one for synchronisation $\tau_{n,a,m,c,d}$. The case $(n, a, m, c, d) = (1, 1, 3, 2, 3)$ is the poset generated by the graph on the bottom left of Figure 1, which we will draw as on the right. The left-hand ternary agent x outputs its 3rd channel, here β, on its 2nd channel, here α. The right-hand unary agent y receives the sent channel on its 1st channel, here α. Both agents have two occurrences, one before and one after the action, respectively marked as x/x' and y/y'. Both x and x' are ternary here, while y is unary and y', having gained knowledge of β, is binary. There are actually three actions here, in the sense that there are three higher-dimensional objects. The first is the output action ϵ from x to x', graphically represented as the middle point of $\rightarrow\!\!-$ (intended to evoke the point where β enters channel α). The second is the input action ρ from y to y', graphically represented as the middle point of $-\!\!\rightarrow$ (where β exits channel α). The third action is the synchronisation itself, which 'glues' the other two together, as represented by the squiggly line.

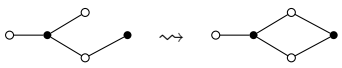
We leave the computation of other categories of elements as an exercise to the reader. The remaining string diagrams are depicted in the top row of Figure 2, for $p = 2$ and $(n, a, m, c, d) = (1, 1, 3, 2, 3)$.

The first two are *views*, in the game semantical sense, of the fork action π_2 explained above. The next two, $o_{m,c,d}$ (for 'output') and $\iota_{n,a}$ (for 'input'), respectively are views for the sender and receiver in a synchronisation action. The τ_p action is a silent, dummy action as standard in the π-calculus. The $\heartsuit_n$ action is a special 'tick' action used for defining fair testing equivalence. The last one is a channel creation action.

2.2 From string diagrams to actions

In the previous section, we have defined our category of string diagrams as $\widehat{\mathbb{C}}$, and provided some graphical intuition on its objects. The next step is to construct a bicategory whose objects are configurations, and whose morphisms represent traces. We start in this section by defining in which sense higher-dimensional objects of $\mathbb{C}$ represent actions, and continue in the next one by explaining how to compose actions to form traces. Actions are defined in two stages: *seeds*, first, give the local forms of actions, which are then defined by embedding seeds into bigger configurations.

To start with, until now, our string diagrams contain no information about the 'flow of time', although we mentioned it informally in the previous section. To add this information, for each string diagram M representing an action, we define its initial and final configurations, say X and Y, and view the whole action as a cospan $Y \xrightarrow{s} M \xleftarrow{t} X$.



We have taken care, in drawing our pictures before, of placing initial configurations at the bottom, and final configurations at the top. So, e.g., the initial and final configurations for the synchronisation action are pictured above and they map into (the representable presheaf over) $\tau_{1,1,3,2,3}$ in the obvious ways, yielding the cospan $Y \xrightarrow{s} M \xleftarrow{t} X$.

We leave it to the reader to define, based on the above pictures, the expected cospans for forking and synchronisation as on the right, plus the remaining ones specified in the bottom row of Figure 2 (where again $p = 2$ and $(n, a, m, c, d) = (1, 1, 3, 2, 3)$). Initial configurations are at the bottom, and we denote by $[m]\ {}_{a_1,\ldots,a_p}|_{c_1,\ldots,c_p}\ [n]$ the configuration consisting of an m-ary agent x and an n-ary agent y, quotiented by the equations $x \cdot s_{a_k} = y \cdot s_{c_k}$ for all $k \in p$. When both lists are empty, by convention, $m = n$ and the agents share all channels in order.

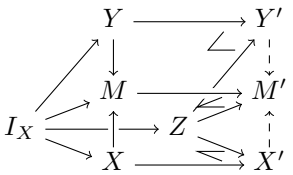
$$\begin{array}{cc} [p] \mid [p] & [m]\ {}_{c,d}|_{a,n+1}\ [n+1] \\ \downarrow & \downarrow \\ \pi_p & \tau_{n,a,m,c,d} \\ \uparrow & \uparrow \\ [p] & [m]\ {}_{c}|_{a}\ [n] \end{array}$$

▶ **Definition 10.** These cospans are called *seeds*.

We now define actions from seeds by embedding the latter into bigger configurations. E.g., we allow a fork action to occur in a configuration with more than one agent.

▶ **Definition 11.** The *interface* I_X of a seed $Y \xrightarrow{s} M \xleftarrow{t} X$ denotes the configuration consisting only of the channels of the initial configuration X.

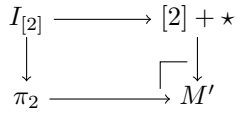
Since channels occurring in the initial configuration remain in the final one, we have for each seed a cone from I_X to the seed. For any morphism of positions $I_X \to Z$, pushing the cone along $I_X \to Z$ using the universal property of pushout as on the right yields a new cospan, say $Y' \to M' \leftarrow X'$.



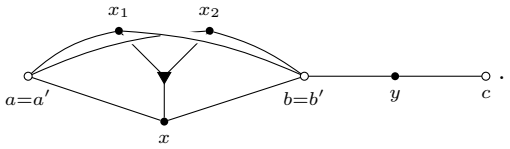
▶ **Definition 12.** Let *actions* be all such pushouts of seeds.

Intuitively, taking pushouts glues string diagrams together. Let us do a few examples.

▶ **Example 13.** The seed $[2] \mid [2] \xrightarrow{[ls,rs]} \pi_2 \xleftarrow{lt} [2]$ has as interface the presheaf $I_{[2]} = \star + \star$, consisting of two channels, say a and b. Consider the configuration $[2] + \star$ consisting of an agent y connected to two channels b' and c, plus an additional channel a'. Further consider the map $h\colon I_{[2]} \to [2] + \star$ defined by $a \mapsto a'$ and $b \mapsto b'$. The pushout



is



The meaning of such an action is that x forks while y is passive.

▶ **Example 14.** Because we push along *initial* channels, the interface of a seed may not contain all involved channels. E.g., in an input action (not part of any synchronisation), the received channel cannot be part of the initial configuration.

2.3 From actions to traces

Having defined actions, we now define their composition to yield our bicategory $\mathbb{D}_v$ of configurations and traces. Consider $\mathsf{Cospan}(\widehat{\mathbb{C}})$, the bicategory which has as objects all presheaves of finite sets on $\mathbb{C}$, as morphisms $X \to Y$ all cospans $X \to U \leftarrow Y$, and obvious 2-cells. Composition is given by pushout, and hence is not strictly associative.

Figure 3 Example traces.

▶ **Notation 15.** *By convention, the initial configuration is the* target *of the morphism in* $\mathsf{Cospan}(\widehat{\mathbb{C}})$*. We denote morphisms in* $\mathsf{Cospan}(\widehat{\mathbb{C}})$ *with special arrows* $X \multimap Y$*; composition and identities are denoted with* $\bullet$ *and* $id^{\bullet}$*.*

▶ **Definition 16.** A *trace* is a finite, possibly empty composite of actions in $\mathsf{Cospan}(\widehat{\mathbb{C}})$. Let $\mathbb{D}_v$ denote the locally full subbicategory of $\mathsf{Cospan}(\widehat{\mathbb{C}})$ with configurations as objects and traces as morphisms.

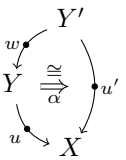
Thus, arrows $X \to Y$ in $\mathbb{D}_h$ denote embeddings of X into Y (up to identification of channels), whereas arrows $Y \multimap X$ in $\mathbb{D}_v$ denote traces with X initial and Y final. Intuitively, composition in $\mathbb{D}_v$ glues string diagrams on top of each other, which yields a truly concurrent notion of trace: the only information retained in a trace about the order of occurrence of actions is their causal dependencies.

▶ **Example 17.** Composing the action of Example 13 with a forking action by y yields the first string diagram of Figure 3, which shows that the ordering between remote actions is irrelevant. To illustrate how composition retains causal dependencies between actions, consider the second string diagram. It is unfolded for readability: one should identify both framed nodes, resp. both circled ones. In the initial configuration, there are channels a, b, and c, and three agents $x(a, b)$, $y(b)$, and $z(a, c)$ (channels known to each agent are in parentheses). In a first action, x sends a on b, and y receives it. In a second action, z sends c on a, and the avatar y' of y receives it. The second action is enabled by the first, by which y gains knowledge of a.

3 Strategies, behaviours, and semantic fair testing

3.1 Strategies and behaviours

We now investigate notions of strategies. As announced in the introduction, we define a category $\mathbb{T}(X)$ combining prefix ordering and isomorphism of traces: $\mathbb{T}(X)$ has traces $u \colon Y \multimap X$ as objects, and as morphisms $u \to u'$ all pairs (w, α) with $w \colon Y' \multimap Y$ and α an isomorphism $u \bullet w \to u'$ in the hom-category $\mathbb{D}_v(Y', X)$, as on the right[1]. Thus, u' is an extension of u by w.



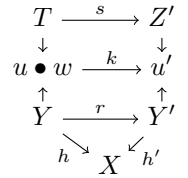
▶ **Definition 18.** Let the category of *(naive) strategies* on X be $\widehat{\mathbb{T}(X)}$.

Strategies do not yield a satisfactory model for π:

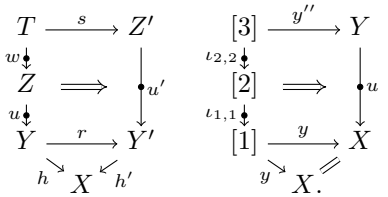
[1] There is a small problem, however: morphisms should only describe how u maps to u', not w. We actually quotient them out to rectify this.

▶ **Example 19.** Consider the configuration X with three agents x, y, z sharing a channel a, and the following traces on it: in $u_{x,y}$, x sends a on a, and y receives it; in $u_{x,z}$, x sends a on a, and z receives it; in i_z, z inputs on a. One may define a strategy S mapping $u_{x,y}$ and i_z to a singleton, and $u_{x,z}$ to $\emptyset$. Because $u_{x,y}$ is accepted, x accepts to send a on a; and because i_z is accepted, z accepts to input on a. The problem is that S rejecting $u_{x,z}$ roughly amounts to x refusing to synchronise with z, or conversely.

We want to rule out this kind of strategy from our model, by adapting the idea of innocence. We start by extending $\mathbb{T}(X)$ with objects representing traces on sub-configurations of X. For this, we consider the following category $\mathbb{T}_X$. It has as objects pairs (u, h) of a trace $u\colon Z \rightarrowtail\!\!\!\!\bullet\, Y$ and a morphism $h\colon Y \to X$ in $\mathbb{D}_h$. A morphism $(u, h) \to (u', h')$ consists of a trace $w\colon T \rightarrowtail\!\!\!\!\bullet\, Z$ and a triple (s, k, r) making the diagram on the right commute[1].



▶ **Example 20.** We adopt the convention of picturing the above diagram for morphisms as the left-hand diagram below: Now recalling the right-hand trace of Figure 3, say $u\colon Y \rightarrowtail\!\!\!\!\bullet\, X$, y's first action is an input on its unique channel b. This yields a trace $\iota_{1,1}\colon [2] \rightarrowtail\!\!\!\!\bullet\, [1]$.



There is a morphism $(\iota_{1,1}, y) \to (u, id_X)$ in $\mathbb{T}_X$, pictured as the right-hand diagram, which we think of as an occurrence of the trace $\iota_{1,1}$ in u. Thus, morphisms in $\mathbb{T}_X$ account both for prefix inclusion and for 'spatial' inclusion, i.e., inclusion of a trace into some other trace on a larger configuration.

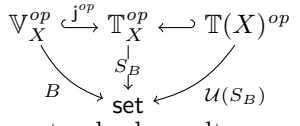
We now define *views* within $\mathbb{T}_X$:

▶ **Definition 21.** Let *basic* seeds be all seeds of any shape among $\iota_{n,a}$, $o_{n,a,b}$, ν_n, $\heartsuit_n$, τ_n, π^l_n, and π^r_n, for $a, b \in n$. *Views* are (possibly empty) composites of basic seeds in $\mathbb{D}_v$. Let $\mathbb{V}_X$ denote the full subcategory of $\mathbb{T}_X$ spanning pairs (u, h) where u is a view.

Intuitively, basic seeds follow exactly one agent through an action. An object of $\mathbb{V}_X$ consists of a view, say $v\colon [n'] \rightarrowtail\!\!\!\!\bullet\, [n]$, plus a morphism $h\colon [n] \to X$ in $\mathbb{D}_h$, which by Yoneda is just an agent of X. So an object of $\mathbb{V}_X$ is just an agent of X and a view from it.

▶ **Definition 22.** The inclusion $\mathbb{V}_X \hookrightarrow \mathbb{T}_X$ induces a Grothendieck topology, for which a family $(u_i \xrightarrow{\alpha_i} u)_{i \in I}$ of morphisms to some trace u is *covering* iff it contains all morphisms from views into u. Let the category $\mathsf{S}_X \hookrightarrow \widehat{\mathbb{T}_X}$ of *innocent* strategies be the category of sheaves of finite sets for this topology. Let the category B_X of *behaviours* over X be $\widehat{\mathbb{V}_X}$.

As promised, S_X and B_X are equivalent. We obtain the innocent strategy S_B associated to a behaviour $B \in \mathsf{B}_X$, by taking its *right Kan extension* [29] along the inclusion $\mathsf{j}^{op}\colon \mathbb{V}_X^{op} \hookrightarrow \mathbb{T}_X^{op}$, as on the right. Explicitly, using standard results, we obtain the *end* $S_B(u, h) = \int_{(v,x) \in \mathbb{V}_X} B(v, x)^{\mathbb{T}_X((v,x),(u,h))}$, which is a kind of generalised product. In the boolean setting (functors to $\mathsf{2}$), this end reduces to a conjunction $\bigwedge_{\{(v,x) \in \mathbb{V}_X \mid \exists \alpha\colon (v,x) \to (u,h)\}} B(v, x)$, demanding precisely that all views of u are accepted by B. In the general case, the intuition is that a way of accepting u for S_B is a compatible family of ways of accepting the views of u for B. Existence of the right Kan extension is proved in the general case in [22, Lemma 4.34], and follows from the general fact that the considered limits are essentially finite. The forgetful functor $\mathcal{U}$ to naive strategies is then given by restricting along $\mathbb{T}(X)^{op} \hookrightarrow \mathbb{T}_X^{op}$ as above right. Some local information may be forgotten by $\mathcal{U}$, which is neither injective on objects, nor full, nor faithful. E.g., if two

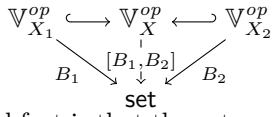


behaviours differ, but are both empty on the views of some agent, then both are mapped to the empty naive strategy.

▶ **Example 23.** Recalling X and S from Example 19, let us show that for any $B \in \mathsf{B}_X$, the associated strategy $\mathcal{U}(S_B) \in \widehat{\mathbb{T}(X)}$ cannot be S. Indeed, assuming it is, then because S accepts $u_{x,y}$ and i_z, B accepts the following views: (1) i_z, (2) o_x, in which x sends a on a (without any matching input), (3) i_y, in which y inputs on a, and (4) all identity views on x, y, and z. But then $\mathcal{U}(S_B)$ accepts both $u_{x,y}$ and $u_{x,z}$, because B accepts all views mapping into them.

3.2 Semantic fair testing

We now define our semantic analogue of fair testing equivalence, sketch our translation from π, and state our main result. Semantic fair testing rests on two main ingredients: a notion of *closed-world* trace, and an analogue of *parallel composition* in game semantics.



The intuitive purpose of parallel composition is to let strategies interact. If we partition the agents of a configuration X into two teams, we obtain two subconfigurations $X_1 \hookrightarrow X \hookleftarrow X_2$, each agent of X belonging to X_1 or X_2 according to its team. The crucial fact is that the category $\mathbb{V}_X$ of views on X is isomorphic to the coproduct category $\mathbb{V}_{X_1} + \mathbb{V}_{X_2}$. Parallel composition of any $B_1 \in \mathsf{B}_{X_1}$ and $B_2 \in \mathsf{B}_{X_2}$ is then simply given by copairing $[B_1, B_2]$ as above.

We now describe closed-world actions and traces, which are then used as a criterion for success of tests. *Closed-world* actions are those which do not involve interaction with the environment, i.e., formally, pushouts of a seed of any shape among ν_n,τ_n,$\heartsuit_n$,π_n, and $\tau_{n,a,m,c,d}$. A trace is *closed-world* when it is a composite of closed-world actions. Let $\mathbb{W}(X) \overset{\mathsf{i}}{\hookrightarrow} \mathbb{T}(X)$ denote the full subcategory of $\mathbb{T}(X)$ consisting of closed-world traces, and let the category of *closed-world strategies* be $\widehat{\mathbb{W}(X)}$. Further, denote by $B \mapsto \overline{B}$ the composite functor $\widehat{\mathbb{V}_X} \to \widehat{\mathbb{T}(X)} \xrightarrow{\Delta_{\mathsf{i}^{op}}} \widehat{\mathbb{W}(X)}$, where $\Delta_{\mathsf{i}^{op}}$ denotes restriction along i^{op}.

A closed-world trace is *successful* when it contains a $\heartsuit$ action, and *unsuccessful* otherwise. A state $\sigma \in S(u)$ of a strategy $S \in \widehat{\mathbb{W}(Z)}$ over a closed-world trace $u \colon Z' \to\!\!\!\!\bullet\, Z$ is successful iff u is. Define $\perp\!\!\!\perp_Z$ as the set of closed-world strategies $S \in \widehat{\mathbb{W}(Z)}$ such that any unsuccessful closed-world state admits a successful extension, i.e. $S \in \perp\!\!\!\perp_Z$ iff for all unsuccessful $u \in \mathbb{W}(Z)$ and $\sigma \in S(u)$, there exists a successful $u' \in \mathbb{W}(Z)$, a morphism $f \colon u \to u'$, and a state $\sigma' \in S(u')$ such that $\sigma' \cdot f = \sigma$. Finally, in order to compare behaviours for semantic fair testing equivalence, we specify what a test is for a given behaviour $B \in \mathsf{B}_X$. A *test* consists of a configuration Y and a behaviour $T \in \mathsf{B}_Y$. The behaviour B then *should pass* the test (Y, T) iff $I_X = I_Y$ and $\overline{[B, T]} \in \perp\!\!\!\perp_Z$, where I_X consists of all channels of X (recall Definition 11) and Z is the pushout $X +_{I_X} Y$ (X and Y thus form two teams on Z). At last, we define *semantic fair testing equivalence*, for any $B \in \mathsf{B}_X$ and $B' \in \mathsf{B}_{X'}$:

▶ **Definition 24.** Let $B \sim_f B'$ iff B and B' should pass the same tests.

3.3 Intensional full abstraction

We now sketch our translation from π-calculus processes to behaviours and state our main result. First, we consider processes to be infinite terms as generated by the grammar

$$P, Q ::= \textstyle\sum_{i \in n} G_i \quad | \quad (P \mid Q) \qquad\qquad G ::= \bar{a}\langle b\rangle.P \mid a(b).P \mid \nu a.P \mid \tau.P \mid \heartsuit.P,$$

up to renaming of bound variables as usual. Such a coinductive definition requires some care [12]: notably, processes come equipped with their finite set Γ of free channels, which we denote by $\Gamma \vdash P$. In order to translate processes to behaviours, we denote the coproduct in B_X by $\oplus$ (which is the pointwise coproduct of presheaves). Furthermore, let us denote by $\mathbb{B}_{[n]}$ the set of basic seeds $b\colon [n_b] \multimap [n]$ from $[n]$. For any family $(B_b)_{b\in\mathbb{B}_{[n]}}$ of behaviours, where each B_b is a behaviour over $[n_b]$, we denote by $\langle (B_b)_{b\in\mathbb{B}_{[n]}} \rangle$ the behaviour B' over $[n]$ such that $B'(id^{\bullet}_{[n]}, id_{[n]}) = 1$, and for any view $b \bullet v$, $B'(b \bullet v, id_{[n]}) = B_b(v, id_{[n_b]})$. Armed with this notation, we coinductively map processes with free channels in $\{1, \ldots, \Gamma\}$ for some $\Gamma \in \mathbb{N}$ to behaviours on $[\Gamma]$ like so:

$$[\![\Gamma \vdash \textstyle\sum_i \alpha_i.P_i]\!] = \langle b \mapsto \oplus_{\{i \mid [\![\alpha_i]\!]=b\}} [\![\Gamma \cdot \alpha_i \vdash P_i]\!] \rangle \quad [\![\Gamma \vdash P \mid Q]\!] = \left\langle \begin{array}{lcl} \pi^l_\Gamma & \mapsto & [\![\Gamma \vdash P]\!] \\ \pi^r_\Gamma & \mapsto & [\![\Gamma \vdash Q]\!] \end{array} \right\rangle,$$

where (1) $[\![\bar{a}\langle b\rangle]\!] = o_{\Gamma,a,b}$, $[\![a(b)]\!] = \iota_{\Gamma,a}$, $[\![\nu a]\!] = \nu_\Gamma$, $[\![\heartsuit]\!] = \heartsuit_\Gamma$, and $[\![\tau]\!] = \tau_\Gamma$, (2) all unmentioned basic seeds are mapped to the everywhere empty behaviour $\emptyset$, (3) $\Gamma \cdot \alpha_i$ denotes $\Gamma + 1$ when α_i is an input or a channel creation[2], and Γ otherwise. E.g., we have $[\![\Gamma \vdash a(b).P + a(b).Q]\!] = \langle \iota_{\Gamma,a} \mapsto [\![\Gamma + 1 \vdash P]\!] \oplus [\![\Gamma + 1 \vdash Q]\!] \rangle$. We then define fair testing equivalence $\sim_f^{Pi}$ for π-calculus processes as in [7]: let $\perp^{Pi}$ denote the set of processes P such that for all $P \Rightarrow P'$ there exists $P' \Rightarrow P'' \xrightarrow{\heartsuit} P'''$, and, given any two processes $\Gamma \vdash P$ and $\Gamma \vdash Q$, let $P \sim_f^{Pi} Q$ iff for all $\Gamma \vdash T$ we have $(P \mid T \in \perp^{Pi}) \Leftrightarrow (Q \mid T \in \perp^{Pi})$. Finally, our main result is:

▶ **Theorem 25.**
1. *For all* P, Q, $(\Gamma \vdash P) \sim_f^{Pi} (\Gamma \vdash Q)$ *iff* $[\![\Gamma \vdash P]\!] \sim_f [\![\Gamma \vdash Q]\!]$.
2. *For all* B *over* $[\Gamma]$, *there exists a process* $\Gamma \vdash P$ *such that* $[\![\Gamma \vdash P]\!] \sim_f B$.

4 Conclusion and future work

We have described our notion of trace and the induced model of π. We then have stated our main theorem. In our online long version [12], the interested reader may find the proof that our traces organise into a playground [22], and the proof of Theorem 25. For lack of space, we cannot give any detail. Still, we sketch the latter.

The idea is to reduce semantic fair testing equivalence to fair testing equivalence in the standard sense for some *ad hoc* LTS $\mathcal{S}$. We then single out a particular quotient $\mathcal{M}$ of $\mathcal{S}$, which admits a syntactic description very close to Berry and Boudol's *chemical abstract machine* [2], though with a kind of persistent explicit substitutions. Elements of $\mathcal{M}$ thus roughly consist of finite multisets of *molecules*. Multisets are here thought of as *chemical soups*, in which synchronisation is viewed as interaction between compatible molecules. In order to simplify matters, we also work with a chemical abstract machine presentation of the π-calculus. We then define a candidate 'pseudo-inverse' ζ to the translation map $[\![-]\!]$. These are maps between molecules for π and molecules for $\mathcal{M}$, which extend straightforwardly to chemical soups. We finally design a relation between π-calculus soups and $\mathcal{M}$ soups, which modularly allows π-calculus processes to correspond to their translation, and $\mathcal{M}$-molecules to correspond to their image under ζ. We are then able to show that this relation is a weak bisimulation which straightforwardly entails that $[\![-]\!]$ both preserves and reflects fair testing equivalence and is surjective up to fair testing equivalence, i.e., is intensionally fully-abstract.

Regarding future work, we of course plan to extend our approach to more complex calculi, e.g., calculi with passivation or functional calculi, and eventually consider some full-fledged

[2] W.l.o.g., we choose representatives of processes so that if $\Gamma \vdash a(b).P$ or $\Gamma \vdash \nu b.P$, then $b = (\Gamma + 1)$.

functional language with concurrency primitives. Furthermore, we consider generalising our technique for constructing playgrounds and applying them, e.g., to graph rewriting, error diagnostics, or efficient machine representation of reversible π-calculus processes.

References

1 Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Inf. Comput.*, 163(2):409–470, 2000.

2 Gérard Berry and Gérard Boudol. The chemical abstract machine. In *POPL*, pages 81–94, 1990.

3 Filippo Bonchi, Marcello M. Bonsangue, Jan J. M. M. Rutten, and Alexandra Silva. Deriving syntax and axioms for quantitative regular behaviours. In *CONCUR*, volume 5710 of *LNCS*, pages 146–162. Springer, 2009.

4 Michele Boreale and Davide Sangiorgi. A fully abstract semantics for causality in the π-calculus. *Acta Inf.*, 35(5):353–400, 1998.

5 Ed Brinksma, Arend Rensink, and Walter Vogler. Fair testing. In *CONCUR*, volume 962 of *LNCS*, pages 313–327. Springer, 1995.

6 Nadia Busi and Roberto Gorrieri. Distributed semantics for the π-calculus based on Petri nets with inhibitor arcs. *J. Log. Algebr. Program.*, 78(3):138–162, 2009.

7 Diletta Cacciagrano, Flavio Corradini, and Catuscia Palamidessi. Explicit fairness in testing semantics. *LMCS*, 5(2), 2009.

8 Simon Castellan, Pierre Clairambault, and Glynn Winskel. The parallel intensionally fully abstract games model of pcf. In *LICS*. IEEE, 2015.

9 Gian Luca Cattani and Peter Sewell. Models for name-passing processes: Interleaving and causal. In *LICS*, pages 322–333. IEEE, 2000.

10 Silvia Crafa, Daniele Varacca, and Nobuko Yoshida. Event structure semantics of parallel extrusion in the pi-calculus. In *FoSSaCS*, volume 7213 of *LNCS*, pages 225–239. Springer, 2012.

11 Rocco De Nicola and Matthew Hennessy. Testing equivalences for processes. *TCS*, 34:83–133, 1984.

12 Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller. A πlayground. `http://lama.univ-smb.fr/~hirschowitz/papers/pilayground.pdf`, 2015.

13 Joost Engelfriet. A multiset semantics for the pi-calculus with replication. *TCS*, 153(1&2):65–94, 1996.

14 Marcelo P. Fiore, Eugenio Moggi, and Davide Sangiorgi. A fully-abstract model for the pi-calculus (extended abstract). In *LICS*, pages 43–54. IEEE, 1996.

15 Peter Freyd and G. M. Kelly. Categories of continuous functors, I. *JPAA*, 2:169–191, 1972.

16 Jean-Yves Girard. Locus solum: From the rules of logic to the logic of rules. *MSCS*, 11(3):301–506, 2001.

17 Gregor Gößler, Daniel Le Métayer, and Jean-Baptiste Raclet. Causality analysis in contract violation. In *Runtime Verification*, volume 6418 of *LNCS*, pages 270–284. Springer, 2010.

18 Russell Harmer, Martin Hyland, and Paul-André Melliès. Categorical combinatorics for innocent strategies. In *LICS*, pages 379–388. IEEE, 2007.

19 Matthew Hennessy. A fully abstract denotational semantics for the π-calculus. *TCS*, 278(1-2):53–89, 2002.

20 Thomas T. Hildebrandt. Towards categorical models for fairness: fully abstract presheaf semantics of SCCS with finite delay. *TCS*, 294(1/2):151–181, 2003.

21 Tom Hirschowitz. Full abstraction for fair testing in CCS. In *CALCO*, volume 8089 of *LNCS*, pages 175–190. Springer, 2013.

22 Tom Hirschowitz. Full abstraction for fair testing in CCS (expanded version). *LMCS*, 10(4), 2014.

23 Tom Hirschowitz and Damien Pous. Innocent strategies as presheaves and interactive equivalences for CCS. In *ICE*, pages 2–24, 2011.

24 J. M. E. Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II, and III. *Inf. Comput.*, 163(2):285–408, 2000.

25 Bart Jacobs. *Categorical Logic and Type Theory*. Number 141 in Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1999.

26 André Joyal, Mogens Nielsen, and Glynn Winskel. Bisimulation and open maps. In *LICS*, pages 418–427. IEEE, 1993.

27 James Laird. Game semantics for higher-order concurrency. In *FSTTCS*, volume 4337 of *LNCS*, pages 417–428. Springer, 2006.

28 Saunders Mac Lane. Duality for groups. *Bull. AMS*, 56, 1950.

29 Saunders Mac Lane. *Categories for the Working Mathematician*. Number 5 in Graduate Texts in Mathematics. Springer, 2nd edition, 1998.

30 Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Universitext. Springer, 1992.

31 Guy McCusker, John Power, and Cai Wingfield. A graphical foundation for schedules. *ENTCS*, 286:273–289, 2012.

32 Paul-André Melliès. Asynchronous games 4: A fully complete model of propositional linear logic. In *LICS*, pages 386–395. IEEE, 2005.

33 Paul-André Melliès. Game semantics in string diagrams. In *LICS*, pages 481–490. IEEE, 2012.

34 Ugo Montanari and Marco Pistore. Concurrent semantics for the π-calculus. *ENTCS*, 1:411–429, 1995.

35 V. Natarajan and Rance Cleaveland. Divergence and fair testing. In *ICALP*, volume 944 of *LNCS*, pages 648–659. Springer, 1995.

36 Hanno Nickau. Hereditarily sequential functionals. In *LFCS*, volume 813 of *LNCS*, pages 253–264. Springer, 1994.

37 M. Nielsen, G. Plotkin, and G. Winskel. Event structures and domains, part 1. *TCS*, 13:65–108, 1981.

38 Arend Rensink and Walter Vogler. Fair testing. *Inf. Comput.*, 205(2):125–198, 2007.

39 Silvain Rideau and Glynn Winskel. Concurrent strategies. In *LICS*, pages 409–418. IEEE, 2011.

40 Ian Stark. A fully abstract domain model for the π-calculus. In *LICS*, pages 36–42. IEEE, 1996.

41 Takeshi Tsukada and C.-H. Luke Ong. Nondeterminism in game semantics via sheaves. In *LICS*. IEEE, 2015.

42 Glynn Winskel. Event structure semantics for CCS and related languages. In Mogens Nielsen and Erik Meineche Schmidt, editors, *ICALP*, volume 140 of *LNCS*, pages 561–576. Springer, 1982.

43 Glynn Winskel. Strategies as profunctors. In *FoSSaCS*, volume 7794 of *LNCS*, pages 418–433. Springer, 2013.