Playgrounds, Factorization Systems, and Graphs

Clovis Eberhart
under the supervision of Tom Hirschowitz

September 9, 2014

1 Introduction

Playgrounds [5, 4, 2] are a formal tool that Tom Hirschowitz has developed to bridge the gap between concurrent languages and game semantics. Game semantics was proved to be very successful to describe the behavior of functional languages as the interaction, in some sort of game, between a function and its environment. What "behavior" means in a functional language is fairly straightforward: a functional program "behaves" like a mathematical function. However, it is much less clear what a behavior is in the world of concurrent languages. Indeed, a behavior is an interaction between the different processes, which is much harder to understand from a mathematical point of view. Playgrounds are one of the multiple attempts at using game semantics to give a mathematical meaning to the interaction between processes.

We assume the reader to be familiar with basic notions of category theory, such as morphisms, functors, natural transformations, limits and colimits [7]. We will first give an extensive introduction to playgrounds and a categorical tool that has proved to be very useful in our work: factorization systems. Then, in a first part, we will show several advances or changes in the general approach to building a playground from its basic blocks. Finally, we will discuss a double factorization system on the category of graphs as a very simple example of playgrounds, which has lead us to a better understanding of double factorization systems for playgrounds, which seem to be the right tool to understand some their properties and prove some of their axioms.

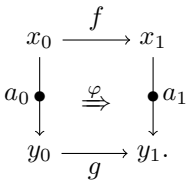
1.1 Playgrounds

The formalism of *playgrounds* is born from an attempt at reconciling concurrent calculi and game semantics. Basically, they can be seen as a game semantics that is naturally concurrent. It is a game semantics in the sense that there are players that behave according to their strategies, and that there is a notion of innocence that corresponds to that of innocence in classic game semantics (i.e., that a player's strategy can only be based on what that player observed). It is naturally concurrent in the sense that multiplayer games are defined as naturally as two-player games (while in classic game semantics, only two- and three-player games are studied, and the latter only for composition of plays), and that natural concurrent operators – such as parallel composition – exist by construction.

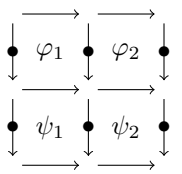
Playgrounds are simply pseudo double categories which satisfy some axioms that ensure that they yield a semantics. Pseudo double categories can be seen as a structure that has: objects, vertical morphisms, horizontal morphisms, and 2-cells, which satisfy that:

- objects and horizontal morphisms form a category, and so do vertical morphisms and 2-cells;
- objects and vertical morphisms form a category with weak composition (i.e., composition is associative only up to isomorphism), and so do horizontal morphisms and 2-cells;
- 2-cells verify the interchange law.

Basically, pseudo double categories are categories with two types of morphisms: horizontal (denoted $\longrightarrow$) and vertical (denoted $\multimap$) and "squares" (or 2-cells, denoted $\Longrightarrow$):



2-cells can be composed horizontally and vertically (horizontal composition of 2-cells is denoted $\circ$, while the vertical one is denoted $\bullet$). The interchange law simply states that the two ways of evaluating



yield the same result $((\psi_2 \circ \psi_1) \bullet (\varphi_2 \circ \varphi_1) = (\psi_2 \bullet \varphi_2) \circ (\psi_1 \bullet \varphi_1))$.

In our case, the objects of the pseudo double category represent "positions", vertical morphisms represent "plays" (i.e., the interaction of players in a position that leads to another position), and horizontal morphisms essentially represent "spatial inclusion" (i.e., the inclusion of a position in another position) – however, 2-cells do not have such a simple interpretation.

The question we are mainly interested in is how to build such a pseudo double category, i.e., how to build a playground. In order to do that, we will first introduce a base category $\mathcal{C}$ that describes the terms of the language we are studying. Then, we will take presheaves $\widehat{\mathcal{C}}$ over $\mathcal{C}$: some of these presheaves will describe positions (and which will be the objects of our playground), some will (essentially) describe moves or plays, while others will hold no meaning. These presheaves can simply be thought of as higher-dimensional graph-like objects. Then, we will introduce some cospans over $\widehat{\mathcal{C}}$ that will describe the reductions of the terms of the language, and which we will call "seeds". Based on these seeds, we will select a subcategory of the category of cospans over $\widehat{\mathcal{C}}$, which we will call plays and will be the vertical morphisms of our playground (since composition

of cospans is associative only up to isomorphism, it will also be the case for our vertical morphisms). Horizontal morphisms will (essentially) be the injective presheaf morphisms, and 2-cells will be morphisms of cospans.

Now, we need to introduce some technical terms related to playgrounds and illustrate them using the base category for the join-calculus (formally described in A). The base category $\mathcal{C}$ comes equipped with a functor $F : \mathcal{C} \to \omega$, i.e., each object c of $\mathcal{C}$ has a dimension $F(c)$. *Channels* are objects of dimension 0 (or an element of a presheaf over $\mathcal{C}$ associated to an object of dimension 0), and will be graphically denoted $\circ$ (there will be at most a single type of channel in our examples). *Players* are objects of dimension 1, and will be graphically denoted $\bullet$. *Moves* are objects of dimension greater than 1 (there may be several graphical notations for them). In the base category for the join-calculus, the only channel is $*$, the players are $[n]$ and $\langle n \rangle$, and the moves are all the other objects. A *position* is a presheaf that is empty in dimensions greater than one (i.e., it contains only channels and players).

Let us give a few simple illustrated examples:

represents a player who can communicate on two channels. Formally, it is represented by the presheaf $[2]$ (we make a slight abuse of notation here, as we collapse a presheaf and its representing object in $\mathcal{C}$). Note that this is indeed a description of $[2]$, as it actually is its category of elements:

$$(*, s_1) \xleftarrow{s_1} ([2], \mathrm{id}_{[2]}) \xrightarrow{s_2} (*, s_2).$$

Similarly,

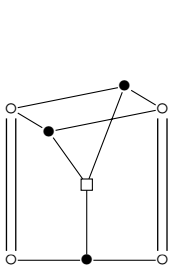
represents two players (one who can communicate on two channels, and the other who can communicate on only one channel) sharing a channel. It is formally represented by a presheaf whose category of elements is:

$$(*, c_1) \xleftarrow{s_1} ([2], p_1) \xrightarrow{s_2} (*, c_2) \xleftarrow{s_1} ([1], p_2).$$

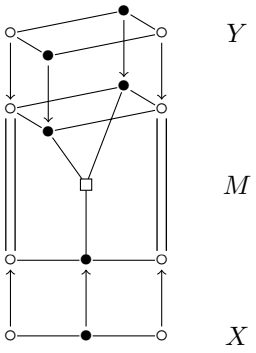
Seeds are the basic components of moves. There is a one-to-one correspondence between seeds and move object (we use the term "move object" to distinguish between the objects called "moves" in the base category $\mathcal{C}$ and what we simply call "moves", which are the basic components of plays, the vertical morphisms in our playground) in the base category $\mathcal{C}$, and each seed represents the local modification of a position after a given move, i.e., seeds are "local moves". Therefore, one could think that seeds are described by an initial position and a final position. However, we are not simply interested in whether the game can evolve from a position to another, but also in *how* the game can evolve from that position to another, i.e., which moves can change the position into another position. Therefore, seeds cannot be described simply by two positions. Seeds are cospans $Y \xrightarrow{s} M \xleftarrow{t} X$ in the category $\widehat{\mathcal{C}}$ of presheaves over $\mathcal{C}$. X and Y are positions, more precisely, X represents the initial position of the local move, while Y represents its final position. M represents *how* the local moves goes

from X to Y, and it is not a position. M is the representable presheaf of the move object corresponding to the seed $Y \xrightarrow{s} M \xleftarrow{t} X$.

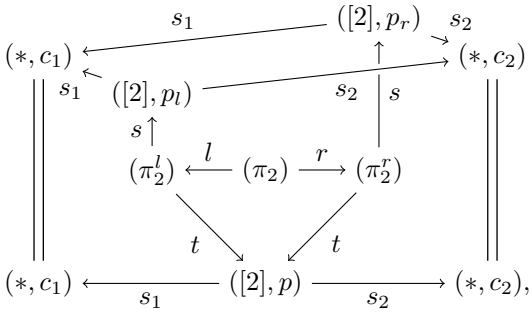
Once again, let us illustrate seeds using the join-calculus. The following seed corresponds to π_2:



which could be drawn as

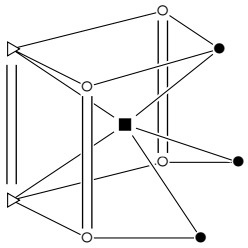


The initial position X is drawn at the bottom, and the final position is drawn at the top. In all the drawings, time will flow upwards. Here, the initial position consists of a binary player (a player who knows two channels), and the final position two binary players who share both their channels (the sign between the channels are = signs that show which channels are the same in the initial and final positions). If we think of players as equipped with strategies that are (infinite) terms of the join-calculus, then π_n is a move where a player of the form $P \,|\, Q$ (that can communicate on n channels) forks into two players P and Q, who know exactly the same channels as $P \,|\, Q$ did (and therefore share all their channels), which is exactly what is expressed in this seed. Note that this seed is, again, the category of elements of the representable presheaf for π_2:



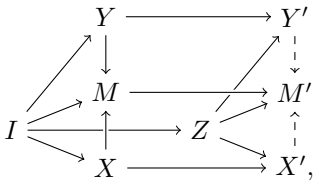
where the square stands for the three moves in the middle of the category of elements. The move that is characteristic of the join-calculus is its reduction: if $D = x\langle u\rangle \,|\, y\langle v\rangle \rhd R$ is a definition (a sort of process that exists in the join-calculus), then def D in $x\langle s\rangle \,|\, y\langle t\rangle \longrightarrow$ def D in $R\{x \leftarrow s, y \leftarrow t\}$. In that move, there is a definition (D) and two processes ($x\langle s\rangle$ and $y\langle t\rangle$). The definition can communicate on at least two channels (x and y), while each process can communicate on one of these channels. Both processes "die" when they send the name of the channels s and t on x and y (they do not appear, even transformed, in the final expression). When D receives a communication on these two channels, it spawns a new process $R\{x \leftarrow s, y \leftarrow t\}$ that can communicate on all of the channels D knows, plus s and t (it is clear this process is spawn by D and not

one of the other processes because it contains R, which is information only D knows). Here, for the sake of simplicity, we assumed that $x = s$ and $y = t$, that D only knows two channels, and the processes only one. It corresponds, in the base category for the join-calculus, to $\tau_{2,1,1\langle 1\rangle,1\langle 1\rangle}$. It is modeled as:



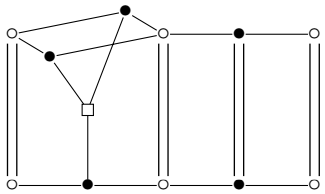
It also corresponds to $\tau_{2,1,1\langle 1\rangle,1\langle 1\rangle}$'s category of elements, but it is a bit convoluted, and will probably not be enlightening to the reader, so we will not draw it.

Each seed comes with a *canonical interface* I, which tells how this local move can be embedded into a bigger position to make a global move. A canonical interface is an example of interface, which are presheaves of dimension 0, i.e., they only contain channels. If $Y \xrightarrow{s} M \xleftarrow{t} X$ is a seed, its canonical interface must have morphisms to X and Y that are equal when composed with s and t. At the start of this work, canonical interfaces were defined as the set of all channels in the initial position of the seed. We will explain later why this definition, allied with that of move, is not a satisfactory definition and had to be changed. Now, a *global move*, is simply a pushout of the seed along a morphism $I \to Z$, where Z is a position:



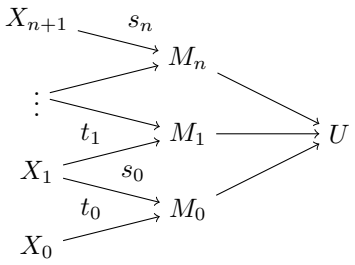
i.e., X', Y', and M' are pushouts of X, Y, and M along $I \to Z$, and the morphisms $X' \to M'$ and $Y' \to M'$ are defined by universal property of the pushout. Basically, what this does is embed the move $Y \to M \leftarrow X$ into the position Z by identifying some channels between X, Y, M and Z.

Let us illustrate this in the join-calculus. Take π_2. Its canonical interface is made of two channels a and b. Let Z be the position with an isolated channel a', and a player that knows two channels b' and c', and let $I \to Z$ that associates a' to a and b' to b. Then the global move resulting of pushing-out π_2's seed along $I \to Z$ is:

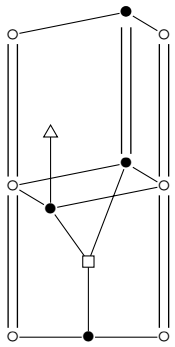


As the drawing shows, the move behaves exactly the same way as the seed did, but in a larger position, and the rest of the position is not affected by the move. The pushout can also identify two or more channels that in the canonical interface to describe a move where a player knows the same channel multiple times (for example, a player x whose first and second channels are the same and who forks).

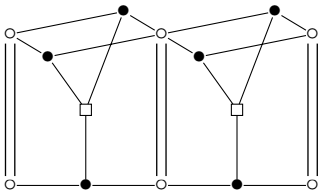
Now that we have defined (global) moves, let us define *plays*, which are the vertical morphisms in our playground. Basically, plays are sequences of moves. However, they are not exactly sequences, since, in concurrent languages, it does not make sense to remember the ordering between two moves that could have happened in any order. Therefore, they will be equivalent to sequences of moves, quotiented by reordering when two moves can be done in any order. Technically, they are defined as generalized pushouts. Take a sequence of moves $X_{i+1} \xrightarrow{s_i} M_i \xleftarrow{t_i} X_i$, then the resulting play is the generalized pushout U:



Let us give two examples using the join-calculus. The first one means to show that plays are basically piled-up moves. Take a position X with a binary player that forks (π_2), and then, its left avatar communicates the name of its second channel on its first, therefore dying (this move, $\sigma_{2,1\langle 2\rangle}$, does not make sense in a closed-world execution, i.e., when the process evolves on its own, but it does if we assume that it can communicate with an environment, and is necessary in the definition of the playground). Then, the resulting play is:



The second example is meant to show that the order between moves is not remembered when these moves can be executed in any order. Take two binary players who share a channel, and both fork, in any order. Then, the resulting play is:



The last elements of the playground left to define are the double cells. They simply are cospan morphisms, i.e., a double cell α as in the diagram below on the left is actually a presheaf morphism as on the right:

$$\begin{array}{ccc} Y & \xrightarrow{k} & Y' \\ U & \overset{\alpha}{\Rightarrow} & V \\ X & \xrightarrow[h]{} & X' \end{array} \qquad \begin{array}{ccc} Y & \xrightarrow{k} & Y' \\ {\scriptstyle s}\downarrow & & \downarrow{\scriptstyle s'} \\ U & \xrightarrow{\alpha} & V \\ {\scriptstyle t}\uparrow & & \uparrow{\scriptstyle t'} \\ X & \xrightarrow[h]{} & X', \end{array}$$

such that the diagram commutes.

1.2 Factorization Systems

A crucial tool that will be used in the construction of playgrounds (or, more precisely, in the proof that a pseudo double category is indeed a playground) is that of *factorization systems*.

Definition 1. *A* factorization system *on a category $\mathcal{C}$ is a pair $(\mathcal{L}, \mathcal{R})$ of a left class of morphisms $\mathcal{L}$ and a right class of morphisms $\mathcal{R}$ such that:*

- *every morphism $f : X \to Y$ in $\mathcal{C}$ admits a factorization rl into a morphism $l : X \to Z$ in $\mathcal{L}$ followed by a morphism $r : Z \to Y$ in $\mathcal{R}$, and this factorization is unique up to a unique isomorphism;*
- *$\mathcal{L}$ and $\mathcal{R}$ contain the isomorphisms and are closed under composition.*

A *weak factorization system* is a factorization system in which the factorization needs not be unique.

Example 1. *Here is a short list of examples of factorization systems and weak factorization systems:*

- *in every category $\mathcal{C}$,* $(\mathbf{Mor}, \mathbf{Iso})$ *and* $(\mathbf{Iso}, \mathbf{Mor})$ *are trivial factorization systems;*
- *in every category $\mathcal{C}$,* $(\mathbf{Mor}, \mathbf{Mor})$ *is a trivial weak factorization system;*
- *in the category* **Set** *of sets and functions,* $(\mathbf{Epi}, \mathbf{Mono})$ *is a factorization system (every map can be factored through its image, and that is the only* $(\mathbf{Epi}, \mathbf{Mono})$ *factorization up to isomorphism);*

- *in the category* **Set**, $(\mathbf{Mono}, \mathbf{Epi})$ *is a weak factorization system (every map* $X \xrightarrow{f} Y$ *can be factored through its graph* $X \xrightarrow{(id_X, f)} X \times Y \xrightarrow{\pi_Y} Y$*, but also through its (generally not isomorphic) cograph* $X \xrightarrow{\iota_X} X + Y \xrightarrow{[f, id_Y]} Y$*).*

Factorization systems enjoy strong properties, such as some cancellation properties:

Property 1. *If* $(\mathcal{L}, \mathcal{R})$ *is a factorization system, then:*

- $\mathcal{L}$ *has the right cancellation property: if* $vu \in \mathcal{L}$ *and* $u \in \mathcal{L}$*, then* $v \in \mathcal{L}$*;*
- $\mathcal{R}$ *has the left cancellation property: if* $vu \in \mathcal{R}$ *and* $v \in \mathcal{R}$*, then* $u \in \mathcal{R}$*.*

Factorization systems are strongly linked to orthogonality between maps.

Definition 2. *A map* $f : X \to Y$ *is* left orthogonal *to* $g : X' \to Y'$ *(or that* g *is* right orthogonal *to* f*) and note it* $f \perp g$ *if for every commutative square*

$$\begin{array}{ccc} X & \xrightarrow{u} & X' \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ Y & \xrightarrow[v]{} & Y', \end{array}$$

there is a unique map $d : Y \to X'$ *such that*

$$\begin{array}{ccc} X & \xrightarrow{u} & X' \\ {\scriptstyle f}\downarrow & \overset{d}{\nearrow} & \downarrow{\scriptstyle g} \\ Y & \xrightarrow[v]{} & Y' \end{array}$$

commutes.

We say that f is *weakly left orthogonal* to g (or that g is *weakly right orthogonal* to f) and note it $f \pitchfork g$ if they fulfill the same conditions, except that the diagonal d needs not be unique.

We write $\mathcal{M} \perp \mathcal{M}'$ if for all f in $\mathcal{M}$ and all g in $\mathcal{M}'$, $f \perp g$. If $\mathcal{M}$ is a class of morphisms of $\mathcal{C}$, we denote by $\mathcal{M}^{\perp}$ its right orthogonal (the class of all morphisms that are right orthogonal to all morhisms in $\mathcal{M}$), and by ${}^{\perp}\mathcal{M}$ its left orthogonal. We do the same for weak orthogonality, replacing $\perp$ by $\pitchfork$.

Theorem 1. *A pair* $(\mathcal{L}, \mathcal{R})$ *of classes of maps in a category* $\mathcal{C}$ *is a factorization system if and only if:*

- *every map* f *admits an* $(\mathcal{L}, \mathcal{R})$*-factorization;*
- $\mathcal{L}$ *and* $\mathcal{R}$ *are replete (stable under isomorphisms);*
- $\mathcal{L} \perp \mathcal{R}$*.*

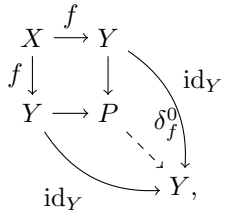
Our main interest lies in "strong" factorization systems (as opposed to weak ones). However, there are links between the two notions, and weak factorization systems come with interesting constructions, such as the *small object argument*. Take a set S of maps of $\mathcal{C}$, then $({}^{\pitchfork}(S^{\pitchfork}), S^{\pitchfork})$ is a weak factorization system, and we say it is cofibrantly generated by S (note that a weak factorization system $(\mathcal{L}, \mathcal{R})$ may be cofibrantly generated by different sets). Telling which morphisms are in the right class is quite straightforward: they're the morphisms that are weakly right orthogonal to every morphism in S, which is usually a property that is easy to understand. What is less clear is what morphisms are in the left class. The answer to this question is given by Quillen's *small object argument* [3]. In any locally presentable category (such as presheaf categories, the ones we are interested in), the left class is the smallest class of maps that:

- contains S,
- is stable under pushouts, transfinite composition, and retracts.

This class can also be described as the class of transfinite compositions of pushouts of elements of S. We note this class $\mathrm{Sat}(S)$.

Example 2. *In the category of sets and functions, take S to be the one-function set containing the only function $f : \{a, b\} \to \{c\}$. Then $S^{\pitchfork}$ is the set of all injective functions. Indeed, if $g : X \to Y$ is not injective, then there are x and y, different, in X such that $g(x) = g(y)$. Take $u : a \mapsto x, b \mapsto y$ and $v : c \mapsto g(x)$, then the orthogonality square commutes, but there is no diagonal that can make both triangles commute. If, however, g is injective, then the only possibility to make a commutative square from f to g is for u to map both a and b to the same element in X, and therefore there is a diagonal lifting. Furthermore, ${}^{\pitchfork}(S^{\pitchfork})$ the set of all surjective functions. Indeed, pushouts of f are surjective, and so are their transfinite compositions, and any surjective function can easily be written as a transfinite composition of pushouts of f. Therefore, the resulting factorization system is* $(\mathbf{Epi}, \mathbf{Mono})$.

Another very interesting link between strong and weak factorization systems is that, if the category has nice properties, then a variant of the small object argument still works for strong factorization systems. First, let us define what the *codiagonal* of a map $f : X \to Y$ is. It is the map δ^0_f defined by universal property of the pushout:



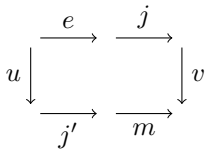
where P is the pushout of f and f. If we note $\delta^0(S)$ the class of codiagonals of elements of S, the following theorem holds [6]:

Theorem 2. *Let S be a set of maps between small objects in a cocomplete category $\mathcal{C}$. Then the pair $(Sat(S \cup \delta^0(S)), S^{\perp})$ is a factorization system.*

Finally, we will need another notion related to factorization systems to explain this work, and that is the notion of *double factorization system* [8].

Definition 3 (Double Factorization System)**.** *A double factorization system is made of three classes of maps $(\mathcal{E}, \mathcal{J}, \mathcal{M})$ such that:*

- $\mathbf{Iso} \cdot \mathcal{E} \subseteq \mathcal{E}$, $\mathbf{Iso} \cdot \mathcal{J} \cdot \mathbf{Iso} \subseteq \mathcal{J}$, *and* $\mathcal{M} \cdot \mathbf{Iso} \subseteq \mathcal{M}$,
- $\mathbf{Mor} = \mathcal{M} \cdot \mathcal{J} \cdot \mathcal{E}$,
- *for any commutative diagram*



with $e \in \mathcal{E}$, $j, j' \in \mathcal{J}$, and $m \in \mathcal{M}$, there are unique "diagonals" s and t such that $se = u$, $tj = j's$ and $mt = v$.

Double factorization systems have nice properties that relate them to standard factorization systems.

Proposition 1. *If $(\mathcal{E}, \mathcal{J}, \mathcal{M})$ is a double factorization system, then* **Iso** $= \mathcal{E} \cap \mathcal{J} \cap \mathcal{M}$*, and $(\mathcal{E}, \mathcal{M} \cdot \mathcal{J})$ and $(\mathcal{J} \cdot \mathcal{E}, \mathcal{M})$ are both factorization systems.*

Corollary 1. *If $(\mathcal{E}, \mathcal{J}, \mathcal{M})$ is a double factorization system, then $\mathcal{E} = (\mathcal{M} \cdot \mathcal{J})^{\perp}$, $J = {}^{\perp}\mathcal{E} \cap \mathcal{M}^{\perp}$, and $\mathcal{M} = {}^{\perp}(\mathcal{J} \cdot \mathcal{E})$.*

There is even a bijective correspondence between the double factorization systems and pairs of *comparable* factorization systems. We say that two factorization systems $(\mathcal{E}, \mathcal{N})$ and $(\mathcal{D}, \mathcal{M})$ are comparable if one of the following (equivalent) properties holds: (1) $\mathcal{E} \subseteq \mathcal{D}$, (2) $\mathcal{M} \subseteq \mathcal{N}$, (3) $\mathcal{E} \perp \mathcal{M}$, (4) $\mathcal{D} = (\mathcal{D} \cap \mathcal{N}) \cdot \mathcal{E}$, (5) $\mathcal{N} = \mathcal{M} \cdot (\mathcal{D} \cap \mathcal{N})$.

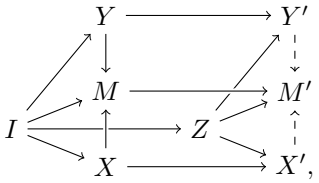
Theorem 3. *For every double factorization system $(\mathcal{E}, \mathcal{J}, \mathcal{M})$, $(\mathcal{E}, \mathcal{M} \cdot \mathcal{J})$ and $(\mathcal{J} \cdot \mathcal{E}, \mathcal{M})$ are comparable factorization systems. For every pair of comparable factorization systems $(\mathcal{E}, \mathcal{M})$, $(\mathcal{D}, \mathcal{N})$, $(\mathcal{E}, \mathcal{D} \cap \mathcal{N}, \mathcal{M})$ is a double factorization system. This correspondence is bijective.*

2 Building Playgrounds

The study of the join-calculus and attempts at generalizing the approach used to define a playground from a category of presheaves has lead us to a deeper insight of some properties that must be asked of the basic blocks the plays are based on, namely *seeds*.

2.1 Canonical Interfaces and Moves

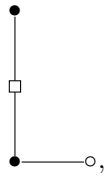
As stated in the introduction, each seed has a *canonical interface* that describes how a seed, i.e., a local move, can be embedded in a larger position, creating a global move. The canonical interface is an *interface*, i.e., a presheaf that only contains channels. Basically, these channels are the ones that may be shared between the local move and a position, to create a global move. More precisely, every seed $Y \to M \leftarrow X$ is equipped with a canonical interface I and morphisms from I to X and Y (such that they're equal when composed with $X \to M$ and $Y \to M$ respectively). Global moves are just pushouts of this seed along a morphism from the interface to a position Z:



where the dashed arrows are obtained by universal property of the pushout. Basically, it just amounts to pasting a seed next to a position, while identifying some channels between them.

In the cases of CCS, the π-calculus, or the join-calculus, finding the right canonical interfaces is very easy. Indeed, since all channels behave well (they are only ever created or left untouched, never destroyed, much less fused or duplicated), the canonical interface corresponds exactly to the channels that are present in the initial position of the seed.

However, generalizing this definition of the canonical interface as the channels present in the initial position makes it impossible to define a language with garbage collection, i.e., in which channels may be destroyed. Indeed, the problem is that if a channel is destroyed by a move $Y \to M \leftarrow X$, i.e., that it exists in X but not in Y, then there is in general no way to define a morphism from the canonical interface I to Y. Indeed, consider the following, simple example, where the only seed is:

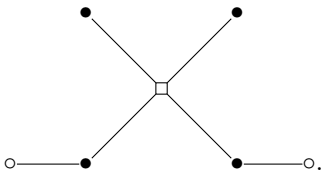


i.e., a move where a player that knows a single channel can destroy this channel. Then, if the canonical interface is the interface that contains the channel in X, then it contains exactly one channel, and there is no morphism from this interface to Y. The reader may think at first sight that such a move should not be accepted anyway, since it would be impossible to write the final position of a global version of this move, even if the initial position is defined, but that exactly means that a player cannot destroy a channel on his their own accord if they're not the only one who knows of this channel.

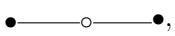
To be able to account for garbage collection, we therefore decided to change the definition of the canonical interface. Instead of taking the channels in X, we take the pullback of the channels in X and those in Y. Then, there are obviously morphisms from I to X and Y, and they commute when composed with $X \to M$ and $Y \to M$. The pullback gives the channels that are both in X and Y: if channels are only created, destroyed, or left untouched (i.e., they are never fused or duplicated), then the pullback corresponds to the channels that are left untouched (if there are some fused or duplicated channels, then things are a bit more complicated, and there are several copies of the "same" channel in the interface). For example, in the example above, the canonical interface would be empty. In the case where channels are only created or left untouched, as in CCS or the π-calculus, the canonical interface consists of the channels in X, which is what we expect. It also gives a satisfactory canonical interface in

the case of interaction nets, where the channels can be fused, but we will not go into the details.

However, this solution is not exactly satisfactory. Indeed, consider a playground where one of the moves is of the form:

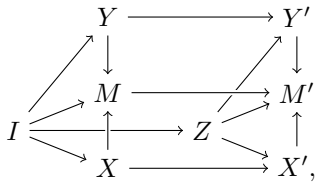


We would like this move to be playable in the following position:



since both players want to destroy that channel, it should not matter whether they share it or not. However, it is not possible, since it would ask that the map $I \to Z$ identifies these channels, but I is empty by definition. Therefore, this definition, though better than the first one, is not satisfactory either.

The core problem is that the channels that can be shared in the initial position are exactly the channels that can be linked to the global position. However, changing the definition of the canonical interface will not give the solution to this problem, since the channels that can be shared are exactly the channels in the canonical interface, and so are the channels that can be linked to the global position. Therefore, the definition that needs to be changed is that of moves, so that the canonical interface will not handle two roles that should be different exactly the same way. Canonical interfaces are still defined as pullbacks, as before, but the definition of a move has been changed to reflect this fact. A move is still a cospan that has the same shape as before:



except we do not ask that the rectangles be pushouts. Instead, we ask that they be collectively surjective pullbacks, and that all functions be injective in dimension greater than 0. First, this definition allows moves such as the one discussed above, even with the channel shared by the two players (take $Z = \emptyset$). Furthermore, in the simple cases where all the t's and s's in seeds $Y \xrightarrow{s} M \xleftarrow{t} X$ are injective (such as in CCS, the π-calculus, or the join-calculus), the old definition of moves is a particular case of this new definition. Indeed, in that case, we have that the following pushout:

$$\begin{array}{ccc} I & \xrightarrow{k} & Z \\ {\scriptstyle l}\downarrow & & \downarrow{\scriptstyle f} \\ X & \xrightarrow[g]{} & X' \end{array}$$

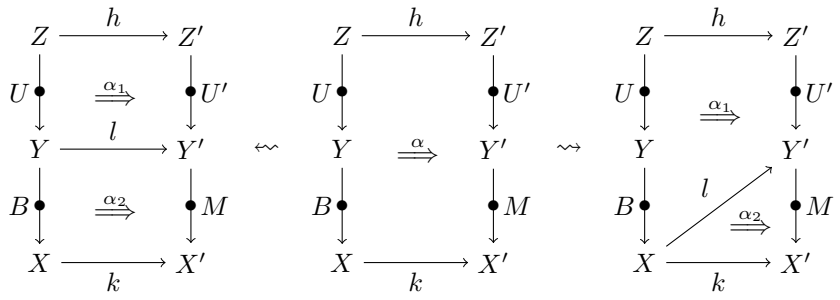
is of course collectively surjective, and that all the functions are injective in dimension greater than 0. Furthermore, in adhesive categories (such as presheaf categories), pushouts of monomorphisms are pullbacks, so the square is a pullback square.

However, this definition is, again, not entirely satisfactory. Indeed, take a move that creates two channels. Then, using this definition, it is possible for the two channels to be identified in the final position, which is a behavior that we do not wish to capture. This is still a work in progress, and there are multiple ways to solve the problem. One would be to allow the strange behavior described above, where only one channel is created when two different ones should be, and to rule out this possibility by constraining the strategies a player can play. Another way would be to rule out cases where the morphisms in the seeds are not injective. However, it would be better to find a solution without having to resort to any of these, simply by understanding better what the good definition of a move should be, and it seems that the right tool to understand what this definition should be is (very close to) a factorization system, a version of which is given for graphs in section 3.

2.2 Right Decomposition Axiom

We have proved one of the playground axioms in the general case, the proof can be found in annex B. The right decomposition axiom states the following:

Theorem 4. *Any double cell (in the middle), where B is a basic move and M is a move, decomposes into exactly one of the forms (on the left and on the right):*



Basically, this axiom states that, if B is a basic move (which are a certain type of move), then a morphism into a play can be decomposed into morphisms between smaller plays: either B can be embedded in the first move of the play M and U into U', or all of $B \bullet U$ is embedded into U'.

2.3 Building Factorizations in the Finite Case

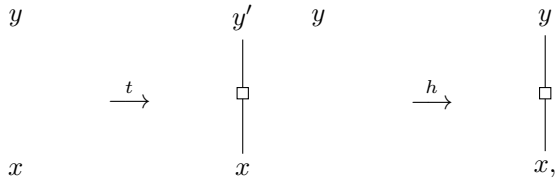
In this part, we discuss how to build factorizations in the finite case. The idea is that plays are finite objects, and that we would like a way to factor a morphism between plays as the composition of two morphisms that show different things about the play. We want the first class of morphisms to show how a play is included in another one "in time", and the second one how it is included in the other "in space". We also want to show that when a morphism f between two plays U and V is factorized this way, it is factorized through another play W.

Here, to factorize morphisms, we use the factorization system cofibrantly generated by the set T_0 of all "t-legs" of seeds, the morphisms from the initial position of a seed to its middle presheaf, that is, if $Y \xrightarrow{s} M \xleftarrow{t} X$ is a seed, then $t \in T_0$. We call them t-legs because they are always labelled with the letter t (the letter t itself stands for "target", while s stands for source, reminding that, in the playground, plays are arrows from their final position to their initial position). Let (T, H) denote this factorization system. A morphism $h : X \to Y$ is in H if, when Y "contains" a move whose initial position is in X, X already contains the whole move. This intuition is made more precise by the orthogonality square:

$$\begin{array}{ccc} X_0 & \xrightarrow{u} & X \\ {\scriptstyle t}\downarrow & & \downarrow{\scriptstyle h} \\ M_0 & \xrightarrow[v]{} & Y. \end{array}$$

Such an orthogonality square exists when $v : M_0 \to Y$ and $u : X_0 \to X$ exist such that the square commutes (what we meant above by "Y contains a move and X its initial position"), and the existence of a unique diagonal filler $d : M_0 \to X$ shows that X already "contains" the whole move. According to the variant of the small object argument for strong factorization systems, a morphism t is in T when it is the transfinite composition of pushouts of t-legs or codiagonals of t-legs. We can factorize any morphism between plays $f : U \to V$ as $f = ht$, where $t : U \to W$ is in T and $h : W \to V$ is in H, and we want W to be a play.

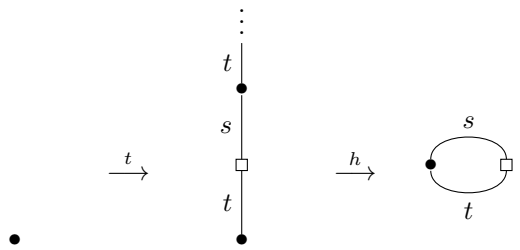
The first idea we had was that, if a morphism between plays is injective, then its factors should be injective as well, which would have guaranteed the finiteness of all objects, since V is finite, and would have been a step towards showing that W is a play. However, that is impossible. Consider a move m that goes from a nullary player to another nullary player. Now, take the play U that only contains two nullary players, and V that only contains m (and, therefore, the two players). There is an obvious injective morphism $f : U \to V$ that maps each player in U to a player in V. The factorization of that morphism is not injective. Indeed, its factorization is the following (where we named the players for convenience):



where t creates the whole move starting at x, with a new y' as its final position, and h identifies y and y' (and therefore is not injective). It can easily be checked that t is in T (as a pushout of a t-leg) and that h is in H. Also, that is the only factorization up to isomorphism, by definition of a factorization system.

Another idea we had was that, if the factorization system does not preserve injectivity, maybe it would at least preserve finiteness. However, that is also wrong. Here, however, the counter example is cyclic, in the sense that the target

of one move is also its source, and it therefore is not a play. Again, take the move m that goes from a nullary player to another nullary player. Take U the play with just one nullary player, and V the play that contains m whose initial position is also its final position. There is a morphism from U to V, but W in its factorization $U \xrightarrow{t} W \xrightarrow{h} V$ is not finite. Indeed, its factorization is (we wrote the t-legs and s-legs, because they are not necessarily obvious in this example):



Again, it can easily be checked that t is in T and h in H, and the presheaf in the middle is not finite.

However, plays are not simply finite objects. The counter-example above has a particular shape that plays cannot have. Indeed, in a sense, plays are acyclic objects, as it would not make sense for a play to contain a move that creates a new player, but also needs that player to be played. Formally, a *causal graph* can be defined for each presheaf, and only presheaves with an acyclic causal graph can be plays. Basically, U's causal graph contains all the moves, players, and channels of U, an arrow from a move to a player (or channel) if that player is in U's initial position, and an arrow from a player (or channel) to a move if that player is created by that move (it is slightly more complex, but that is the main idea, for more details, see [1]).

We can write the following algorithm to build the factorization of a morphism between plays. Note that this algorithm is a variant of the small object argument, which is a transfinite construction, so it does not stop in the general case, but it does stop on plays, because they are acyclic and finite. The algorithm is given a morphism $U \xrightarrow{f} V$ between plays, and outputs a factorization $U \xrightarrow{t} W \xrightarrow{h} V$ of f with t in T, h in H, and W a play. This algorithm keeps a factorization $U \xrightarrow{t} W \xrightarrow{h} V$ of f and loops until it has reached a fixed point (at the beginning of the algorithm, $t = f$, $h = \mathrm{id}_V$ and $W = V$). At each iteration, it does the following:

- take I the set of commutative squares from a morphism $t : X \to Y$ in T_0 to h that has no diagonal filler, and take the pushout:

$$\begin{array}{ccc} \coprod_{i\in I} X_i & \xrightarrow{[u_i]_{i\in I}} & W \\ {\scriptstyle \coprod_{i\in I} f_i}\downarrow & & \downarrow{\scriptstyle \widetilde{t}} \\ \coprod_{i\in I} Y_i & \longrightarrow & \widetilde{W}, \end{array}$$

  and call the morphism $\widetilde{W} \to V$ obtained by universal property of the pushout $\widetilde{h}$,

- then do the same with all commutative squares from morphims in $\delta^0(T_0)$ to $\widetilde{h}$, call the new pushout W', $\widetilde{t'} : \widetilde{W} \to W'$, the morphism $W' \to V$ obtained by universal property of the pushout h', and $t' = \widetilde{t'}\widetilde{t}t$.
- If $W' = W$, $t' = t$, and $h' = h$, then stop. Otherwise, loop with the new values $t = t'$, $h = h'$, and $W = W'$.

Next, we describe the idea of the proof that this algorithm works (the formal proof has yet to be completely done). Each loop is done in finite time, since all the objects considered are finite, and there is only a finite number of commutative squares to be considered. We need to show that there is only a finite number of loops. Actually, the number of loops the algorithm runs is less or equal than the height of V's causal graph (give or take one turn to realize it has come to a fixed point). The height of a move m in the causal graph is 0 if it has no path to another move, and $n+1$ if n is the maximum height of other moves m has a path to. The height of the causal graph is the maximum height of a move in it. The reason why the number of loops is not greater than the height of the causal graph is because, at loop n, the smallest subpresheaf of W that contains all the moves of height less than n is left untouched by the algorithm. W being a play is an invariant of the algorithm (while $\widetilde{W}$ is not a play in general), as well as t being in T. h being in H at the end of the algorithm comes from the fact that there is no commutative diagram from a morphism in T_0 or $\delta^0(T_0)$ to h that does not have a unique diagonal filler (this part is a bit tricky, especially the fact that the second pushout does not erase elements introduced by the first one, which would make the statement false).

3 A Factorization System on Graphs

Here, we introduce and study the properties of a double factorization system on graphs. The basic idea is that graphs are proto-playgrounds. Graphs are known to be presheaves on the category:

$$\begin{array}{ccc} & e & \\ s_0 & \curvearrowleft \; \curvearrowright & t_0 \\ & v. & \end{array}$$

Indeed, such a presheaf G is made two sets, a set of vertices $G(v)$ and a set of edges $G(e)$, with two functions $s_0, t_0 : G(e) \to G(v)$ that are the source and target of each edge. Graphs can be thought of as a playground with a single player object v and a single move object e, whose seed is $v \xrightarrow{s_0} e \xleftarrow{t_0} v$ (with a slight abuse of notation, as we write the object instead of its representable presheaf).

3.1 A Factorization System on Graphs

There are three classes of morphisms we are interested in: $T = \{t_0\}$, which represent how a play can evolve "forward" in time, $S = \{s_0\}$, which represents how a play can evolve "backward" in time, and $H = (T \cup S)^{\perp}$, which represents some sort of spatial rearrangement, without "moving in time". Therefore, the construction we are interested in will be a ternary construction. We will use

a double factorization system $(\overline{T}, \overline{S}, H)$. It is built by using two comparable factorization systems, using theorem 3: $({}^{\perp}(T^{\perp}), T^{\perp})$ and $({}^{\perp}((T \cup S)^{\perp}), H)$, which gives us that $T = {}^{\perp}(T^{\perp})$ and $S = T^{\perp} \cap {}^{\perp}((T \cup S)^{\perp})$.

$\overline{T}$ is fairly easy to understand: a morphism $t : G \to G'$ in $\overline{T}$ is a transfinite composite of pushouts of t and codiagonal of t. In terms of graphs, G' is G where some edges to the same target have been identified (potentially recursively), and each vertex is the root of a (potentially infinite) tree of edges pointing towards this root.

A morphism $s : G \to G'$ in $\overline{S}$ is a transfinite composite of pushouts of t_0, s_0, and their codiagonals and is also orthogonal to t. In terms of graphs, G' is G where some edges from the same source (and not from the same target) have been identified (potentially recursively), and each node is the root of a (potentially infinite) tree of edges pointing either towards the root or not, but whose first level of edges must not point to the root.

A morphism $h : G \to G'$ in H is orthogonal to both t_0 and s_0. In terms of graphs, it means that G' is G where some vertices and all the edges pointing to and from them have been identified (two different vertices that h identifies must have the same number of edges pointing to and from them, and each pair of identified edges must point to vertices that are identified by h), plus another graph G'' that is not connected to G.

This factorization system is interesting because it is probably close to the answer to the question "What is the good definition of a move?" and can be used to prove some properties of playgrounds more easily.

3.2 Properties

There are a number of properties that we want this factorization system to have for the axiom of fibration (one of the playground axioms) to hold:

- $T \cdot H \subseteq H \cdot T$
- the pullback of a morphism in H along a morphism in $\overline{S}$ is in H, and the pullback of a morphism in $\overline{S}$ along a morphism in H is in $\overline{S}$,
- the pushout of a morphism in H along a morphism in $\overline{T}$ is in H, and the pushout of a morphism in $\overline{T}$ along a morphism in H is in $\overline{T}$.

The first property has been proven to hold true, and the complete proof can be found in appendix C. The other two properties seem to hold, but proving them seems to be quite difficult.

Quillen factorization systems also have some nice properties and have been extensively studied in category theory, and it would have been interesting if this double factorization system was a Quillen factorization system. However, it is not the case. Recall from [8] that a double factorization system $(\mathcal{E}, \mathcal{J}, \mathcal{M})$ is a Quillen factorization system if and only if both $\mathcal{E} \cdot \mathcal{M} \subseteq \mathcal{M} \cdot \mathcal{E}$ and for all j in $\mathcal{J}$, if there is an e in $\mathcal{E}$ such that ej is in $\mathcal{E}$ or an m in $\mathcal{M}$ such that jm is in $\mathcal{M}$, then j is an isomorphism.

We know that the first property holds true, but the second one does not. Indeed, consider G that has one vertex x and no edge, G' with two vertices x and y and no edge, and G'' with three vertices x, y, and z and an edge between y and z. Take $h : G \to G'$ and $s : G' \to G''$ to be the graph inclusions, h is in H, s in $\overline{S}$, and hs in $\overline{S}$, but s is not an isomorphism.

4 Conclusion

The main goal of this work was to develop tools to allow the automated construction of playgrounds from their most basic blocks, seeds, under some conditions on them. Some of the more difficult axioms have been proven, such as the right decomposition axiom, but others, such as the fibration axiom or the left decomposition axiom, remain unproven yet. However, some drastic advances have been made: the key to understanding playgrounds better, and therefore to prove their axioms in the general case partly lies with factorization systems. The study of one candidate to be a good double factorization system on the category of graphs is encouraging, but not completely satisfactory yet: for example, the dissymmetry between the class $\overline{T}$ and $\overline{S}$ shows that it is perhaps not the most well-suited tool for this work.

Of course, the work left to be done on playgrounds is a real challenge. So far, only CCS and the π-calculus have been proven to form playgrounds, and new techniques have to be invented for each new language. The first goal to achieve is the semi-automated construction of playgrounds based on the seeds, to have a wide range of possible languages to study. Then, once this is done, studying the semantics given by the playgrounds to show that they are adequate. Finally, studying the morphisms between playgrounds could lead to an understanding of what a semantics-preserving transformation, such as compilation, should be.

References

[1] Clovis Eberhart. Towards a Theory of Programming Languages. Master's thesis, ENS Cachan, 2013.

[2] Clovis Eberhart, Tom Hirschowitz, and Thomas Seiller. Fully-abstract concurrent games for pi. 20 pages, submitted, 2013.

[3] Richard Garner. Understanding the small object argument. *Applied Categorical Structures*, 17(3):247–285, 2009.

[4] Tom Hirschowitz. Full abstraction for fair testing in CCS (expanded version). 80 pages, under revision for LMCS., 2013.

[5] Tom Hirschowitz and Damien Pous. Innocent strategies as presheaves and interactive equivalences for CCS (expanded version). *Scientific Annals of Computer Science*, 22(1):147–199, 2012. 53 pages. Expanded version of ICE '11 paper DOI 10.4204/EPTCS.59.2 .

[6] André Joyal. Joyal's CatLab. `http://ncatlab.org/joyalscatlab/published/HomePage`.

[7] Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer, 1998.

[8] Aleš Pultr and Walter Tholen. Free Quillen Factorization Systems. *Georgian Mathematical Journal*, 9(4):807–820, 2002.

A Playground for the join-calculus

This appendix is taken nearly entirely and without change from my Master's Thesis [1].

Consider the graph $G_{\mathcal{C}}$ with vertices:

- a vertex $*$
- for every $n \in \mathbb{N}$, a vertex $[n]$
- for every $n \geq 2$, a vertex $\langle n \rangle$
- for every $n \in \mathbb{N}$, vertices π_n^l, π_n^r, π_n, δ_n^l, δ_n^r, δ_n, reac_n, and freac_n
- for every $n \in \mathbb{N}$ and every $i, j \in \{1, ..., n\}$, a vertex $\sigma_{n,j\langle i \rangle}$
- for every $n \in \mathbb{N}$, every $p \in \mathbb{N}$, every $i, j \in \{1, ..., p\}$, every $q \in \mathbb{N}$ and every $k, m \in \{1, ..., q\}$, a vertex $\tau_{n,p,j\langle i \rangle,q,m\langle k \rangle}$

and edges:

- for every vertex $[n]$, edges $s_1, ..., s_n : * \to [n]$
- for every vertex $\langle n \rangle$, edges $s_1, .., s_n : * \to \langle n \rangle$
- for every vertex π_n^l, edges $s, t : [n] \to \pi_n^l$ (idem for π_n^r)
- for every vertex π_n, an edge $l : \pi_n^l \to \pi_n$ and an edge $r : \pi_n^r \to \pi_n$
- for every vertex δ_n^l, an edge $t : [n] \to \delta_n^l$ and an edge $s : \langle n+2 \rangle \to \delta_n^l$
- for every vertex δ_n^r, an edge $t : [n] \to \delta_n^r$ and an edge $s : [n+2] \to \delta_n^r$
- for every vertex δ_n, an edge $l : \delta_n^l \to \delta_n$ and an edge $r : \delta_n^r \to \delta_n$
- for every vertex reac_n, an edge $t : \langle n \rangle \to \mathrm{reac}_n$ and an edge $s : [n+2] \to \mathrm{reac}_n$
- for every vertex freac_n, an edge $f : \mathrm{reac}_n \to \mathrm{freac}_n$
- for every vertex $\sigma_{n,j\langle i \rangle}$, an edge $t : [n] \to \sigma_{n,j\langle i \rangle}$
- for every vertex $\tau_{n,p,j\langle i \rangle,q,m\langle k \rangle}$, an edge $d : \mathrm{freac}_n \to \tau_{n,p,j\langle i \rangle,q,m\langle k \rangle}$, an edge $l : \sigma_{p,j\langle i \rangle} \to \tau_{n,p,j\langle i \rangle,q,m\langle k \rangle}$, and an edge $r : \sigma_{q,m\langle k \rangle} \to \tau_{n,p,j\langle i \rangle,q,m\langle k \rangle}$

Now, we define $\mathcal{C}$ to be the free category on $G_{\mathcal{C}}$, modulo the relations:

$t \circ s_i = s \circ s_i$	for any $\pi_n^l, \pi_n^r, \delta_n^l, \delta_n^r, \mathrm{reac}_n$, $i \in n$
$l \circ s \circ s_i = r \circ s \circ s_i$	for any δ_n, $i \in \{n+1, n+2\}$
$l \circ t = r \circ t$	for any π_n, δ_n
$f \circ d \circ t \circ s_{n-1} = l \circ t \circ s_j$ $f \circ d \circ t \circ s_n = r \circ t \circ s_m$ $f \circ d \circ s \circ s_{n+1} = l \circ t \circ s_i$ $f \circ d \circ s \circ s_{n+2} = r \circ t \circ s_k$	for any $\tau_{n,p,j\langle i \rangle,q,m\langle k \rangle}$

The reasoning behind the construction of this category is the following. The object $*$ represents channels on which the processes communicate.

The objects $[n]$ are processes that can communicate on n channels, the $s_i : * \to [n]$ represent the channels on which the process can communicate (since positions will be modelled as presheaves, the s_i's will go from a process to a channel).

The objects $\langle n \rangle$ are definitions that know n channels and listen on channels $n-1$ and n, the s_i's represent the channels it knows.

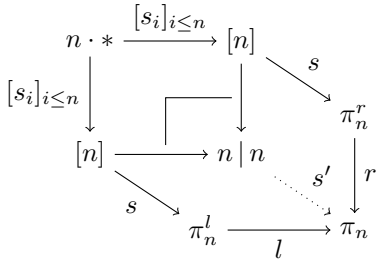
The object $\sigma_{n,j\langle i \rangle}$ represents a process that knows n channels and sends the name i on channel j, i.e. that a process can be of the form $x\langle u \rangle$.

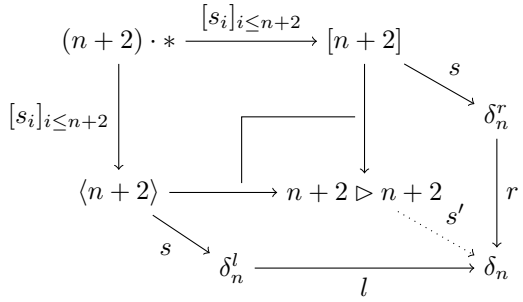
The object π_n represents the move of a process that knows n channels and forks into two different processes, i.e. that a process can be of the form $P_1 \mid P_2$ (π_n^l and π_n^r are the left and right "half-fork" moves, with l and r being morphisms that link a fork move to its left and right half-fork moves). The morphisms t (target) and s (source) represent respectively the player before and after the move (note that "source" and "target" may sound counter-intuitive). We require that $t \circ s_i = s \circ s_i$ because the avatars of the forking player know exactly the same channels as this player, and the fact that $l \circ t = r \circ t$ is because π_n is a forking move, so there is only one player before the move is made.

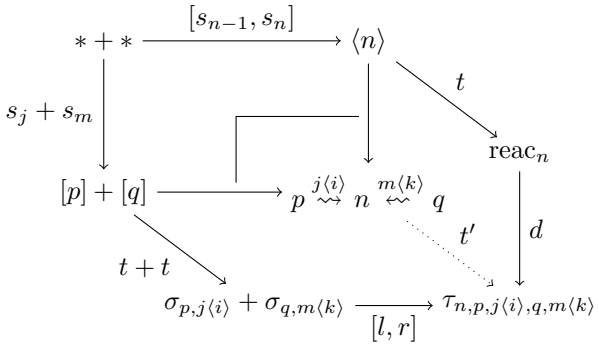
The object δ_n represents the move of a process that knows n channels and forks into a definition that listens on two new channels $n+1$ and $n+2$ and another process (the two channels it creates are shared between the definition and the process), i.e. that a process can be of the form def D in P (δ_n^l and δ_n^r are the left and right half-fork moves). The morphism equations required express exactly the same constraints as for π_n, plus that the two new channels are indeed shared between the definition and the process.

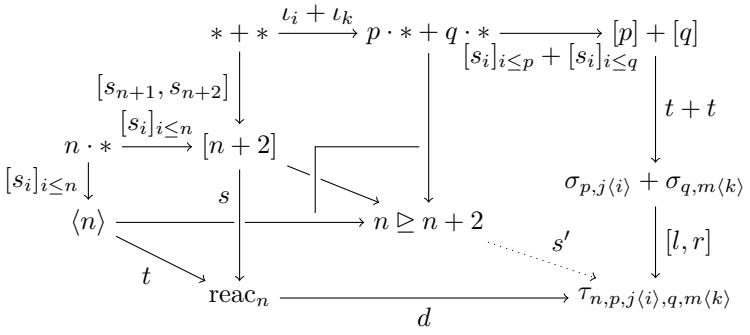
The object $\tau_{n,p,j\langle i \rangle,q,m\langle k \rangle}$ represents the synchronization move between a definition that listens on channels $n-1$ and n and two processes that send the channel names i and k on channels j and m respectively (reac_n is the part of the synchronization that creates a new process from a definition and freac_n is here for technical purposes). The morphism equations required express the fact that channels $n-1$ and n (on which the definition listens) are indeed channels j and m (on which the names are sent) and that channels $n+1$ and $n+2$ of the newly created process are channels i and k (that were sent to the definition).

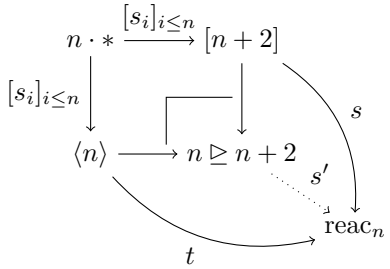
Before we define the seeds, we have to define some positions $n \mid n$, $n{+}2 \triangleright n{+}2$, $p \overset{j\langle i \rangle}{\rightsquigarrow} n \overset{m\langle k \rangle}{\leftsquigarrow} q$, $n \trianglerighteq n+2$, and $n \trianglerighteq n+2, p, q$. They are defined as below, and the corresponding morphisms t' and s' are defined by the universal properties of these colimits.



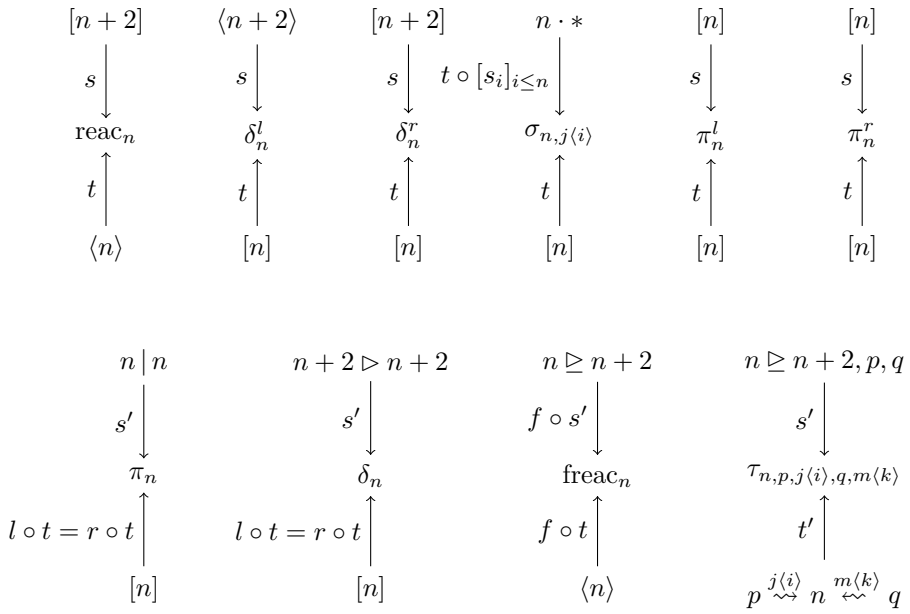








Now we can define the seeds of the join-calculus:



B Right Decomposition Axiom: A Proof

B.1 Conditions on the seeds

- Any t in T is a monomorphism.
- Any s in S is a monomorphism.
- For all seeds $Y \to M \leftarrow X$, X contains at least a player.
- For all basic seeds $Y \overset{s}{\to} B \overset{t}{\leftarrow} X$, and all c of dimension 1, $s_c(Y(c)) \cap t_c(X(c)) = \emptyset$.
- for all seeds $Y \overset{s}{\to} M \overset{t}{\leftarrow} X$ and $Y' \overset{s'}{\to} M' \overset{t'}{\leftarrow} X'$ such that there is a natural transformation $\alpha : M \to M'$, there exist two natural transformations $\alpha_X : X \to X'$ and $\alpha_Y : Y \to Y'$ such that:
  - the diagram

$$\begin{array}{ccc} Y & \xrightarrow{\alpha_Y} & Y' \\ {\scriptstyle s}\downarrow & & \downarrow{\scriptstyle s'} \\ M & \xrightarrow{\alpha} & M' \\ {\scriptstyle t}\uparrow & & \uparrow{\scriptstyle t'} \\ X & \xrightarrow[\alpha_X]{} & X' \end{array}$$

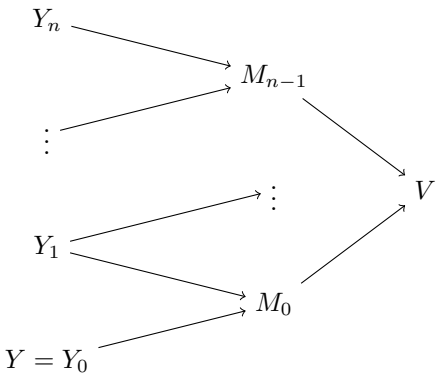
    commutes;

– if $y \in Y$ is a player and $s(y)$ is not in the image of t, then $\alpha(s(y))$ is not in the image of t'.

B.2 Proof

Lemma 1 (Existence of a Path)**.** *For any play $Z \xrightarrow{V} Y \xrightarrow{U} X$ and any core μ in V, there is a path in $G_{U \bullet V}$ from μ to some player in Y, and that path alternates between players and cores.*

Proof. Let us choose a decomposition of V into moves:



and prove the stronger following statements by induction:

- for any player x in Y_i, there is a path from x to some player y in Y, and that path alternates between players and cores;
- there is a path from the core μ in M_i to some player y in Y, and that path alternates between players and cores.

Let us prove the statement for Y_i:

- for $i = 0$, it is obviously true for $Y_0 = Y$: for every x in Y, the empty path goes from x to itself;
- for $i > 0$, we assume the corresponding statements are true for Y_{i-1} and M_{i-1} respectively, then:
  - either x is created by the core μ in M_{i-1}, in which case $x \to \mu$ in $G_{U \bullet V}$, and since μ has a path that alternates between players and cores to a player in Y by induction, then so does x;
  - or x is already in Y_{i-1}, so the statement is true by induction.

Let us prove the statement for the core μ in M_i, assuming the corresponding statement is true for Y_i: since for any seed $Y \to M \leftarrow X$, there is at least one player in X_0, we know that there is at least one player x in Y_i such that $\mu \to x$. Therefore, since x has a path that alternates between players and cores to a player in Y by induction, so does μ. □

Lemma 2 (Arrow Transfer)**.** *Let U and V be plays, $\alpha : U \to V$, and x and y in U such that $c(x) \to c(y)$ in G_U. Then:*

- *if x is a player and y a move,*
- *or x is a move and y a player,*

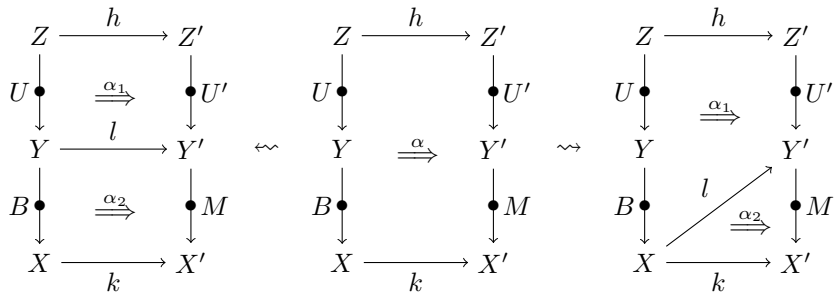
$c(\alpha(x)) \to c(\alpha(y))$ *in* G_V.

Proof. Let us study both cases:

- If x is a player and y is a move, then $c(x) = x$, and $x \to c(y)$, which means that x is created by $c(y)$. Let us denote $Y \to M \leftarrow X$ the seed that corresponds to $c(y)$ and $X' \to M' \leftarrow Y'$ the one that corresponds to $c(\alpha(y))$. Note that $c(\alpha(y)) = c(\alpha(c(y))$, so there is a morphism $c(y) \to c(\alpha(y))$, and therefore a cospan morphism between their seeds, so $\alpha(x)$ is created by $c(\alpha(y))$, so there is an arrow $\alpha(x) \to c(\alpha(y))$ in G_V.
- If x is a move and y is a player, then $c(y) = y$, and $c(x) \to y$, which means that y is initial for $c(x)$. Let us denote $Y \to M \leftarrow X$ the seed that corresponds to $c(x)$ and $X' \to M' \leftarrow Y'$ the one that corresponds to $c(\alpha(x))$. Note that $c(\alpha(x)) = c(\alpha(c(x))$, so there is a morphism $c(x) \to c(\alpha(x))$, and therefore a cospan morphism between their seeds, so $\alpha(y)$ is initial for $c(\alpha(x))$, so there is an arrow $c(\alpha(x)) \to \alpha(y)$ in G_V.

□

Lemma 3 (Right Decomposition)**.** *Any double cell (in the middle), where B is a basic move and M is a move, decomposes into exactly one of the forms (on the left and on the right):*



Proof. Let b be B's core. $\alpha(t_3(b))$ is in $M \bullet U'$, therefore either in the image of t'_3 or in that of s'_3. Let us study both cases:

- $\alpha(t_3(b))$ is in the image of t'_3: then there is a b' in M such that $t'_3(b') = \alpha(t_3(b))$ (this b' is actually unique because Y is empty in dimension greater than 1, so $(M \bullet U')(c) = M(c) + U'(c)$ for objects c of dimension greater than 1). Since $Y' \to M \leftarrow X'$ is a move, it stems from a diagram of the form:

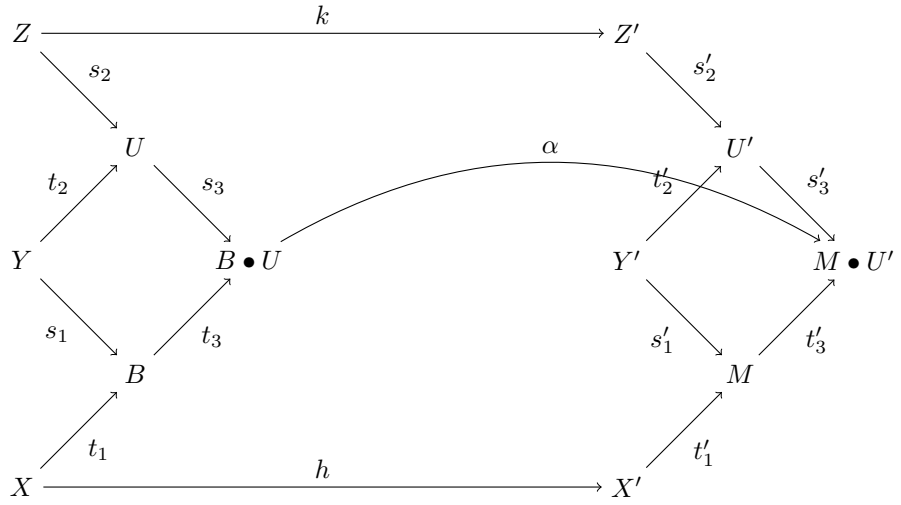
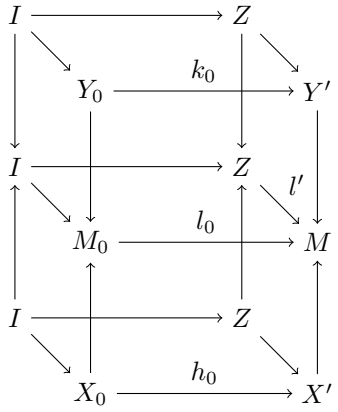
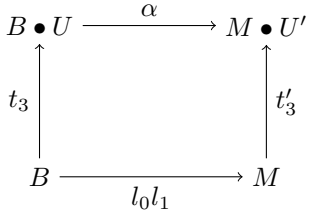


Figure 1: The double cell from lemma 3 in terms of cospans.



where $Y_0 \to M_0 \leftarrow X_0$ is a seed, I and Z are interfaces, the horizontal squares are pullbacks and collectively surjetive. Since limits are computed pointwise in presheaf categories, Z is empty in dimension greater than 1, and of the aforementioned properties of horizontal squares, there is a unique b'' in M_0 such that $l_0(b'') = b'$. We define $l_1 : B \to M_0$ to be the unique natural transformation that maps b to b''. Then, by construction,



commutes (note that we didn't have a choice of where to map b to in M_0

for this square to commute, and that a morphism $B \to M$ can always be factored through M_0). Now, since $Y \to B \leftarrow X$ and $Y_0 \to M_0 \leftarrow X_0$ are seeds, there are morphisms $h_1 : X \to X_0$ and $k_1 : Y \to Y_0$ such that the corresponding diagrams commute. Therefore, we have $k_0 k_1 : Y \to Y'$. To show that $l_0 l_1 : B \to M$ is a double cell

$$\begin{array}{ccc} Y & \xrightarrow{k_0 k_1} & Y' \\ B \downarrow & \overset{l_0 l_1}{\Rightarrow} & \downarrow M \\ X & \xrightarrow[h]{} & X', \end{array}$$

it only remains to show that $h = h_0 h_1$. They are equal when composed with $t'_3 t'_1$, and since $t'_3 t'_1$ is monic – t'_3 is monic as a pushout of t'_2 monic (pushouts of monomorphisms are monomorphisms because $\widehat{\mathcal{C}}$ is adhesive), itself monic as a composition of pushout of monomorphisms –, it entails that they are equal.

Now, we need to define a double cell

$$\begin{array}{ccc} Z & \xrightarrow{k} & Z' \\ U \downarrow & \overset{\alpha_2}{\Rightarrow} & \downarrow U' \\ Y & \xrightarrow[k_0 k_1]{} & Y' \end{array}$$

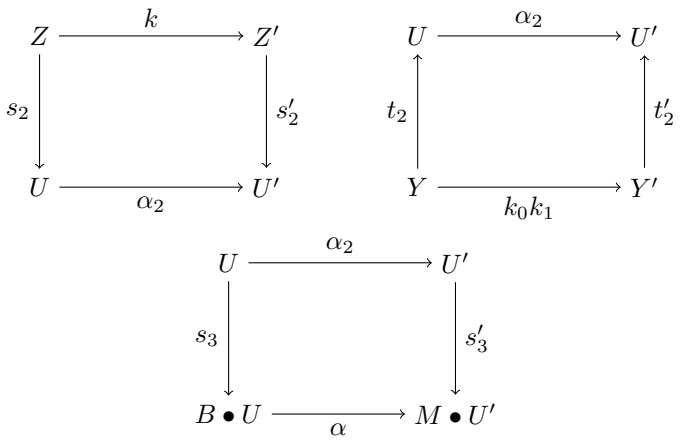
such that $l_0 l_1 \bullet \alpha_2 = \alpha$, i.e., such that

$$\begin{array}{ccc} U & \xrightarrow{\alpha_2} & U' \\ s_3 \downarrow & & \downarrow s'_3 \\ B \bullet U & \xrightarrow[\alpha]{} & M \bullet U' \end{array}$$

commutes. By lemma 1, we know that every core m in U has a path to some player y in Y in $G_{B \bullet U}$, and that this path alternates between players and cores. Now, by lemma 2, we know that such a path is transformed by α into a path from $c(\alpha(m))$ to $\alpha(y)$ in $G_{M \bullet U'}$. However, y is created by b, so $\alpha(y)$ is created by $c(\alpha(b))$, which is the core that corresponds to M, so it is impossible for a m in U to be mapped into the image of t'_2. Therefore, we know that every core m in U is mapped by αs_3 into the image of s'_3. Therefore, since s'_3 is injective on moves, we know that there is a unique m' in U' that m can be mapped to for the square to commute. Therefore, we define $\alpha_2(m)$ to be m'. For any player, channel, or move x in U that is "under" a core m in U, $\alpha_2(x)$ is defined by naturality. Now we have to define $\alpha_2(x)$ for players and channels x that are not "under"

any core m in U. However, such an x is final in U, therefore it is the image of a unique x' by s_2, and we define $\alpha_2(x)$ to be $s'_2k(x')$ (the only possible choice for the top square to commute).

We have to show that the three squares



commute. Under the assumption that the third square commutes, the first two commute because they commute when composed with s'_3, which is a monomorphism.

Now, all we need to show is that the third square commutes. Let us take $x \in U$ and show that $s'_3\alpha_2(x) = \alpha s_3(x)$:

- if x is "under" a move, then it is true because it's true for moves and by naturality;
- otherwise, x is final, so there is a unique x' in Z such that $s_2(x') = x$, and we have $\alpha_2(x) = s'_2k(x')$, therefore $s'_3\alpha_2(x) = s'_3s'_2k(x') = \alpha s_3 s_2(x') = \alpha s_3(x)$.

- $\alpha(t_3(b))$ is in the image of s'_3: the proof is similar.

□

C $\overline{T} \cdot H \subseteq H \cdot \overline{T}$

Lemma 4 (Morphisms in H)**.** *A morphism $h : G \to G'$ is in H if and only if for all vertex x in G, h restricted to edges to x is bijective onto the edges to $h(x)$ and h restricted to edges from x is bijective onto the edges from $h(x)$.*

Proof. This is just a direct reformulation of strong right orthogonality to both t_0 and s_0. □

Lemma 5 ($\overline{T}H \subseteq H\overline{T}$)**.** *Any morphism f that is decomposable into th with $t \in \overline{T}$ and $h \in H$ can be decomposed into $h't'$ with $t' \in \overline{T}$ and $h' \in H$.*

Proof. First, let us prove the more simple cases where t is a pushout of either t_0 or $\delta^0_{t_0}$.

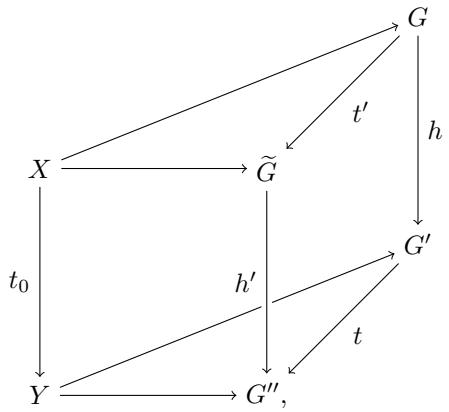
- t is a pushout of t_0: we have $G \xrightarrow{h} G' \xrightarrow{t} G''$ where $G''(v) = G'(v) \cup \{*\}$, $G''(e) = G'(e) \cup \{\rightarrow\}$, $G''(s_0) = G'(s_0) \cup \{\rightarrow \mapsto *\}$, and $G''(t_0) = G'(t_0) \cup \{\rightarrow \mapsto x\}$, where x is in $G'(v)$. Let $(x_i)_{i \in I}$ be all the antecedents of x by h. We build $\widetilde{G}$ as follows: $\widetilde{G}(v) = G(v) \cup \{*_i \mid i \in I\}$, $\widetilde{G}(e) = G(e) \cup \{\rightarrow_i \mid i \in I\}$, $\widetilde{G}(s_0) = G(s_0) \cup \{\rightarrow_i \mapsto *_i \mid i \in I\}$, and $\widetilde{G}(t_0) = G(t_0) \cup \{\rightarrow_i \mapsto x_i \mid i \in I\}$, and we define $t' : G \to \widetilde{G}$ to be the inclusion. t' is in $\overline{T}$, as a composite of pushouts of t_0. Now, we define $h' : \widetilde{G} \to G''$ as follows : $h'(*_i) = *$, $h'(\rightarrow_i) = \rightarrow$, and $h'(x) = h(x)$ otherwise. We only need to show that h' is in H, i.e., that it is strongly right orthogonal to both t_0 and s_0.

  Assume a commutative square

$$\begin{array}{ccc} X & \longrightarrow & \widetilde{G} \\ {\scriptstyle t_0}\downarrow & & \downarrow{\scriptstyle h'} \\ Y & \longrightarrow & G'', \end{array}$$

  there are two possible cases:

  - either the arrow picked in G'' is $\rightarrow$, in which case the vertex picked in $\widetilde{G}$ is an antecedent of x, i.e., one of the x_i's, so there is a unique lifting $Y \to \widetilde{G}$ that makes both triangles commute (namely, the one that picks $\rightarrow_i$);
  - or the arrow picked in G'' is already in G'. In this case, we have the following diagram:



    where the top and bottom triangles commute because the arrow picked in G'' is already in G', and therefore the vertex picked in $\widetilde{G}$ is already in G. Consequently, the square in the back commutes (since it commutes when composed with t, which is injective). Therefore, since h is strongly right orthogonal to t_0, there is a unique lifting $f : Y \to G$ such that both triangles commute, and so, by injectivity of t', $t'f : Y \to \widetilde{G}$ is the unique lifting such that both triangles commute.

The proof that h' is strongly right orthogonal to s_0 follows exactly the same pattern.

- t is a pushout of $\delta^0_{t_0}$: we have that $G \xrightarrow{h} G' \xrightarrow{t} G''$ where G' is of the form: $G'(v) = V \cup \{x^l, x^r, *\}$ (where some of the elements x^l, x^r, and $*$ may be identified), $G'(e) = E \cup \{l, r\}$, $G'(s_0) = f \cup \{l \mapsto x^l, r \mapsto x^r\}$ (where $f : E \to G'(v)$), and $G'(t_0) = g \cup \{l, r \mapsto *\}$ (where $g : E \to G'(v)$), and G'' is of the form: $G''(v) = V \cup \{x, *\}$ (where x and $*$ may be identified – they are if and only if it is forced by naturality, i.e., $x^l = *$ or $x^r = *$), $G''(e) = E \cup \{\to\}$, $G''(s_0) = f \cup \{\to \mapsto x\}$, and $G''(t_0) = g \cup \{\to \mapsto *\}$ (t maps l and r to $\to$ – and x^l and x^r to x by naturality – and is the identity on the rest of the graph).

  Let $(*_i)_{i \in I}$ be all the antecedents of $*$ by h. Since h is in H, for each $*_i$, there are two unique arrows l_i and r_i to x_i that are mapped to l and r respectively, and we define $x^l_i = l_i \cdot s$ and $x^r_i = r_i \cdot s$. Note that any edge mapped to l has $*$ as its target and is therefore an l_i (idem for edges mapped to r), and that any vertex v mapped to x^l has a unique edge from it that is mapped to l, which is therefore an l_i, so v is therefore an x^l_i (idem for x^r).

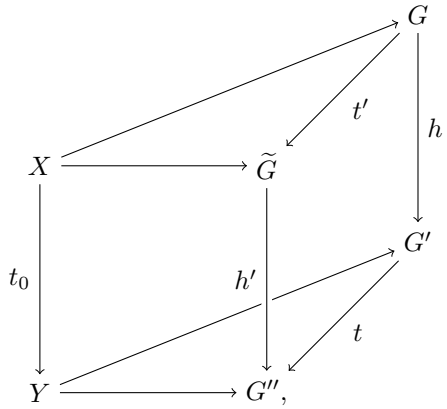
  Let us denote $V_0 = (G(v) \backslash \{x^l_i, x^r_i \mid i \in I\}) \cup \{*_i \mid i \in I\}$ (i.e., $G(v)$ where all the x^l_i and x^r_i have been erased, except if they were equal to some $*_j$ – note that if one $x^r_{i_0}$ (or $x^l_{i_0}$) is equal to some $*_{j_0}$, then any x^l_i (or x^r_i) is equal to some $*_j$) and $E_0 = G(e) \backslash \{l_i, r_i \mid i \in I\}$. Then, we define $\widetilde{G}$ as follows: $\widetilde{G}(v) = V_0 \cup \{x_i \mid i \in I\}$ (where $x_i = *_i$ if $x = *$), $\widetilde{G}(e) = E_0 \cup \{\to_i \mid i \in I\}$, $\widetilde{G}(s) = \mathcal{F}(G(s))$, and $\widetilde{G}(t) = \mathcal{F}(G(t))$, where for $f : G(e) \to G(v)$, $\mathcal{F}(f) : \widetilde{G}(e) \to \widetilde{G}(v)$ is defined by $\mathcal{F}(f) = \{x^l_i, x^r_i \mapsto x_i\} \circ f \circ \{\to_i \mapsto l_i \text{ (or } r_i) \mid i \in I\}$. We define $t' : G \to \widetilde{G}$ as $t'(x^r_i) = t'(x^l_i) = x_i$, $t'(l_i) = t'(r_i) = \to$, and the identity on the rest of the graph. t' is in $\overline{T}$, as a composite of pushouts of $\delta^0_{t_0}$. Now, we define $h' : \widetilde{G} \to G''$ as $h'(x_i) = x$, $h'(\to_i) = \to$, and $h'(\_) = h(\_)$ otherwise (note that this mapping indeed goes from $\widetilde{G}$ to G'' because no vertex that is not x^l_i may be mapped to x^l by h – idem for x^r). Now, we only need to show that h' is in H, i.e., that it is strongly right orthogonal to both t_0 and s_0.

  Assume a commutative square

$$\begin{array}{ccc} X & \longrightarrow & \widetilde{G} \\ {\scriptstyle t_0}\downarrow & & \downarrow{\scriptstyle h'} \\ Y & \longrightarrow & G'', \end{array}$$

  there are two possible cases:

  - either the arrow picked in G'' is $\to$, in which case the vertex picked in $\widetilde{G}$ is an antecedent of x, i.e., one of the x_i's, so there is a unique lifting $Y \to \widetilde{G}$ that makes both triangles commute (namely, the one that picks $\to_i$);
  - or the arrow picked in G'' is already in G'. In this case, we have the following diagram:



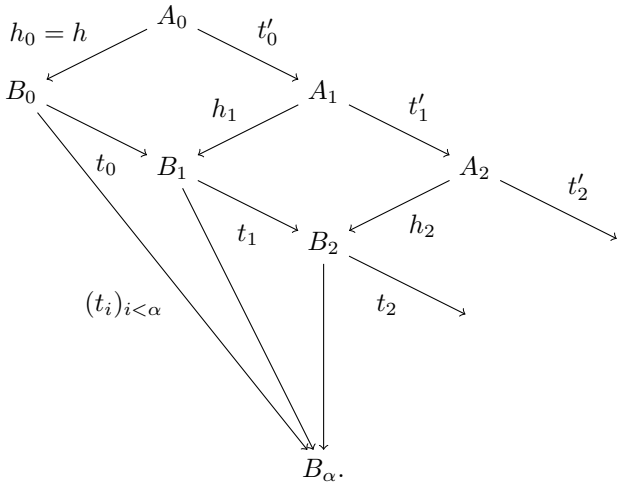
where the top and bottom triangles commute because the arrow picked in G'' is already in G' – i.e., in E –, and therefore the vertex picked in $\widetilde{G}$ is already in G – i.e., in V_0. Consequently, the square in the back commutes (since it commutes when composed with t, which is injective above V). Therefore, since h is strongly right orthogonal to t_0, there is a unique lifting $f : Y \to G$ such that both triangles commute, and that lifting is such that $f(e) \in E_0$ (where e is the only edge in Y) – otherwise, it is impossible to have $hf = Y \to G'$. Therefore, by injectivity of t' above E_0, $t'f : Y \to \widetilde{G}$ is the unique lifting such that both triangles commute.

The proof that h' is strongly right orthogonal to s_0 follows exactly the same pattern.

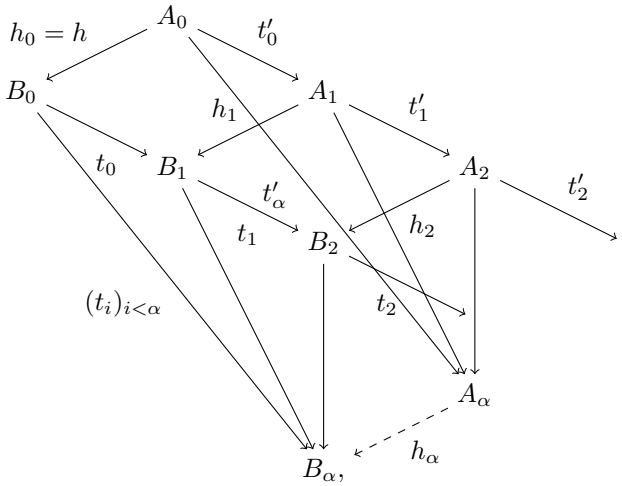
Now, let us prove the property in the general case.

t is a transfinite composite $(t_i)_{i<\lambda}$ of pushouts of t_0 and $\delta^0_{t_0}$. Let us show by transfinite induction that, for any ordinal α, we know how to factor $(t_i)_{i<\alpha}h$ as $h_\alpha(t'_i)_{i<\alpha}$:

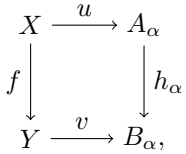
- it is true for $\alpha = 0$, since we can take $h_0 = h$;
- if α is of the form $\beta + 1$, then we have a factorization $h_\beta(t'_i)_{i<\beta} = (t_i)_{i<\beta}h$ by induction, and therefore, since we can factorize $t_\beta h_\beta$ as some $h_\alpha t'_\beta$ by the proof above, we have a factorization $h_\alpha(t'_i)_{i<\alpha}$ of $(t_i)_{i<\alpha}h$;
- otherwise, α is a limit ordinal. In this case, we have the following diagram:



We define t'_α to be the transfinite composite of $(t'_i)_{i<\alpha}$, so we have the following:



where h_α is obtained by universal property of the colimit. t'_α is in $\overline{T}$ since $\overline{T}$ is stable under transfinite composition. We only need to show that h_α is in H, i.e., that it is right orthogonal to t_0, s_0, $\delta^0_{t_0}$, and $\delta^0_{s_0}$ (not necessarily strongly orthogonal). Take a commutative square



where f is either t_0, s_0, $\delta^0_{t_0}$, or $\delta^0_{s_0}$. Since X is finitely presentable, u factors through A_n for some $n < \alpha$: $u = (t'_i)_{n \le i < \alpha} \tilde{u}$. For the same reason,

Y factors through B_m for some $m < \alpha$: $v = (t_i)_{m \leq i < \alpha} \tilde{v}$. Therefore, for $k = \max(m, n)$, we have:

$$\begin{array}{ccccc} X & \xrightarrow{(t'_i)_{n \leq i < k} \tilde{u}} & A_k & \xrightarrow{(t'_i)_{k \leq i < \alpha}} & A_\alpha \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle h_k} & & \downarrow{\scriptstyle h_\alpha} \\ Y & \xrightarrow{(t_i)_{m \leq i < k} \tilde{v}} & B_k & \xrightarrow{(t_i)_{k \leq i < \alpha}} & B_\alpha, \end{array}$$

where the left square has a lifting $d : Y \to A_k$ because h_k is in H. Therefore, $(t'_i)_{k \leq i < \alpha} d$ is a lifting for the big square.

□