Innocent strategies as presheaves and interactive equivalences for CCS

Tom Hirschowitz *
CNRS, Université de Savoie
Chambéry, France

Damien Pous
CNRS, Laboratoire d'Informatique de Grenoble
Grenoble, France

Seeking a general framework for reasoning about and comparing programming languages, we derive a new view of Milner's CCS [30]. We construct a category $\mathbb{E}$ of *plays*, and a subcategory $\mathbb{V}$ of *views*. We argue that presheaves on $\mathbb{V}$ adequately represent *innocent* strategies, in the sense of game semantics [19]. We then equip innocent strategies with a simple notion of interaction. This results in an interpretation of CCS.

Based on this, we propose a notion of *interactive equivalence* for innocent strategies, which is close in spirit to Beffara's interpretation [1] of testing equivalences [6] in concurrency theory. In this framework we prove that the analogues of *fair* and *must* testing equivalences coincide, while they differ in the standard setting.

1 Overview

Theories of programming languages Research in programming languages is mainly technological. Indeed, it heavily relies on techniques which are ubiquitous in the field, but almost never formally made systematic. Typically, the definition of a language then quotiented by variable renaming (α-conversion) appears in many theoretical papers about functional programming languages. Why isn't there yet any abstract framework performing these systematic steps for you? Because the quest for a real *theory* of programming languages is not achieved yet, in the sense of a corpus of results that actually help developing them or reasoning about them. However, many attempts at such a theory do exist.

A problem for most of them is that they do not account for the dynamics of execution, which limits their range of application. This is for example the case of Fiore et al.'s second-order theories [9, 16, 17]. A problem for most of the other theories of programming languages is that they neglect denotational semantics, i.e., they do not provide a notion of model for a given language. This is for example the case of Milner et al.'s *bigraphs* [21], or of most approaches to *structural operational semantics* [32], with the notable exception of the *bialgebraic semantics* of Turi and Plotkin [36]. A recent, related, and promising approach is *Kleene coalgebra*, as advocated by Bonsangue et al. [2]. Finally, *higher-order rewriting* [31], and its semantics in double categories [11] or in cartesian closed 2-categories [18], is not currently known to adequately account for process calculi.

Towards a new approach The most relevant approaches to us are bialgebraic semantics and Kleene coalgebra, since the programme underlying the present paper concerns a possible alternative. A first difference, which is a bit technical but may be of importance, is that both bialgebraic semantics and Kleene coalgebra are based on labelled transition systems (LTSs), while our approach is based on reduction semantics. Reduction semantics is often considered more primitive than LTSs, and much work has been

*Both authors have been partially funded by the French projects CHoCo (ANR-07-BLAN-0324), PiCoq (ANR 2010 BLAN 0305 01), and CNRS PEPS CoGIP.

Bliudze, S., Bruni, R., Carbone, M., Silva, A. (Eds.); ICE 2011
EPTCS 59, 2011, pp. 2–24, doi:10.4204/EPTCS.59.2

devoted to deriving the latter from the former [35, 21, 34, 33]. It might thus be relevant to propose a model based only on the more primitve reduction semantics.

More generally, our approach puts more emphasis on interaction between programs, and hence is less interesting in cases where there is no interaction. A sort of wild hope is that this might lead to unexpected models of programming languages, e.g., physical ones. This could also involve finding a good notion of morphism between languages, and possibly propose a notion of compilation. At any rate, the framework is not set up yet, so investigating the precise relationship with bialgebraic semantics and Kleene coalgebra is deferred to further work.

How will this new approach look like? Compared to such long-term goals, we only take a small step forward here, by considering a particular case, namely Milner's CCS [30], and providing a new view of it. This view borrows ideas from the following lines of research: game semantics [19], and in particular the notion of an *innocent strategy* and *graphical games* [7, 15], Krivine realisability [25], ludics [12], testing equivalences in concurrency [6, 1], the presheaf approach to concurrency [22, 23], and sheaves [28]. But it is also related to, e.g., graph rewriting [8], and computads [3].

From games to presheaves Game semantics [19] has provided fully complete models of programming languages. However, it is based on the notion of a *strategy*, i.e., a set of *plays* in the game, satisfying a few conditions. In concurrency theory, taking as a semantics the set of accepted plays, or 'traces', is known as *trace semantics*. Trace semantics is generally considered too coarse, since it equates, for a most famous example, the right and the wrong coffee machines, $a.(b+c)$ and $ab+ac$ [30].

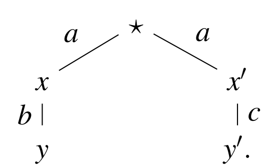
An observation essentially due to Joyal, Nielsen, and Winskel is that strategies, i.e., prefix-closed sets of plays, are actually particular *presheaves of booleans* on the category $\mathbb{C}$ with plays as objects, and prefix inclusions as morphisms. By presheaves of booleans on $\mathbb{C}$ we here mean functors $\mathbb{C}^{op} \to 2$, where 2 is the preorder category $0 \leq 1$. If a play p is *accepted*, i.e., mapped to 1, then its prefix inclusions $q \hookrightarrow p$ are mapped to the unique morphism with domain 1, i.e., id_1, which entails that q is also accepted.

We consider instead presheaves (of sets) on $\mathbb{C}$. So, a play p is now mapped to a set $S(p)$, to be thought of as the set of ways for p to be accepted by the strategy S. Considering the set of players as a team, $S(p)$ may also be thought of as the set of *states* of the team after playing p.

Presheaves are fine enough to account for bisimilarity [22, 23]. Indeed, they are essentially forests with edges labelled in moves. For example, in the setting where plays are finite words on an alphabet, the wrong coffee machine may be represented by the presheaf S defined by the equations on the left and pictured as on the right:

- $S(\varepsilon) = \{\star\}$,
- $S(a) = \{x, x'\}$,
- $S(ab) = \{y\}$,
- $S(ac) = \{y'\}$,

- $S(\varepsilon \hookrightarrow a) = \{x \mapsto \star, x' \mapsto \star\}$,
- $S(a \hookrightarrow ab) = \{y \mapsto x\}$,
- $S(a \hookrightarrow ac) = \{y' \mapsto x'\}$:



So, in summary: the standard notion of strategy may be generalised to account for branching equivalences, by passing from presheaves of booleans to presheaves of sets.

Multiple players Traditional game semantics mostly emphasises two-player games. There is an implicit appearance of three-player games in the definition of composition of strategies, and of four-player games in the proof of its associativity, but these games are never given a proper status. A central idea of graphical games, and to a lesser extent of ludics, is the emphasis on multiple-player games.

Here, there first is a base category $\mathbb{B}$ of *positions*, whose objects represent configurations of players to which the game may arrive at. Since the game represents CCS, it should be natural that players are related to each other via the knowledge of *communication channels*. So, positions are bipartite graphs with vertex sets *players* and *channels*, and edges from channels to players indicating when the former is known to the latter. As a first approximation, morphisms of positions may be thought of as just embeddings of such graphs.

Second, there is a category $\mathbb{E}$ of *plays*, with a functor to $\mathbb{B}$ sending each play to its initial position. Plays are represented in a more flexible way than just sequences of moves, namely using a kind of string diagrams. This echoes the idea [29] that two moves may be independent, and that plays should not depend on the order in which two independent moves are performed. Furthermore, our plays are a rather general notion, allowing, e.g., to look at how only some players of the initial position evolve. Morphisms of plays account both for:

- prefix inclusion, i.e., inclusion of a play into a longer play, and
- position enlargement, e.g., inclusion of information about some players into information about more players.

Now, restricting to plays above a given position X, and then taking presheaves on this category $\mathbb{E}_X$, we have a category of strategies on X.

Innocence A fundamental idea of game semantics is the notion of *innocence*, which says that players have a restricted *view* of the play, and that their actions may only depend on that view.

We implement this here by defining a subcategory $\mathbb{V}_X \hookrightarrow \mathbb{E}_X$ of *views* on X, and deeming a presheaf F on $\mathbb{E}_X$ *innocent* when it is determined by its restriction F' to $\mathbb{V}_X$, in the sense that it is isomorphic to the *right Kan extension* [27] of F' along $\mathbb{V}_X^{op} \hookrightarrow \mathbb{E}_X^{op}$.

Given this, it is sensible to define innocent strategies to be just presheaves on $\mathbb{V}_X$, and view them as strategies via the (essential) embedding $\widehat{\mathbb{V}_X} \hookrightarrow \widehat{\mathbb{E}_X}$ induced by right Kan extension.

Interaction For each position X, we thus have a category $\mathsf{S}_X = \widehat{\mathbb{V}_X}$ of innocent strategies. In game semantics, composition of strategies is achieved in two steps: *interaction* and *hiding*. Essentially, interaction amounts to considering the three-player game obtained by letting two two-player games interact at a common interface. Hiding then forgets what happens at that interface, to recover a proper two-player game.

We have not yet investigated hiding in our approach, but, thanks to the central status of multiple-player games, interaction is accounted for in a very streamlined way. For any position X with two subpositions $X_1 \hookrightarrow X$ and $X_2 \hookrightarrow X$ such that each player is in either X_1 or X_2, but none is in both, given strategies $F_1 \in \mathsf{S}_{X_1}$ and $F_2 \in \mathsf{S}_{X_2}$, there is a unique innocent strategy (up to canonical isomorphism in S_X), the *amalgamation* $[F_1, F_2]$ of F_1 and F_2, whose restrictions to X_1 and X_2 are F_1 and F_2 (again up to isomorphism).

Amalgamation in this sense models interaction in the sense of game semantics, and, using the correspondence with presheaves on $\mathbb{E}_X$ given by right Kan extension, it is the key to defining interactive equivalences.

CCS Next, we define a translation of CCS terms with recursive equations into innocent strategies. This rests on *spatial* and *temporal* decomposition results, which entail that innocent strategies are a solution of a system of equations of categories (up to equivalence). A natural question is then: which equivalence does this translation induce on CCS terms ?

Interactive equivalences Returning to the development of our approach, we then define a notion of *interactive equivalence*, which is close in spirit to both testing equivalences in concurrency theory and Krivine realisability and ludics.

The game, as sketched above, allows interacting with players which are not part of the considered position. E.g., a player in the considered position X may perform an input which is not part of any synchronisation. A *test* for a strategy F on X is then, roughly, a strategy G on a position X' with the same names as X. To decide whether F *passes* the test G, we consider a restricted variant of the game on the 'union' $X \cup X'$, forbidding any interaction with the outside. We call that variant the *closed-world* game. Then F passes G iff the amalgamation $[F,G]$, right Kan extended to $\mathbb{E}_{X \cup X'}$ and then restricted to the closed-world game, belongs to some initially fixed class of strategies, $\perp\!\!\!\perp_{X \cup X'}$. Finally, two strategies F and F' on X are equivalent when they pass the same tests.

Examples of $\perp\!\!\!\perp$ include:

- $\perp\!\!\!\perp^m$, consisting of all strategies whose maximal states (those that admit no strict extensions) all play a *tick* move, fixed in advance; the tick move plays a rôle analogous to the daimon in ludics: it is the only move which is observable from the outside;
- $\perp\!\!\!\perp^f$, consisting of all strategies in which all states admit an extension playing tick.

From the classical concurrency theory point of view on behavioural equivalences, the first choice mimicks *must* testing equivalence, while the second mimicks *fair* testing equivalence [6]. As a somehow surprising result, we prove that $\perp\!\!\!\perp^f$ and $\perp\!\!\!\perp^m$ yield the same equivalence. The reason is that our notion of play is more flexible than just sequences of moves, as we explain in Section 4.3.

Summary In summary, our approach emphasises a flexible notion of multiple-player play, encompassing both views in the sense of game semantics, closed-world plays, and intermediate notions. Strategies are then described as presheaves on plays, while innocent strategies are presheaves on views. Innocent strategies admit a notion of interaction, or amalgamation, and are embedded into strategies via right Kan extension. This allows a notion of testing, or interactive equivalence by amalgamation with the test, right Kan extension, and finally restriction to closed-world.

Our main technical contributions are then a translation of CCS terms with recursive equations into innocent strategies, and the result that fair and must equivalence coincide in our setting.

Perspectives So, we have defined a flexible category of multiple-player play, combining inclusion in time (more moves) and in width (more players). Having isolated a subcategory of views, we have defined innocent strategies as presheaves on views, relative to a base position. We have then translated CCS processes with recursive definitions into innocent strategies. Then, using right Kan extension and restriction to so-called closed-world plays, we have defined a notion of interactive equivalence. Finally, we have proved that two interactive equivalences, fair and must testing, coincide.

Our next task is clearly to tighten the link with CCS. Namely, we should explore which equivalence on CCS is induced via our translation, for a given interactive equivalence. We will start with $\perp\!\!\!\perp^m$. Furthermore, the very notion of interactive equivalence might deserve closer consideration. Its current form is rather *ad hoc*, and one could hope to see it emerge more naturally from the game. For instance, the fixed class $\perp\!\!\!\perp$ of 'successful' strategies should probably be subjected to more constraints than is done here, but two examples were not enough to make any guess. Also, the paradigm of observing via the set of successful tests might admit sensible refinements, e.g., probabilistic ones.

Another possible research direction is to tighten the link with 'graphical' approaches to rewriting, such as graph rewriting or computads. E.g., our plays might be presented by a computad [14], or be the

Category	Description of its objects
$\widehat{\mathbb{C}}$	'diagrams'
$\mathbb{B} \hookrightarrow \widehat{\mathbb{C}}$	positions
$\mathbb{E} \hookrightarrow (\mathbb{B} \downarrow_{\widehat{\mathbb{C}}} \widehat{\mathbb{C}})$	plays
$\mathbb{E}_X = (\mathbb{E} \downarrow_{\mathbb{B}} (\mathbb{B}/X))$	plays on a position X
$\mathbb{V}_X \hookrightarrow \mathbb{E}_X$	views on X
$\mathsf{S}_X = \widehat{\mathbb{V}_X}$	innocent strategies on X
$\mathbb{W} \hookrightarrow \mathbb{E}$	closed-world plays
$\mathbb{W}(X)$	closed-world plays on X

Table 1: Summary of categories and functors

bicategory of rewrite sequences up to shift equivalence, generated by a graph grammar in the sense of Gadducci et al. [10]. Both goals might require some technical adjustments, however. For computads, we would need the usual yoga of U-turns to flexibly model our positions; however, e.g., zigzags of U-turns are usually only equal up to a higher-dimensional cell, while they would map to equal positions in our setting. For graph rewriting, the problem is that our positions are not exactly graphs (e.g., the channels known to a player are totally ordered).

Other perspectives include the treatment of more complicated calculi like π or λ. In particular, calculi with duplication of terms will pose a serious challenge. An even longer-term hope is to be able to abstract over our approach. Is it possible to systematise the process starting from a calculus as studied in programming language theory, and generating its strategies modulo interactive equivalence? If this is ever understood, the next question is: when does a translation between two such calculi preserve a given interactive equivalence? Finding general criteria for this might have useful implications in programming languages, especially compilation.

Notation The various categories and functors used in the development are summed up with a short description in Table 1. There, given two functors $\mathbb{C} \xrightarrow{F} \mathbb{E} \xleftarrow{G} \mathbb{D}$, we denote (slightly abusively) by $\mathbb{C} \downarrow_{\mathbb{E}} \mathbb{D}$ the *comma* category: it has as objects triples (C,D,u) with $C \in \mathbb{C}$, $D \in \mathbb{D}$, and $u\colon F(C) \to G(D)$ in $\mathbb{E}$, and as morphisms $(C,D,u) \to (C',D',u')$ pairs (f,g) making the square on the right commute. Also, when F is the identity on $\mathbb{C}$ and $G\colon 1 \to \mathbb{C}$ is an object C of $\mathbb{C}$, this yields the usual *slice* category, which we abbreviate as $\mathbb{C}/C$. Finally, the category of presheaves on any category $\mathbb{C}$ is denoted by $\widehat{\mathbb{C}} = \mathsf{Set}^{\mathbb{C}^{op}}$.

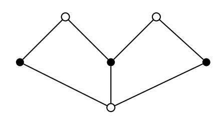
$$\begin{array}{ccc} FC & \xrightarrow{F(f)} & FC' \\ u\downarrow & & \downarrow u' \\ GD & \xrightarrow[G(g)]{} & GD' \end{array}$$

Furthermore, we denote, for any category $\mathbb{C}$, by $\mathrm{ob}(\mathbb{C})$ its set of obects and by $\mathrm{mor}(\mathbb{C})$ its set of morphisms. For any functor $F\colon \mathbb{C} \to \mathbb{D}$, we denote by $\mathbb{F}^{op}\colon \mathbb{C}^{op} \to \mathbb{D}^{op}$ the functor induced on opposite categories, defined exactly as F on both objects and morphisms. Also, recall that an *embedding* of categories is an injective-on-objects, faithful functor. This admits the following generalisation: a functor $F\colon \mathbb{C} \to \mathbb{D}$ is *essentially injective on objects* when $FC \cong FC'$ implies $C \cong C'$. Any faithful, essentially injective on objects functor is called an *essential embedding*.

2 Plays as string diagrams

We now describe our approach more precisely, starting with multiple-player plays. We remain at a not completely formal level, especially for presenting plays, because our experience is that most readers get stuck on that. However, the interested reader may have a look at the formal definition in Appendix A.

2.1 Positions



Since the game represents CCS, it should be natural that players are related to each other via the knowledge of *communication channels*. This is represented by a kind of[1] bipartite graph: an example position is on the right. Bullets represent players, circles represent channels, and edges indicate when a player knows a channel. The channels known by a player are linearly ordered, e.g., counterclockwise, starting from the lower one. Formally, as explained in Appendix A, positions are presheaves over a certain category $\mathbb{C}_1$. Morphisms of positions are natural transformations, which are roughly morphisms of graphs, mapping players to players and channels to channels. In full generality, morphisms thus do not have to be injective. However, let us restrict to injective morphisms for this expository paper. Positions and morphisms between them form a category $\mathbb{B}$.

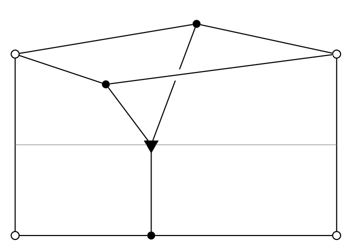
2.2 Moves

Plays are then defined as glueings of *moves* derived from the very definition of CCS, which we now sketch. Moves come in three layers:

- *basic* moves, which are used to define views below,
- *full* moves, which are used in the statement of temporal decomposition (Theorem 2),
- and *closed-world* moves, which are used to define closed-world plays (which in turn are central to the notion of interactive equivalence).

It might here be worth providing some intuition on the difference between the three notions of move. A closed-world move roughly consists of some players (one or more) synchronising together in some specified way, each of them forking into several 'avatars'. A full move gathers what concerns one of the players involved in such a synchronisation. A basic move is what one of its avatars remembers of the move.

Let us start with a closed-world move which concerns only one player, and which is hence also a full move: *forking* In the case of a player knowing two channels, the forking move is represented by the diagram *P*:

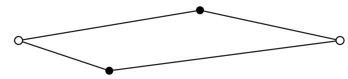


, (1)

[1]Only 'a kind of', because, as mentioned above, the channels known to a player are linearly ordered.

to be thought of as a move from the bottom position X

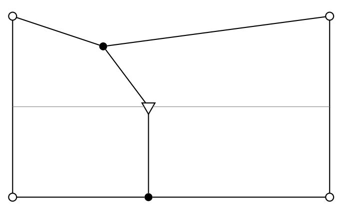
to the top position Y



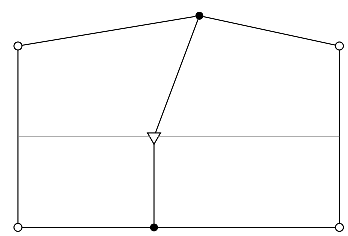
The whole move may be viewed as a cospan $X \hookrightarrow P \hookleftarrow Y$ in the category of diagrams (technically a presheaf category $\widehat{\mathbb{C}}$). Both legs of the cospan are actually monic arrows in $\widehat{\mathbb{C}}$, as will be the case for all cospans considered here. The vertical lines represent dots (channels and players) moving in time, upwards. So for example the left- and right-hand borders are just channels evolving in time, not noticing that the represented player forks into two. The surfaces spread between those vertical lines represent links (edges in the involved positions) evolving in time. For example, each link here divides into two when the player forks, thus representing the fact that both of the newly created players retains knowledge of the corresponding name. As for channels known to a player, the players and channels touching the black triangle are ordered: in particular there are different 'ports' for the initial player and its two avatars.

There is of course an instance π_n of forking for each n, according to the number of channels known to the player. Again, these explanations are very informal, but the diagrams have a very precise combinatorial definition.

The above forking move has two *basic* sub-moves, *left* and *right half-forking*, respectively represented by the diagrams



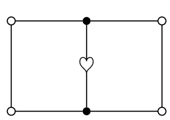
and



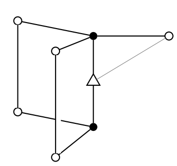
, (2)

which represent what each of the 'avatars' of the forking player sees of the move. If a play contains both of the latter moves, then it contains the full move (1). Forking, being the only move with more than one player in the final position, is the only one subject to such a decomposition. We call π_n^l and π_n^r the respective instances of the left-hand and right-hand basic moves for a player knowing n names.

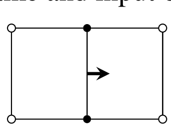
Let us now review the other basic moves of the game, which are also full. As for forking, there is an instance of each of them following the number n of channels known to the player, and we only show the case $n = 2$. First, we have the *tick* move $\heartsuit_n$, whose role is to define successful plays, and the usual *name creation*, or *restriction* ν of CCS, here ν_n. They are graphically represented as



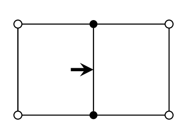
and



respectively. We finally have input and output, $a.P$ and $\overline{a}.P$ in CCS, respectively $\iota_{n,i}^{+}$ and $\iota_{n,i}^{-}$ here (n is the number of known channels, i is the index of the channel bearing the synchronisation). Here, output on the right-hand name and input on the left-hand name respectively look like



and

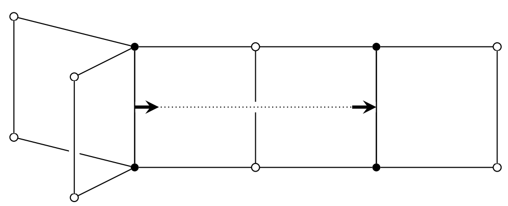


.

Basic	Full	Closed-world
Left half-forking Right half-forking	Forking	Forking
Input Output	Input Output	Synchronisation
Tick Name creation	Tick Name creation	Tick Name creation

Table 2: Summary of classes of moves

We have now defined all basic and full moves, and move on to define closed-world moves. Forking, name creation, and tick are all closed-world moves, and there is only one more closed-world move, which models CCS-like synchronisation. For all n and m, representing the numbers of channels known to the players involved in the synchronisation, and for all $i \in n$, $j \in m$ (seeing n and m as finite ordinals), there is a *synchronisation* $\tau_{n,i,m,j}$, represented, (in the case where one player outputs on channel $1 \in 2$ and the other inputs on channel $0 \in 1$,) by



Let us emphasise here that the dotted wire in the picture is actually a point in the formal representation (i.e., an element of the corresponding presheaf).

Table 2 summarises the various classes of moves, and altogether they form the set of *moves*.

2.3 Plays

We now sketch how plays are defined as glueings of moves. We start with the following example, depicted in Figure 1. The initial position consists of two players p_1 and p_2 sharing knowledge of a name a, each of them knowing another name, resp. a_1 and a_2. The play consists of four moves: first p_1 forks into $p_{1,1}$ and $p_{1,2}$, then p_2 forks, and then $p_{1,1}$ does a left half-fork into $p_{1,1,1}$; finally $p_{1,1,1}$ synchronises (as the sender) with $p_{2,1}$. Now, we reach the limits of the graphical representation, but the order in which the forks of p_1 and p_2 occur is irrelevant: if p_2 had forked before p_1, we would obtain the same play. This means that glueing the various parts of the picture in Figure 1 in different orders formally yields the same result (although there are subtle issues in representing this result graphically in a canonical way).

Now, recall that moves may be seen as cospans $X \hookrightarrow M \hookleftarrow Y$. Now, consider an *extended* notion of move, which may occur in a larger position than just one player (two for synchronisation). For example, the moves in Figure 1 are extended moves in this sense. We may now state:

Definition 1. *A* play *is an embedding $X_0 \hookrightarrow U$ in the category $\widehat{\mathbb{C}}$ of diagrams, isomorphic to a possibly denumerable 'composition' of moves in the (bi)category $\mathsf{Cospan}(\widehat{\mathbb{C}})$ of cospans in $\widehat{\mathbb{C}}$, i.e., obtained as a colimit:*

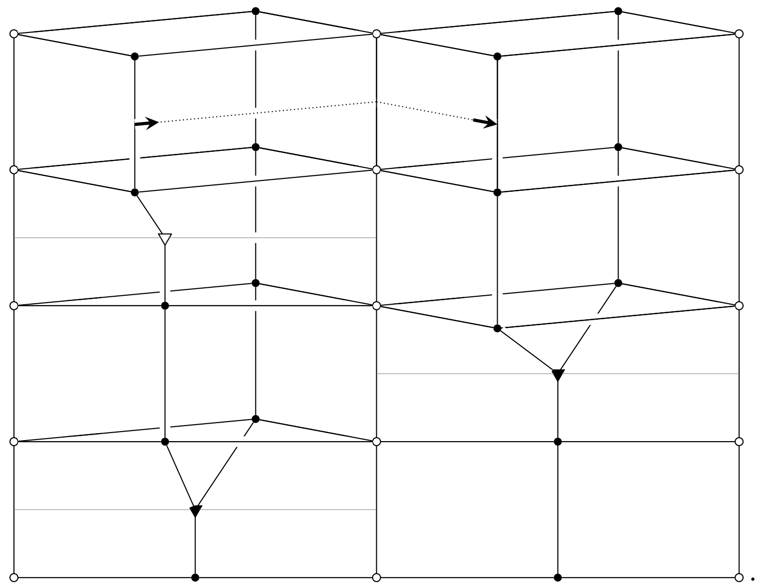
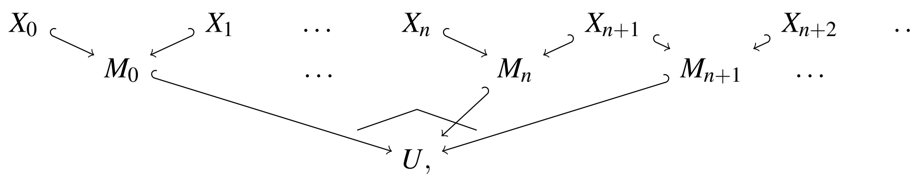


Figure 1: An example play



where each $X_i \hookrightarrow M_i \hookleftarrow X_{i+1}$ is an extended move.

Notation: we often denote plays just by U, leaving the embedding $X \hookrightarrow U$ implicit.

Remark 1. *For finite plays, one might want to keep track not only of the initial position, but also of the final position. This indeed makes sense. Finite plays then compose 'vertically', and form a double category. But infinite plays do not really have any final position, which explains our definition.*

Let a morphism $(X \hookrightarrow U) \to (Y \hookrightarrow V)$ of plays be a pair (h,k) making the diagram on the right commute in $\widehat{\mathbb{C}}$. This permits both inclusion 'in width' and 'in height'. E.g., the play consisting of the left-hand basic move in (2) embeds in exactly two ways into the play of Figure 1. (Only two because the image of the base position must lie in the base position of the codomain.) We have:

$$\begin{array}{ccc} U & \xrightarrow{k} & V \\ \uparrow & & \uparrow \\ X & \xrightarrow{h} & Y. \end{array}$$

Proposition 1. *Plays and morphisms between them form a category $\mathbb{E}$.*

There is a projection functor $\mathbb{E} \to \mathbb{B}$ mapping each play $X \hookrightarrow U$ to its base position X. This functor has a section, which is an embedding $\mathbb{B} \hookrightarrow \mathbb{E}$, mapping each position X to the empty play $X \hookrightarrow X$ on X.

Remark 2 (Size)**.** *The category $\mathbb{E}$ is only locally small. Since presheaves on a locally small category are less well-behaved than on a small category, we will actually consider a skeleton of $\mathbb{E}$. Because $\mathbb{E}$ consists only of denumerable presheaves, this skeleton is a small category. Thus, our presheaves in the next section may be understood as taken on a small category.*

Remark 3. *Plays are not very far from being just (infinite) abstract syntax trees (or forests) 'glued together along channels'.*

2.4 Relativisation

If we now want to restrict to plays over a given base position X, we may consider

Definition 2. *Let the category $\mathbb{E}_X$ have*

- *as objects pairs of a play $Y \hookrightarrow U$ and a morphism $Y \to X$,*
- *as morphisms $(Y \hookrightarrow U) \to (Y' \hookrightarrow U')$ all pairs (h,k) making the following diagram commute:*

$$\begin{array}{ccc} U & \xrightarrow{\quad k \quad} & U' \\ \uparrow & & \uparrow \\ Y & \xrightarrow{\quad h \quad} & Y' \\ & \searrow \quad X \quad \swarrow & \end{array}$$

in $\widehat{\mathbb{C}}$.

We will usually abbreviate $U \hookleftarrow Y \hookrightarrow X$ as just U when no ambiguity arises. As for morphisms of positions, in full generality, h and k, as well as the morphisms $Y \to X$, do not have to be injective. However, for the purpose of this expository paper, let us again restrict to injective h,k, and $Y \to X$.

Example 1. *Let X be the position* *. The play in Figure 1, say $Y \hookrightarrow U$, equipped with the injection $Y \hookrightarrow X$ mapping the two players of Y to the two leftmost players of X, is an object of $\mathbb{E}_X$.*

3 Innocent strategies as sheaves

Now that the category of plays is defined, we move on to defining innocent strategies. There is a notion of a Grothendieck *site* [28], which consists of a category equipped with a (generalised) topology. On such sites, one may define a category of sheaves, which are very roughly the presheaves that are continuous for the topology. We claim that there is a topology on each $\mathbb{E}_X$, for which sheaves adequately model innocent strategies. Fortunately, in our setting, sheaves admit a simple description, so that in this expository paper we can avoid the whole machinery.

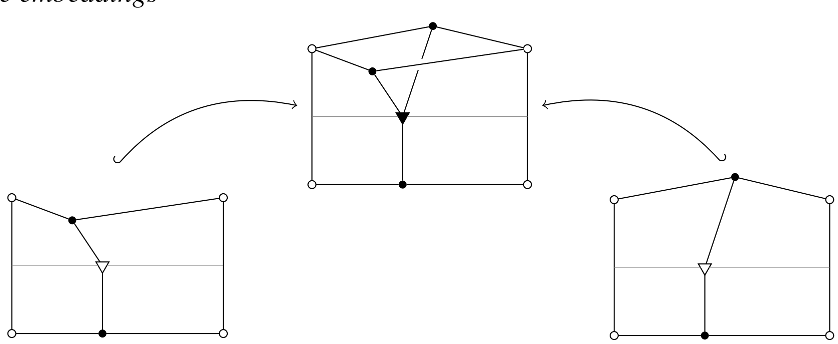
3.1 Innocent strategies

Definition 3. *A* view *is a finite 'composition' $n \hookrightarrow V$ of basic moves in* $\mathsf{Cospan}(\widehat{\mathbb{C}})$.

Example 2. *Forking* (1) *has two non-trivial views, namely the basic moves* (2)*.*

Example 3. *In Figure 1, the left-hand branch contains a view consisting of three basic moves: two π_2^l and an input.*

Example 4. *The embeddings*



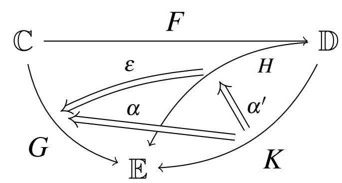
have views as domains.

For any position X, let $\mathbb{V}_X$ be the subcategory of $\mathbb{E}_X$ consisting of views.

Definition 4. *Let the category* S_X *of* innocent strategies *on* X *be the category* $\widehat{\mathbb{V}_X}$ *of presheaves on* $\mathbb{V}_X$.

A possible interpretation is that for a presheaf $F \in \widehat{\mathbb{V}_X}$ and view $V \in \mathbb{V}_X$, $F(V)$ is the set of possible *states* of the strategy F after playing V.

It might thus seem that we could content ourselves with defining only views, as opposed to full plays. However, in order to define interactive equivalences in Section 4, we need to view innocent strategies as (particular) presheaves on the whole of $\mathbb{E}_X$.



The connection is as follows. Recall from MacLane [27] the notion of *right Kan extension*. Given functors F and G as on the right, a right Kan extension $\mathrm{Ran}_F(G)$ of G along F is a functor $H \colon \mathbb{D} \to \mathbb{E}$, equipped with a natural transformation $\varepsilon \colon HF \to G$, such that for all functors $K \colon \mathbb{D} \to \mathbb{E}$ and transformations $\alpha \colon KF \to G$, there is a unique $\alpha' \colon K \to H$ such that $\alpha = \varepsilon \circ_1 (\alpha' \circ id_F)$, where $\circ_1$ is vertical composition of natural transformations. Now, precomposition with F induces a functor $\mathsf{Cat}(F,\mathbb{E}) \colon \mathsf{Cat}(\mathbb{D},\mathbb{E}) \to \mathsf{Cat}(\mathbb{C},\mathbb{E})$, where $\mathsf{Cat}(\mathbb{D},\mathbb{E})$ is the category of functors $\mathbb{D} \to \mathbb{E}$ and natural transformations between them. When $\mathbb{E}$ is complete, right Kan extensions always exist (and an explicit formula for our setting is given below), and choosing one of them for each functor $\mathbb{C} \to \mathbb{E}$ induces a right adjoint to $\mathsf{Cat}(F,\mathbb{E})$. Furthermore, it is known that when F is full and faithful, then ε is a natural isomorphism, i.e., $HF \cong G$, which entails that Ran_F is a full essential embedding.

Returning to views and plays, the embedding $i_X \colon \mathbb{V}_X \hookrightarrow \mathbb{E}_X$ is full, so right Kan extension along $i_X^{op} \colon \mathbb{V}_X^{op} \to \mathbb{E}_X^{op}$ induces a full essential embedding $\mathrm{Ran}_{i_X^{op}} \colon \widehat{\mathbb{V}_X} \to \widehat{\mathbb{E}_X}$. The (co)restriction of this essential embedding to its essential image thus yields an essentially surjective, fully faithful functor, i.e., an equivalence of categories:

Proposition 2. *The category* S_X *is equivalent to the essential image of* $\mathrm{Ran}_{i_X^{op}}$.

The standard characterisation of right Kan extensions as ends [27] yields, for any $F \in \widehat{\mathbb{V}_X}$ and $U \in \mathbb{E}_X$:

$$\mathrm{Ran}_{i_X^{op}}(F)(U) = \int_{V \in \mathbb{V}_X} F(V)^{\mathbb{E}_X(V,U)},$$

i.e., an element of $\mathrm{Ran}_{i_X^{op}}(F)$ on a play U consists, for each view V and morphism $V \to U$, of an element of $F(V)$, satisfying some compatibility conditions. In Example 5 below, we compute an example right Kan extension.

The interpretation of strategies in terms of states extends: for any presheaf $F \in \widehat{\mathbb{E}_X}$ and play $U \in \mathbb{E}_X$, $F(U)$ is the set of possible *states* of the strategy F after playing U. That F is in the image of $\mathrm{Ran}_{i_X^{op}}$ amounts to $F(U)$ being a compatible tuple of states of F after playing each view of U.

Example 5. *Here is an example of a presheaf* $F \in \widehat{\mathbb{E}_X}$ *which is not innocent, i.e., not in the image of* $\mathrm{Ran}_{i_X^{op}}$. *Consider the position* X *consisting of three players, say* x,y,z*, sharing a name, say* a. *Let* X_x *be the subposition with only* x *and* a*, and similarly for* X_y*,* X_z*,* $X_{x,y}$*, and* $X_{x,z}$. *Let* $I_x = (\iota^-_{1,0} \hookleftarrow X_x \hookrightarrow X)$ *be the play where* x *inputs on* a*, and similarly let* O_y *and* O_z *be the plays where* y *and* z *output on* a*, respectively. Let* $F(I_x) = F(O_y) = F(O_z) = 1$ *be singletons. Let now* $S_{x,y} = (\tau_{1,0,1,0} \hookleftarrow X_{x,y} \hookrightarrow X)$ *be the play where* x *and* y *synchronise on* a *(*x *inputs and* y *outputs), and similarly let* $S_{x,z}$ *be the play where* x *and* z *synchronise on* a. *Let* $F(S_{x,y}) = 2$ *be a two-element set, and let* $F(S_{x,z}) = \emptyset$. *Finally, let* F *map any subplay of the above plays to a singleton, and any strict superplay to the empty set.*

This F fails to be innocent on two counts. First, since x and y accept to input and output in only one way, it is non-innocent to accept that they synchronise in more than one way. Formally, $S_{x,y}$ has two non-trivial views, I_x and O_y, so since F maps the empty view to a singleton, $F(S_{x,y})$ should be isomorphic to $F(I_x) \times F(O_y) = 1 \times 1 = 1$. Second, since x and z accept to input and output, it is non-innocent to not accept that they synchronise. Formally, $F(S_{x,z})$ should also be a singleton. This altogether models the fact that in CCS, processes do not get to choose with which other processes they synchronise.

The restriction of F to $\mathbb{V}_X$, i.e., $F' = F \circ i_X^{op}$, in turn has a right Kan extension F'', which is innocent. (In passing, the unit of the adjunction $\mathsf{Cat}(i_X^{op}, \mathsf{Set}) \dashv \mathrm{Ran}_{i_X^{op}}$ is a natural transformation $F \to F''$.) To conclude this example, let us compute F''. First, F' only retains from F its values on views. So, if X_x denotes the empty view on X_x, $F'(X_x) = 1$, and similarly $F'(X_y) = F'(X_z) = 1$. Furthermore, $F'(I_x) = F'(O_y) = F'(O_z) = 1$. Finally, for any view V not isomorphic to any of the previous ones, $F'(V) = \emptyset$. So, recall that F'' maps any play $U \hookleftarrow Y \hookrightarrow X$ to $\int_{V \in \mathbb{V}_X} F'(V)^{\mathbb{E}_X(V,U)}$. So, e.g., since the views of $S_{x,y}$ are subviews of I_x and O_y, we have $F''(S_{x,y}) = F'(I_x) \times F'(O_y) = 1$. Similarly, $F''(S_{x,z}) = 1$. But also, for any play U such that all views $V \to U$ are subviews of either of I_x, O_y, or O_z, we have $F''(U) = 1$. Finally, for any play U such that there exists a view $V \to U$ which is not a subview of any of I_x, O_y, or O_z, we have $F''(U) = \emptyset$.

One way to understand Proposition 2 is to view $\widehat{\mathbb{V}_X}$ as the syntax for innocent strategies: presheaves on views are (almost) infinite terms in a certain syntax (see Section 3.2 below). On the other hand, seeing them as presheaves on plays will allow us to consider their global behaviour: see Section 4 when we restrict to the closed-world game. Thus, right Kan extension followed by restriction to closed-world will associate a semantics to innocent strategies.

So, we have defined for each X the category S_X of innocent strategies on X. This assignment is actually functorial $\mathbb{B}^{op} \to \mathsf{CAT}$, as follows (where CAT is the large category of locally small categories). Any morphism $f\colon Y \to X$ induces a functor $f_!\colon \mathbb{V}_Y \to \mathbb{V}_X$ sending $(V \hookleftarrow Z \to Y)$ to $(V \hookleftarrow Z \to Y \to X)$. Precomposition with $(f_!)^{op}$ thus induces a functor $\mathsf{S}_f\colon \widehat{\mathbb{V}_X} \to \widehat{\mathbb{V}_Y}$.

Proposition 3. *This defines a functor* $\mathsf{S}\colon \mathbb{B}^{op} \to \mathsf{CAT}$.

But there is more: for any position, giving a strategy for each player in it easily yields a strategy on the whole position. We call this *amalgamation* of strategies. Formally, consider any subpositions X_1 and X_2 of a given position X, inducing a partition of the players of X, i.e., such that $X_1 \cup X_2$ contains all the players of X, and $X_1 \cap X_2$ contains none. Then $\mathbb{V}_X$ is isomorphic to the coproduct $\mathbb{V}_{X_1} + \mathbb{V}_{X_2}$. (Indeed, a view contains in particular an initial player in X, which forces it to belong either in $\mathbb{V}_{X_1}$ or in $\mathbb{V}_{X_2}$.)

Definition 5. *Given strategies F_1 on X_1 and F_2 on X_2, let their* amalgamation *be their image in $\mathbb{V}_X$ via the above equivalence, i.e., the copairing*

$$[F_1, F_2] : \mathbb{V}_X^{op} \cong (\mathbb{V}_{X_1} + \mathbb{V}_{X_2})^{op} \cong \mathbb{V}_{X_1}^{op} + \mathbb{V}_{X_2}^{op} \to \mathsf{Set}.$$

By universal property of coproduct:

Proposition 4. *Amalgamation yields an equivalence of categories $\widehat{\mathbb{V}_X} \simeq \widehat{\mathbb{V}_{X_1}} \times \widehat{\mathbb{V}_{X_2}}$.*

Example 6. *Consider again the position X from Example 5, and let $X_{y,z}$ be the subposition with only y and z. We have $\mathbb{V}_X \simeq (\mathbb{V}_{X_x} + \mathbb{V}_{X_{y,z}})$, which we may explain by hand as follows. A view on X has a base player, x, y, or z, and so belongs either in $\mathbb{V}_{X_x}$ or in $\mathbb{V}_{X_{y,z}}$. Furthermore, if V is a view on x and W is a view on y, then $\mathbb{V}_X(V,W) = \emptyset$ (and similarly for any pair of distinct players in X).*

Now, recall F', the restriction of F to $\mathbb{V}_X$. We may define $F_x\colon \mathbb{V}_{X_x}^{op} \to \mathsf{Set}$ to be the restriction of F' along the (opposite of the) embedding $\mathbb{V}_{X_x} \hookrightarrow \mathbb{V}_X$, and similarly $F_{y,z}$ to be the restriction of F' along $\mathbb{V}_{X_{y,z}} \hookrightarrow \mathbb{V}_X$. Observe that F' sends any view V on x to $F_x(V)$, and similarly for views on y and z, we conclude that F' is actually the copairing $[F_x, F_{y,z}]$.

Analogous reasoning leads to the following. For any X, let $\mathrm{Pl}(X)$ denote the set of pairs (n,x), where x is a player in X, knowing n names. This yields the *spatial decomposition* theorem, where n is abusively seen as the position with one player knowing n names:

Theorem 1. *We have* $\widehat{\mathbb{V}_X} \simeq \prod_{(n,x)\in \mathrm{Pl}(X)} \widehat{\mathbb{V}_n}$.

There is actually more structure than that, namely the functor S is a stack [37], but we do not need to spell out the definition here.

3.2 Temporal decomposition and languages

Let us now describe *temporal* decomposition. The main goal here is to sketch the interpretation of CCS in innocent strategies. This material is not needed for Section 4.

Recall that full moves are forking (1), tick, name creation, input, and output.

Definition 6. *Let $\mathcal{M}_n$ be the set of all full moves starting from n. For each $M \in \mathcal{M}_n$, let* $\mathrm{cod}(M)$ *be the final position of the corresponding move.*

Strictly speaking, $\mathcal{M}_n$ is a proper class, but we may easily choose one representative of each isomorphism class. For instance, all moves are actually representable presheaves in $\widehat{\mathbb{C}}$, so we may just pick these.

To state the temporal decomposition theorem, we need a standard [20] categorical construction, the category of families on a given category $\mathbb{C}$. First, given a set X, consider the category $\mathrm{Fam}(X)$ with as objects X-indexed families of sets, i.e., sets $(Y_x)_{x\in X}$, and as morphisms $Y \to Z$ families $(f_x \colon Y_x \to Z_x)_{x\in X}$ of maps. This category is equivalently described as the slice category Set/X. To see the correspondence, consider any family $(Y_x)_{x\in X}$, and map it to the projection function $\sum_{x\in X} Y_x \to X$ sending (x,y) to x. Generalising from sets X to arbitrary categories $\mathbb{C}$, $\mathrm{Fam}(\mathbb{C})$, has as objects families $f \colon Y \to \mathrm{ob}(\mathbb{C})$ indexed by the objects of $\mathbb{C}$, and as morphisms $(Y,p) \to (Z,q)$ pairs of $u \colon Y \to Z$ and $v \colon Y \to \mathrm{mor}(\mathbb{C})$, such that $qu = p$, $\mathrm{dom}(v(x)) = p(x)$, and $\mathrm{cod}(v(x)) = q(u(x))$. Thus, any element $y \in Y$ above $C \in \mathbb{C}$ is mapped to some $u(y) \in Z$ above $C' \in \mathbb{C}$, and this mapping is labelled by a morphism $v(y) \colon C \to C'$ in $\mathbb{C}$. We may now state:

Theorem 2. *There is an equivalence of categories* $\mathsf{S}_n \simeq \mathrm{Fam}\left(\prod_{M\in\mathcal{M}_n} \mathsf{S}_{\mathrm{cod}(M)}\right)$.

The main intuition for the proof is that a strategy is determined up to isomorphism by (i) its initial states, and (ii) what remains of them after each possible full move. The family construction is what permits non-deterministic strategies: a given move may lead to different states.

Remark 4. *The theorem almost makes strategies into a* sketch *(on the category with positions as objects, finite compositions of extended moves as morphisms, and the $\mathcal{M}_X$'s as distinguished cones). Briefly, being a sketch would require a bijection of sets $\mathsf{S}_n \cong \prod_{M\in\mathcal{M}_n} \mathsf{S}_{\mathrm{cod}(M)}$. Here, the bijection becomes an equivalence of categories, and the family construction sneaks in.*

Putting the decomposition theorems together, we obtain

$$\mathsf{S}_n \simeq \mathrm{Fam}\left(\prod_{M\in\mathcal{M}_n} \prod_{(n',x')\in \mathrm{Pl}(\mathrm{cod}(M))} \mathsf{S}_{n'}\right),$$

for all n. Considering a variant of this formula as a system of equations will lead to our interpretation of CCS. The first step is to consider the formula as an endofunctor F_0 on Cat/ω, where ω is the set of finite ordinals, seen as a discrete category. This functor is defined on any family of categories $\mathbb{X} = (\mathbb{X}_i)_{i\in\omega}$ by:

$$(F_0(\mathbb{X}))_n = \mathrm{Fam}\left(\prod_{M\in\mathcal{M}_n} \prod_{(n',x')\in \mathrm{Pl}(\mathrm{cod}(M))} \mathbb{X}_{n'}\right).$$

Then, using the notation just before Theorem 2, we restrict attention to families $Y \to \mathrm{ob}(\mathbb{C})$ where Y is a finite ordinal $n \in \omega$. Simultaneously, we restrict attention to discrete categories $\mathbb{X}$, i.e., we see the formula as a endofunctor on Set/ω, i.e., ω-indexed families of sets. This yields, for any family $X = (X_i)_{i\in\omega}$,

$$(F(X))_n = \sum_{I\in\omega} \left(\prod_{M\in\mathcal{M}_n} \prod_{(n',x')\in \mathrm{Pl}(\mathrm{cod}(M))} X_{n'} \right)^I .$$

This endofunctor is polynomial [24] and we now give a characterisation of its final coalgebra. Let for any category $\mathbb{C}$ the category $\widehat{\mathbb{C}}^f$ be the functor category $\mathbb{C}^{op} \to \mathsf{FinOrd}$, where FinOrd is the category of finite ordinals and monotone functions between them. By composition with the embedding $\mathsf{FinOrd} \hookrightarrow \mathsf{Set}$, we have an embedding $\widehat{\mathbb{C}}^f \hookrightarrow \widehat{\mathbb{C}}$. We have:

Theorem 3. *The family* $\mathrm{ob}(\widehat{\mathbb{V}_n}^f)$ *formed for each n by the objects of* $\widehat{\mathbb{V}_n}^f$ *is a terminal coalgebra for* F.

By Lambek's lemma [26], there is a bijection (between the objects)

$$\widehat{\mathbb{V}_n}^f \cong \sum_{I\in\omega} \left(\prod_{M\in\mathcal{M}_n} \prod_{(n',x)\in \mathrm{Pl}(\mathrm{cod}(M))} \widehat{\mathbb{V}_{n'}}^f \right)^I . \tag{3}$$

In particular, the family $\widehat{\mathbb{V}_n}^f$ supports the operations of the grammar

$$\frac{\dots \quad n \vdash F_i \quad \dots \qquad (\forall i \in I)}{n \vdash \sum_{i\in I} F_i} \ (I \in \omega) \qquad \frac{\dots \quad n' \vdash F_{M,n',x} \quad \dots \qquad (\forall M \in \mathcal{M}_n, (n',x) \in \mathrm{Pl}(\mathrm{cod}(M)))}{n \vdash \langle (M,(n',x)) \mapsto F_{M,(n',x)} \rangle} .$$

Here, $n \vdash F$ denotes a presheaf of finite ordinals on $\mathbb{V}_n$. The interpretation is as follows: given presheaves $F_0, \dots, F_{I-1}$, for $I \in \omega$, the leftmost rule constructs the finite coproduct $\sum_{i\in I} F_i$ of presheaves (finite coproducts exist in $\widehat{\mathbb{V}_n}^f$ because they do in FinOrd). In particular, when I is the empty ordinal, we sum over an empty set, so the rule degenerates to

$$\frac{}{n \vdash \emptyset} .$$

In terms of presheaves, this is just the constantly empty presheaf.

For the second rule, if for all M, n', x, we are given $F_{M,(n',x)} \in \widehat{\mathbb{V}_{n'}}^f$, then $\langle (M,(n',x)) \mapsto F_{M,(n',x)} \rangle$ denotes the image under (3) of

$$(1, 0 \mapsto M \mapsto (n',x) \mapsto F_{M,n',x}).$$

Here, we provide an element of the right-hand side of (3), consisting of the finite ordinal $I = 1 = \{0\}$, and the function mapping (M,n',x) to $F_{M,n',x} \in \widehat{\mathbb{V}_{n'}}^f$ (up to currying). That was for parsing; the intuition is that we construct a presheaf with one initial state, 0, which maps any view starting with (M,n',x), say $M;V$, to $F_{M,n',x}(V)$. Thus the $F_{M,n',x}$'s specify what remains of our presheaf after each possible basic move. In particular, when all the $F_{M,n',x}$'s are empty, we obtain a presheaf which has an initial state, but which does nothing beyond it. We abbreviate it as $0 = \langle \_ \mapsto \emptyset \rangle$.

$$\textsc{CCSApp}\quad \frac{}{\Xi;\Gamma\vdash x(a_1,\ldots,a_n)}\ ((x\colon n)\in\Xi \text{ and } a_1,\ldots,a_n\in\Gamma) \qquad \frac{\Xi;\Gamma,a\vdash P}{\Xi;\Gamma\vdash \nu a.P}$$

$$\frac{\ldots\quad \Xi;\Gamma\vdash P_i \quad \ldots \quad (\forall i\in I)}{\Xi;\Gamma\vdash \sum_{i\in I}\alpha_i.P_i}\ (I\in\omega \text{ and } \forall i\in I, \lfloor\alpha_i\rfloor\in\Gamma) \qquad \frac{\Xi;\Gamma\vdash P \quad \Xi;\Gamma\vdash Q}{\Xi;\Gamma\vdash P|Q}$$

$$\textsc{Global}\quad \frac{\Xi;\Delta_1\vdash P_1 \quad \ldots \quad \Xi;\Delta_n\vdash P_n \quad \Xi;\Gamma\vdash P}{\Gamma\vdash \texttt{rec}\ x_1(\Delta_1) := P_1,\ldots,x_n(\Delta_n) := P_n\ \texttt{in}\ P}$$

Figure 2: CCS syntax

3.3 Translating CCS

It is rather easy to translate CCS into this language. First, define CCS syntax by the natural deduction rules in Figure 2, where Names and Vars are two fixed, disjoint, and infinite sets of *names* and *variables*; Ξ ranges over finite sequences of pairs $(x\colon n)$ of a variable x and its *arity* $n\in\omega$; Γ ranges over finite sequences of names; there are two judgements: $\Gamma\vdash P$ for *global* processes, $\Xi;\Gamma\vdash P$ for *open* processes. Rule GLOBAL is the only rule for forming global processes, and there $\Xi = (x_1\colon |\Delta_1|,\ldots,x_n\colon |\Delta_n|)$. Finally, α denotes a or $\overline{a}$, for $a\in\mathsf{Names}$, and $\lfloor a\rfloor = \lfloor\overline{a}\rfloor = a$.

First, we define the following (approximation of a) translation on open processes, mapping each open process $\Xi;\Gamma\vdash P$ to $[\![P]\!]\in\widehat{\mathbb{V}_n}^f$, for $n=|\Gamma|$. This translation ignores the recursive definitions, and we will refine it below to take them into account. We proceed by induction on P, leaving contexts $\Xi;\Gamma$ implicit:

$$\begin{array}{rl} x(a_1,\ldots,a_k) \mapsto \emptyset & \qquad \nu a.P \mapsto \langle (\nu_n, n+1, t)\mapsto [\![P]\!], (\_,\_)\mapsto\emptyset\rangle \\ P|Q \mapsto \langle\ (\pi_n,n,t_1)\mapsto [\![P]\!], & \qquad \sum_{i\in I}\alpha_i.P_i \mapsto \langle\ ((\iota^+_{n,j},n,t)\mapsto \sum_{k\in I_{\overline{j}}}[\![P_k]\!], \\ (\pi_n,n,t_2)\mapsto [\![Q]\!], & \qquad\qquad (\iota^-_{n,j},n,t)\mapsto \sum_{k\in I_j}[\![P_k]\!]\ )_{j\in n}, \\ (\_,\_)\mapsto\emptyset\ \rangle & \qquad\qquad (\_,\_)\mapsto\emptyset\ \rangle. \end{array}$$

Let us explain intuitions and notation. In the first case, we assume implicitly that $(x\colon k)\in\Xi$; the intuition is just that we approximate variables with empty strategies. Next, $P|Q$ is translated to the strategy with one initial state, which only accepts the forking move first, and then lets its avatars play $[\![P]\!]$ and $[\![Q]\!]$, respectively. In the definition, we denote by t_1 and t_2 the two players of the final position in the forking move (1). Furthermore, here and in all relevant cases, n is the number of names in Γ. Similarly, $\nu a.P$ is translated to the strategy with one initial state, accepting only the name creation move, and then playing $[\![P]\!]$. Here and in the next case, t is the player of the final position in the involved move. In the last case, the guarded sum $\sum_{i\in I}\alpha_i.P_i$ is translated to the strategy with one initial state, which

- accepts input on any channel a when $\alpha_i = a$ for some $i\in I$, and output on any channel a when $\alpha_i=\overline{a}$ for some $i\in I$;
- after an input on a, plays the sum of all $[\![P_i]\!]$'s such that $\alpha_i = a$; and after an output on a, plays the sum of all $[\![P_i]\!]$'s such that $\alpha_i=\overline{a}$.

Formally, in the definition, we let for all $j\in n$ $I_{\overline{j}} = \{i\in I \mid \alpha_i = \overline{a_j}\}$ and $I_j = \{i\in I\mid \alpha_i = a_j\}$. In particular, for the last case, when $I=\emptyset$, we obtain 0.

Thus, almost all translations of open processes have exactly one initial state, i.e., map the empty view on n to the singleton 1. The only exceptions are variable applications, which are mapped to the empty presheaf.

The translation extends to global processes as follows. Fixing a global process $\texttt{rec}\ x_1(\Delta_1) = P_1, \ldots, x_k(\Delta_k) = P_k\ \texttt{in}\ P$ typed in Γ with n names, define the sequence $(P^i)_{i\in\omega}$ of open processes (all typed in $\Xi;\Gamma$) as follows. First, $P^0 = P$. Then, let $P^{i+1} = \partial P^i$, where ∂ is the *derivation* endomap on open processes typed in any extension $\Xi;(\Gamma,\Delta)$ of $\Xi;\Gamma$, which unfolds one layer of recursive definitions. This map is defined by induction on its argument as follows:

$$\partial(x_l(a_1,\ldots,a_{k_l})) = P_l[b_j \mapsto a_j]_{1\leq j\leq k_l} \qquad \partial(\nu a.P) = \nu a.\partial P$$
$$\partial(P|Q) = \partial P|\partial Q \qquad \partial(\textstyle\sum_{i\in I}\alpha_i.P_i) = \sum_{i\in I}\alpha_i.(\partial P_i),$$

where for all $l \in \{1,\ldots,k\}$, $\Delta_l = (b_1,\ldots,b_{k_l})$, and $P[\sigma]$ denotes simultaneous, capture-avoiding substitution of names in P by σ.

By construction, the translations of these open processes form a sequence $[\![P^0]\!] \hookrightarrow [\![P^1]\!] \ldots$ of inclusions in $\widehat{\mathbb{V}_n}^f$, such that for any natural number i and view $V \in \mathbb{V}_n$ of length i (i.e., with i basic moves), $[\![P^j]\!](V)$ is fixed after $j = (k+1)i$, at worst, i.e., for all $j \geq (k+1)i$, $[\![P^j]\!](V) = [\![P^{(k+1)i}]\!](V)$. Thus, this sequence has a colimit in $\widehat{\mathbb{V}_n}^f$, the presheaf sending any view V of length i to $[\![P^{(k+1)i}]\!](V)$, which we take as the translation of the original process.

Which equivalence is induced by this mapping on CCS, especially when taking into account the interactive equivalences developed in the next section? This is the main question we will try to address in future work.

4 Interactive equivalences

4.1 Fair testing vs. must testing: the standard case

An important part of concurrency theory consists in studying *behavioural equivalences*. Since each such equivalence is supposed to define when two processes behave the same, it might seem paradoxical to consider several of them. Van Glabbeek [13] argues that each behavioural equivalence corresponds to a physical scenario for observing processes.

A distinction we wish to make here is between *fair* scenarios, and *potentially unfair* ones. An example of a fair scenario is when parallel composition of processes is thought of as modelling different physical agents, e.g., in a game with several players. Otherwise said, players are really independent. On the other hand, an example of a potentially unfair scenario is when parallelism is implemented via a scheduler.

Mainstream notions of processes, e.g., transition systems or automata, are actually unfair, as the following example shows. Consider a looping process Ω, which has a silent transition τ to itself. The process $P = (\Omega|\overline{a})$, which in parallel plays Ω and tries to synchronise on a, has an infinite trace

$$P \xrightarrow{\tau} P \xrightarrow{\tau} \ldots$$

This has consequences on so-called *testing* equivalences [6]. Let $\heartsuit$ be a fixed action.

Definition 7. *A process P is* must orthogonal *to a context C, notation $P \perp^m C$, when all maximal traces of $C[P]$ play $\heartsuit$ at some point.*

Here, maximal means either infinite or finite without extensions. Let $P^{\perp^m}$ be the set of all contexts must orthogonal to P.

Definition 8. *P and Q are* must equivalent*, notation $P \sim_m Q$, when $P^{\perp^m} = Q^{\perp^m}$.*

In transition systems, or automata, recalling P above and letting $Q = \Omega$, we have $P \sim_m Q$. This might be surprising, because the context $C = a.\heartsuit \mid \square$ intuitively should distinguish P from Q, by being orthogonal to P but not to Q. However, it is not orthogonal to P, because $C[P]$ has an infinite looping trace giving priority to Ω. This looping trace is unfair, because the synchronisation on a is never performed. Thus, one may view the equivalence $P \sim_m Q$ as taking into account potential unfairness of a hypothetical scheduler. Usually, concurrency theorists consider this too coarse, and resort to *fair* testing equivalence.

Definition 9. *A process P is* fair orthogonal *to a context C, notation $P \perp^f C$, when all finite traces of $C[P]$ extend to traces that play $\heartsuit$ at some point.*

Again, $P^{\perp^f}$ denotes the set of all contexts fair orthogonal to P.

Definition 10. *P and Q are* fair equivalent*, notation $P \sim_f Q$, when $P^{\perp^f} = Q^{\perp^f}$.*

This solves the issue, i.e., $P \not\sim_f Q$.

In summary, the mainstream setting for testing equivalences relies on traces; and the notion of maximality for traces is intrinsically unfair. This is usually rectified by resorting to fair testing equivalence over must testing equivalence. Our setting is more flexible, in the sense that maximal plays are better behaved than maximal traces. In terms of the previous section, this allows viewing the looping trace $P \xrightarrow{\tau} P \xrightarrow{\tau} \ldots$ as non-maximal. In the next sections, we define an abstract notion of interactive equivalence (still in the particular case of CCS but in our setting), instantiate it to define fair and must testing equivalence, which, as we finally show, coincide.

4.2 Interactive equivalences

Definition 11. *A play is* closed-world *when all its inputs and outputs are part of a synchronisation.*

Let $\mathbb{W} \hookrightarrow \mathbb{E}$ be the full subcategory of closed-world plays, $\mathbb{W}(X)$ being the *fibre* over X for the projection functor $\mathbb{W} \to \mathbb{B}$, i.e., the subcategory of $\mathbb{W}$ consisting of closed-world plays with base X, and morphisms (id_X, k) between them[2].

Let the category of *closed-world behaviours* on X be the category $\mathsf{G}_X = \widehat{\mathbb{W}(X)}$ of presheaves on $\mathbb{W}(X)$. We may now put:

Definition 12. *An* observable criterion *consists for all positions X, of a replete subcategory $\perp\!\!\!\perp_X \hookrightarrow \mathsf{G}_X$.*

Recall that $\perp\!\!\!\perp_X$ being replete means that for all $F \in \perp\!\!\!\perp_X$ and isomorphism $f\colon F \to F'$ in G_X, F' and f are in $\perp\!\!\!\perp_X$.

An observable criterion specifies the class of ‘successful’, closed-world behaviours. The two criteria considered below are two ways of formalising the idea that a successful behaviour is one in which all accepted closed-world plays are ‘successful’, in the sense that some player plays the tick move at some point.

We now define interactive equivalences. Recall that $[F, G]$ denotes the amalgamation of F and G, and that right Kan extension along i_Z^{op} induces a functor $\mathrm{Ran}_{i_Z^{op}}\colon \widehat{\mathbb{V}_Z} \to \widehat{\mathbb{E}_Z}$. Furthermore, precomposition

[2]This is not exactly equivalent to what could be noted $\mathbb{W}_X$, since in the latter there are objects $U \hookleftarrow Y \hookrightarrow X$ with a strict inclusion $Y \hookrightarrow X$. However, both should be equivalent for what we do in this paper, i.e., fair and must equivalences.

with the canonical inclusion $j_Z \colon \mathbb{W}(Z) \hookrightarrow \mathbb{E}_Z$ induces a functor $j_Z^* \colon \widehat{\mathbb{E}_Z} \to \widehat{\mathbb{W}(Z)}$. Composing the two, we obtain a functor $\mathsf{Gl} \colon \mathsf{S}_Z \to \mathsf{G}_Z$:

$$\mathsf{S}_Z = \widehat{\mathbb{V}_Z} \xrightarrow{\mathrm{Ran}_{i_Z^{op}}} \widehat{\mathbb{E}_Z} \xrightarrow{j_Z^*} \widehat{\mathbb{W}(Z)} = \mathsf{G}_Z.$$

Definition 13. *For any strategy F on X and any pushout square P of positions as on the right, with I consisting only of channels, let $F^{\perp\!\!\!\perp_P}$ be the class of all strategies G on Y such that $\mathsf{Gl}([F,G]) \in \perp\!\!\!\perp_Z$.*

$$\begin{array}{ccc} I & \longrightarrow & Y \\ \downarrow & & \downarrow \\ X & \longrightarrow & Z \end{array} \qquad (4)$$

Here, G is thought of as a *test* for F. Also, P denotes the whole pushout square and $F^{\perp\!\!\!\perp_P}$ is notation for a notion indexed by such squares, whose definition uses $\perp\!\!\!\perp_Z \hookrightarrow \mathsf{G}_Z$. From the CCS point of view, I corresponds to the set of names shared by the process under observation (F) and the testing context (G).

Definition 14. *Any two strategies $F, F' \in \mathsf{S}_X$ are* $\perp\!\!\!\perp$-equivalent*, notation $F \sim_{\perp\!\!\!\perp} F'$, iff for all pushouts P as in 4, $F^{\perp\!\!\!\perp_P} = F'^{\perp\!\!\!\perp_P}$.*

4.3 Fair vs. must

Let us now define fair and must testing equivalences. Let a closed-world play be *successful* when it contains a $\heartsuit_n$. Furthermore, for any closed-world behaviour $G \in \mathsf{G}_X$ and closed-world play $U \in \mathbb{W}(X)$, an *extension* of a state $\sigma \in G(U)$ to U' is a $\sigma' \in G(U')$ with $i \colon U \to U'$ and $G(i)(\sigma') = \sigma$. The extension σ' is *successful* when U' is. The intuition is that the behaviour G, before reaching U' with state σ', passed through U with state σ.

Definition 15. *The* fair *criterion $\perp\!\!\!\perp^f$ contains all closed-world behaviours G such that any state $\sigma \in G(U)$ for finite U admits a successful extension.*

Now call an extension of $\sigma \in G(U)$ *strict* when $U \to U'$ is not surjective, or, equivalently, when U' contains more moves than U. For any closed-world behaviour $G \in \mathsf{G}_X$, a state $\sigma \in G(U)$ is *G-maximal* when it has no strict extension.

Definition 16. *Let the* must *criterion $\perp\!\!\!\perp^m$ consist of all closed-world behaviours G such that for all closed-world U and G-maximal $\sigma \in G(U)$, U is successful.*

We now show that fair and must testing equivalence coincide. The key result for this is:

Theorem 4. *For any strategy F on X, any state $\sigma \in \mathsf{Gl}(F)(U)$ with finite U admits a $\mathsf{Gl}(F)$-maximal extension.*

The proof basically amounts to implementing a scheduler in our framework — a fair one, of course. Thanks to the theorem, we have:

Lemma 1. *For all $F \in \mathsf{S}_X$, $\mathsf{Gl}(F) \in \perp\!\!\!\perp_X^m$ iff $\mathsf{Gl}(F) \in \perp\!\!\!\perp_X^f$.*

Proof. Let $G = \mathsf{Gl}(F)$.

($\Rightarrow$) By Theorem 4, any state $\sigma \in G(U)$ has a G-maximal extension $\sigma' \in G(U')$, which is successful by hypothesis, hence σ has a successful extension.

($\Leftarrow$) Any G-maximal $\sigma \in G(U)$ admits by hypothesis a successful extension which may only be on U by G-maximality, and hence U is successful. □

Now comes the expected result:

Theorem 5. *For all $F, F' \in \mathsf{S}_X$, $F \sim_{\perp\!\!\!\perp^m} F'$ iff $F \sim_{\perp\!\!\!\perp^f} F'$.*

Proof. *(⇒)* Consider two strategies F and F' on X, and a strategy G on Y (as in the pushout (4)). We have, using Lemma 1:

$$\mathsf{Gl}(F \parallel G) \in \perp\!\!\!\perp^{f} \quad \text{iff} \quad \mathsf{Gl}(F \parallel G) \in \perp\!\!\!\perp^{m} \quad \text{iff} \quad \mathsf{Gl}(F' \parallel G) \in \perp\!\!\!\perp^{m} \quad \text{iff} \quad \mathsf{Gl}(F' \parallel G) \in \perp\!\!\!\perp^{f}.$$

(⇐) Symmetric. □

To explain what is going on here, let us consider again $P = (\Omega|\overline{a})$, $Q = \Omega$, and the context $C = a.\heartsuit \mid \square$. We implement C by choosing as a test the strategy $T = [\![a.\heartsuit]\!]$ on a single player knowing one name a. Taking I to consist of the sole name a, the pushout Z as in Definition 13 consists of two players, say x for the observed strategy and y for the test strategy, sharing the name a. Now, assuming that Ω loops deterministically, the global behaviour $G = \mathsf{Gl}([[\![P]\!], T])$ has exactly one state on the empty play, and again exactly one state on the play π_1 consisting of only one fork move by x. Thus, G reaches a position with three players, say x_1 playing Ω, x_2 playing $\overline{a}$, and y playing $a.\heartsuit$. What makes the theorem work is that the play with ω silent moves by x_1 is not maximal. Indeed, we could insert (anywhere in the sequence of moves by x_1) a synchronisation move by x_2 and y, and then a tick move by the avatar of y. Essentially: our notion of play is more fair than just traces.

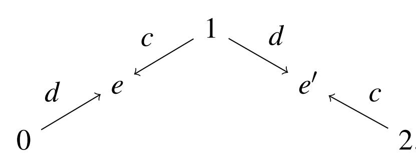
A Diagrams

In this section, we define the category on which our diagrams are presheaves. The techniques used here date back at least to Carboni and Johnstone [4, 5]. Let us first consider two baby examples. It is well-known that directed multigraphs form a presheaf category: consider the category $\mathbb{C}$ freely generated by the graph with two vertices, say $\star$ and $[1]$, and two edges $d, c \colon \star \to [1]$ between them. One way to visualise this is to compute the *category of elements* of a few presheaves on $\mathbb{C}$. Recall that the category of elements of a presheaf F on $\mathbb{C}$ is the comma category $y \downarrow_{\widehat{\mathbb{C}}} F$, where y is the Yoneda embedding. Via Yoneda, it has as elements pairs (C, x) with $C \in \mathrm{ob}(\mathbb{C})$ and $x \in F(C)$, and morphisms $(C, x) \to (D, y)$ morphisms $f \colon C \to D$ in $\mathbb{C}$ such that $F(f)(y) = x$ (which we abbreviate as $y \cdot f = x$ when the context is clear).

Example 7. *Consider the presheaf F defined by the following equations, whose category of elements is actually freely generated by the graph on the right:*

- $F(\star) = \{0, 1, 2\}$,
- $F([1]) = \{e, e'\}$,

- $e \cdot d = 0$,
- $e \cdot c = 1$,
- $e' \cdot d = 1$,
- $e' \cdot c = 2$,



This graph is of course not exactly the expected one, but it does represent it. Indeed, for each vertex we know whether it is in $F(\star)$ or $F([1])$, hence whether it represents a 'vertex' or an 'edge'. The arrows all go from a 'vertex' v to an 'edge' e. They are in $F(d)$ when v is the domain of e, and in $F(c)$ when v is the codomain of e.

Multigraphs may also be seen as a presheaves on the category freely generated by the graph with

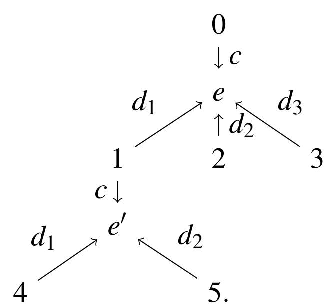
- as vertices: one special vertex $\star$, plus for each natural number n a vertex, say, $[n]$; and
- $n+1$ edges $\star \to [n]$, say $d_1, \ldots, d_n$, and c.

It should be natural for presheaves on this category to look like multigraphs: the elements of a presheaf F above $\star$ are the vertices in the multigraph, the elements above $[n]$ are the n-ary multiedges, and the action of the d_i's give the ith source of a multiedge, while the action of c gives its target.

Example 8. *Similarly, computing a few categories of elements might help visualising. As above, consider F defined by*

- $F(\star) = \{0,1,2,3,4\}$,
- $F([1]) = F([0]) = \emptyset$,
- $F([2]) = \{e'\}$,
- $F([3]) = \{e\}$,
- $F([n+4]) = \emptyset$,
- $e \cdot c = 0$,
- $e \cdot d_1 = 1$,
- $e \cdot d_2 = 2$,
- $e \cdot d_3 = 3$,
- $e' \cdot c = 1$,
- $e' \cdot d_1 = 4$,
- $e' \cdot d_2 = 5$,

whose category of elements is freely generated by the graph:



Now, this pattern may be extended to higher dimensions. Consider for example extending the previous base graph with a vertex $[m_1,\ldots,m_n;p]$ for all natural numbers $n,p,m_1,\ldots,m_n$, plus edges $s_1\colon [m_1] \to [m_1,\ldots,m_n;p],\ldots,s_n\colon [m_n] \to [m_1,\ldots,m_n;p]$, and $t\colon [p] \to [m_1,\ldots,m_n;p]$. Let now $\mathbb{C}$ be the free category on this extended graph. Presheaves on $\mathbb{C}$ are a kind of 2-multigraphs: they have vertices, multiedges, and multiedges between multiedges.

We could continue this in higher dimensions.

Defining the base category of the paper follows a very similar pattern. We start from a slightly different graph: let $\mathbb{G}_0$ have just one vertex $\star$; let $\mathbb{G}_1$, have one vertex $\star$, plus a vertex $[n]$ for each natural number n, plus n edges $d_1,\ldots,d_n\colon \star \to [n]$. Let $\mathbb{C}_0$ and $\mathbb{C}_1$ be the categories freely generated by $\mathbb{G}_0$ and $\mathbb{G}_1$, respectively. So, presheaves on $\mathbb{C}_1$ are a kind of hypergraphs with arity (since vertices incident to a hyperedge are numbered). This is enough to model positions.

Now, consider the graph $\mathbb{G}_2$, which is $\mathbb{G}_1$ augmented with:

- for all n, vertices $\heartsuit_n$, π^l_n, π^r_n, ν_n,
- for all n and $0 \le i < n$, vertices $\iota^+_{n,i}$ and $\iota^-_{n,i}$,
- for all n, edges $s,t\colon [n] \to \heartsuit_n$, $s,t\colon [n] \to \pi^l_n$, $s,t\colon [n] \to \pi^r_n$, $s\colon [n] \to \nu_n$, $t\colon [n+1] \to \nu_n$,
- for all n and $0 \le i < n$, edges $s,t\colon [n] \to \iota^+_{n,i}$, $s,t\colon [n] \to \iota^-_{n,i}$.

Note that only name creation changes the number of names known to the player, and accordingly the corresponding morphism t has domain $[n+1]$. We slightly abuse language here by calling all these t's and s's the same. We could label them with their codomain, but we refrain doing so for the sake of readability.

Now, let $\mathbb{C}_2$ be the category generated by $\mathbb{G}_2$ and the relations $s \circ d_i = t \circ d_i$ for all n and $0 \leq i < n$ (for all possible — common — codomains with main index n for s and t). Presheaves on $\mathbb{C}_2$ are enough to model views, but since we want more, we continue, as follows.

Let $\mathbb{G}_3$ be $\mathbb{G}_2$, augmented with:

- for all n, a vertex π_n, and
- edges $l\colon \pi_n^l \to \pi_n$ and $r\colon \pi_n^r \to \pi_n$.

Let $\mathbb{C}_3$ be the category generated by $\mathbb{G}_3$ and the relations $l \circ s = r \circ s$ (this models the fact that a forking move should be played by just one player). Presheaves on $\mathbb{C}_3$ are enough to model full moves; to model closed-world moves, and in particular synchronisation, we continue as follows.

Let $\mathbb{G}_4$ be $\mathbb{G}_3$, augmented with, for all n, m, $0 \leq i < n$, and $0 \leq j < m$,

- a vertex $\tau_{n,i,m,j}$, and
- edges $\varepsilon\colon \iota^+_{n,i} \to \tau_{n,i,m,j}$ and $\rho\colon \iota^-_{m,j} \to \tau_{n,i,m,j}$.

Let $\mathbb{C}_4$ be the category generated by $\mathbb{G}_4$ and the relations $\varepsilon \circ s \circ d_i = \rho \circ s \circ d_j$ (which models the fact that a synchronisation involves an input and an output on the same name).

Acknowledgements Thanks to the courageous having endured the first versions of this work. Special thanks to Paul-André Melliès for his graphical design skills, and to pseudonymous referee Michel Houellebecq, not only for our very useful and enjoyable discussion, but also for tolerating our rather poor litterary style. Finally, thanks to Mark Weber for teaching the first author the techniques of Appendix A.

References

[1] Emmanuel Beffara (2005): *Logique, réalisabilité et concurrence*. Ph.D. thesis, Université Paris 7.

[2] Marcello M. Bonsangue, Jan J. M. M. Rutten & Alexandra Silva (2009): *A Kleene Theorem for Polynomial Coalgebras*. In Luca de Alfaro, editor: *FOSSACS*, *Lecture Notes in Computer Science* 5504, Springer, pp. 122–136, doi:10.1007/978-3-642-00596-1_10.

[3] Albert Burroni (1993): *Higher-Dimensional Word Problems with Applications to Equational Logic*. *Theoretical Computer Science* 115(1), pp. 43–62, doi:10.1016/0304-3975(93)90054-W.

[4] Aurelio Carboni & Peter Johnstone (1995): *Connected Limits, Familial Representability and Artin Glueing*. *Mathematical Structures in Computer Science* 5(4), pp. 441–459, doi:10.1017/S0960129500001183.

[5] Aurelio Carboni & Peter Johnstone (2004): *Corrigenda for 'Connected limits, familial representability and Artin glueing'*. *Mathematical Structures in Computer Science* 14(1), pp. 185–187, doi:10.1017/S0960129503004080.

[6] Rocco De Nicola & Matthew Hennessy (1984): *Testing Equivalences for Processes*. *Theor. Comput. Sci.* 34, pp. 83–133, doi:10.1016/0304-3975(84)90113-0.

[7] Olivier Delande & Dale Miller (2008): *A Neutral Approach to Proof and Refutation in MALL*. In: *Proc. LICS*, pp. 498–508, doi:10.1109/LICS.2008.35.

[8] H. Ehrig, H.-J. Kreowski, Ugo Montanari & Grzegorz Rozenberg, editors (1999): *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 3: Concurrency, Parallelism and Distribution*. World Scientific.

[9] Marcelo P. Fiore (2008): *Second-Order and Dependently-Sorted Abstract Syntax*. In: *Proc. LICS*, pp. 57–68, doi:10.1109/LICS.2008.38.

[10] Fabio Gadducci, Reiko Heckel & Mercè Llabrés (1999): *A Bi-Categorical Axiomatisation of Concurrent Graph Rewriting*. *Electronic Notes in Theoretical Computer Science* 29, doi:10.1016/S1571-0661(05)80309-3.

[11] Fabio Gadducci & Ugo Montanari (2000): *The tile model*. In Gordon D. Plotkin, Colin Stirling & Mads Tofte, editors: *Proof, Language, and Interaction*, The MIT Press, pp. 133–166.

[12] Jean-Yves Girard (2001): *Locus Solum: From the rules of logic to the logic of rules*. *Mathematical Structures in Computer Science* 11(3), pp. 301–506, doi:10.1007/3-540-44802-0_3.

[13] Rob J. van Glabbeek (1990): *The Linear Time-Branching Time Spectrum (Extended Abstract)*. In Jos C. M. Baeten & Jan Willem Klop, editors: *CONCUR*, *Lecture Notes in Computer Science* 458, Springer, pp. 278–297, doi:10.1007/BFb0039066.

[14] Yves Guiraud & Philippe Malbos (2009): *Higher-dimensional categories with finite derivation type*. *Theory and Applications of Categories* 22(18), pp. 420–278.

[15] André Hirschowitz, Michel Hirschowitz & Tom Hirschowitz (2009): *Contraction-free Proofs and Finitary Games for Linear Logic*. *Electronic Notes in Theoretical Computer Science* 249, pp. 287–305, doi:10.1016/j.entcs.2009.07.095.

[16] André Hirschowitz & Marco Maggesi (2007): *Modules over Monads and Linearity*. In Daniel Leivant & Ruy J. G. B. de Queiroz, editors: *WoLLIC*, *Lecture Notes in Computer Science* 4576, Springer, pp. 218–237, doi:10.1007/3-540-44802-0_3.

[17] André Hirschowitz & Marco Maggesi (2010): *Modules over monads and initial semantics*. *Information and Computation* 208(5), pp. 545–564, doi:10.1016/j.ic.2009.07.003.

[18] Tom Hirschowitz: *Cartesian closed 2-categories and permutation equivalence in higher-order rewriting*. Available at `http://hal.archives-ouvertes.fr/hal-00540205/en/`.

[19] Martin Hyland (1997): *Semantics and Logics of Computation*, chapter Game Semantics. Cambridge University Press, doi:10.1017/CBO9780511526619.005.

[20] Bart Jacobs (1999): *Categorical Logic and Type Theory*. *Studies in Logic and the Foundations of Mathematics* 141, North Holland, Amsterdam.

[21] Ole H. Jensen & Robin Milner (2004): *Bigraphs and mobile processes (revised)*. Technical Report TR580, University of Cambridge. Available at `http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-580.pdf`.

[22] André Joyal, Mogens Nielsen & Glynn Winskel (1993): *Bisimulation and open maps*. In: *LICS*, IEEE Computer Society, pp. 418–427, doi:10.1109/LICS.1993.287566.

[23] Stefano Kasangian & Anna Labella (1999): *Observational trees as models for concurrency*. *Mathematical Structures in Computer Science* 9(6), pp. 687–718, doi:10.1017/S0960129599002935.

[24] Joachim Kock (2011): *Polynomial Functors and Trees*. *International Mathematics Research Notices* 2011(3), pp. 609–673, doi:10.1093/imrn/rnq068.

[25] Jean-Louis Krivine (2003): *Dependent choice, 'quote' and the clock*. *Theor. Comput. Sci.* 308(1-3), pp. 259–276, doi:10.1016/S0304-3975(02)00776-4.

[26] Joachim Lambek (1968): *A fixpoint theorem for complete categories*. *Mathematische Zeitschrift* 103, pp. 151–161, doi:10.1007/BF01110627.

[27] Saunders Mac Lane (1998): *Categories for the Working Mathematician*, 2nd edition. *Graduate Texts in Mathematics* 5, Springer.

[28] Saunders MacLane & Ieke Moerdijk (1992): *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Universitext, Springer.

[29] Paul-André Melliès (2004): *Asynchronous games 2: the true concurrency of innocence*. In: *Proc. CONCUR '04*, *LNCS* 3170, Springer Verlag, pp. 448–465, doi:10.1016/j.tcs.2006.01.016.

[30] Robin Milner (1980): *A Calculus of Communicating Systems*. *LNCS* 92, Springer, doi:10.1007/3-540-10235-3.

[31] Tobias Nipkow (1991): *Higher-Order Critical Pairs*. In: *LICS*, IEEE Computer Society, pp. 342–349, doi:10.1109/LICS.1991.151658.

[32] Gordon D. Plotkin (1981): *A structural approach to operational semantics*. DAIMI Report FN-19, Computer Science Department, Aarhus University.

[33] Julian Rathke & Pawel Sobocinski (2008): *Deconstructing behavioural theories of mobility*. In: *IFIP TCS*, *IFIP* 273, Springer, pp. 507–520, doi:10.1007/978-0-387-09680-3_34.

[34] Vladimiro Sassone & Pawel Sobociński (2003): *Deriving Bisimulation Congruences Using 2-categories*. *Nordic Journal of Computing* 10(2).

[35] Peter Sewell (1998): *From Rewrite to Bisimulation Congruences*. In Davide Sangiorgi & Robert de Simone, editors: *CONCUR*, *Lecture Notes in Computer Science* 1466, Springer, pp. 269–284, doi:10.1007/BFb0055611.

[36] Daniele Turi & Gordon D. Plotkin (1997): *Towards a Mathematical Operational Semantics*. In: *LICS*, pp. 280–291, doi:10.1109/LICS.1997.614955.

[37] Angelo Vistoli (2007): *Notes on Grothendieck topologies, fibered categories and descent theory*. Available at `http://arxiv.org/abs/math/0412512`.