A graphical foundation for interleaving in game semantics

Guy McCusker, John Power[1], Cai Wingfield[2,*]

Department of Computer Science, University of Bath, Bath, United Kingdom

Abstract

In 2007, Harmer, Hyland and Melliès gave a formal mathematical foundation for game semantics using a notion they called a $\multimap$-schedule, and the similar notion of $\otimes$-schedule, both structures describing interleavings of plays in games. Their definition was combinatorial in nature, but researchers often draw pictures when describing schedules in practice. Moreover, several proofs of key properties, such as that the composition of $\multimap$-schedules is associative, involve cumbersome combinatorial detail, whereas in terms of pictures the proof is straightforward, reflecting the geometry of the plane. Here, we give a geometric formulation of $\multimap$-schedules and $\otimes$-schedules, prove that they are isomorphic to Harmer et al.'s definitions, and illustrate their value by giving such geometric proofs. Harmer et al.'s notions may be combined to describe plays in multi-component games, and researchers have similarly developed intuitive graphical representations of plays in these games. We give a characterisation of these diagrams and explicitly describe how they relate to the underlying schedules, finally using this relation to provide new, intuitive proofs of key categorical properties.

Keywords: game semantics, geometry, schedules, composites, associativity, symmetric monoidal closed category

1. Introduction

The use of sketches and pictures to represent abstract structures is common in mathematics [1, 2, 3, 4, 5, 6]. Such planar diagrammatic methods are often employed in an informal setting, such as notes, slides and conversations, where their correspondence to the abstract structure is not made precise, instead being left as suggestive or "obvious". This is often very productive, with geometric arrangements in the plane successfully capturing some essential aspects of the abstract structure, allowing researchers to "see", for example, a transformation of one expression into another with greater ease than when using a symbolic or combinatorial representation. Certain facts about the structures can become "obvious", where in the combinatorial setting there may be much bookwork required to demonstrate the same facts.

This observation motivates further enquiry in two ways. First, the ability of planar geometry to encode abstract structure in such a way as to remove a great deal of bureaucracy is of interest in and of itself. Second, if the correspondence between the diagrammatic methods and the combinatorial structures *can* be made precise, it would permit the use of such methods in formal proofs, without any loss of rigour.

*Corresponding author.

Email addresses: g.a.mccusker@bath.ac.uk (Guy McCusker), a.j.power@bath.ac.uk (John Power), c.a.j.wingfield@bath.ac.uk (Cai Wingfield)

[1]This author would like to acknowledge EPSRC grant EP/K028243/1 "Coalgebraic Logic for Type Inference" and Royal Society grant IE120596 "Universal Algebra and its dual: monads and comonads, Lawvere theories and what?".

[2]This author would like to acknowledge EPSRC PhD funding which made this work possible.

Over recent decades, game semantics has become one of the standard forms of semantics for programming languages [7, 8, 9, 10, 11]. In 2007, Harmer, Hyland and Melliès gave a formal mathematical foundation for game semantics [12]. Their central construct was that of $\multimap$-*scheduling function* (or *schedule*), a combinatorial device which describes an interleaving of plays; a position in the game $A \multimap B$ is given by a position in A, a position in B and a schedule encoding a merge of those positions. Since the semantics of games relies on the ability to compose plays, when games are given in terms of $\multimap$-schedules, an understanding of these schedules and in particular how they may be composed is crucial.

Formally, schedules are defined to be functions $e : \{1, \ldots, n\} \to \{0, 1\}$ with the conditions that $e(1) = 1$ and $e(2k + 1) = e(2k)$. Thus, a schedule e is essentially a binary string of length n, where the domain of e indexes the string left-to-right and where `1`s and `0`s come in pairs after the first `1` (see Section 2.1). For examples, `1001111001` and `1001100001` are schedules $\{1, \ldots, 10\} \to \{0, 1\}$.

Researchers typically describe schedules on the page or blackboard using a graphical representation [13, 14, 15, 10]. These graphical schedules are plane graphs which use the vertical position of nodes to represent the sequence of moves in a game, and the horizontal position of nodes to represent the interleaving of play between different components of the game.

For example, Figure 1(b) has graphical representations of the above two schedules `1001111001` and `1001100001`. These figures are to be read top-to-bottom to give the sequence, with a node on the left corresponding to a `0` and a node on the right to a `1`. Another common practice is to omit the lines: a picture of a play in $A \multimap B$ will be drawn below a heading "$A \multimap B$" and have moves in A written below the "A", moves in B written below the "B", the sequential interleaving given by horizontal and vertical position, but no actual lines drawn. An example of such can be seen in Figure 1(a).

Composites are also typically described graphically, in a manner implied by the description of schedules as pairs of order relations in [12]. Figures 1(c) and 1(d) describe the composite of the two example schedules above.

Pictorially laid-out plays as in Figure 1(a) are still really schedule diagrams like those in Figure 1(b), and the graphical definition of schedules in this paper encompasses them; arguments involving their composition are essentially the same as those to be detailed. In a sense, it is the fact that lines *could* be drawn that means such pictures represent schedules.

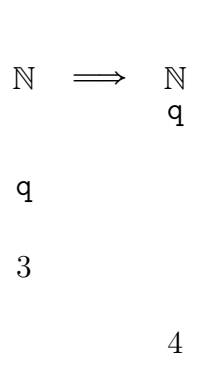
Furthermore, plays of compound games with more than two components are often laid out in a similar style, using graphs we call *interleaving graphs*. Examples of such diagrams can be seen in Figures 20(b) and 21.

This situation gives rise to several natural questions which we explore in this paper. First, in Section 2 we characterise those pictures that arise from schedules, formally prove that Harmer et al.'s combinatorial definition and our geometric definition agree, define a composite of schedules in geometric terms which also agrees with Harmer et al., and prove the associativity of this composition, thereby exhibiting a category $\mathcal{Sched}$ of schedules.

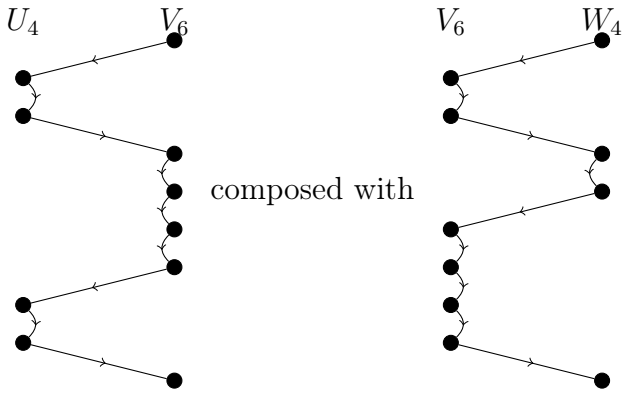
Harmer et al. also assert that composition of schedules yields a category, but they do not include a proof in [12]. A proof in Harmer et al.'s terms is combinatorially cumbersome, whereas in geometric terms it follows directly from the associativity of *juxtaposition in the plane*.

In Section 3 we augment the notion of schedules to describe games and strategies, giving a description of the linear function space $A \multimap B$ of games, and use this to construct a category of games and strategies, $\mathcal{Game}$, based on the geometric notion of $\multimap$-schedule. We also describe $\otimes$-*schedules* in similar graphical terms and use them to describe tensor games $A \otimes B$.

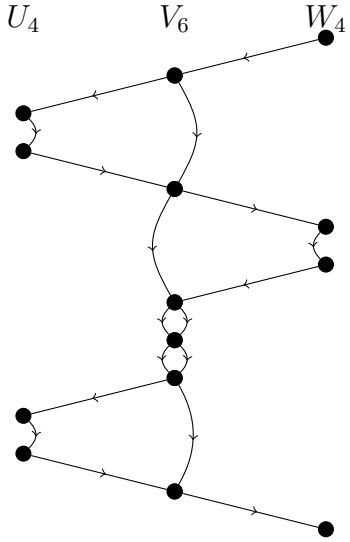
In Section 4 we generalise the notion of schedule and formalise the graphical description of interleaving of plays in games with more than two components [16, 13, 10]. We describe how these



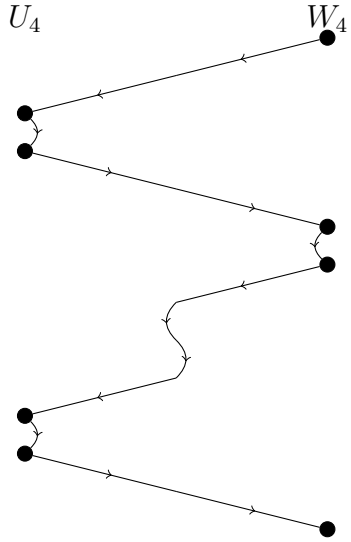
(a) A typical example of a picture representing a play of the game $\mathbb{N} \multimap \mathbb{N}$, from [13].



(b) Two examples of graphical schedules which may be composed.



(c) The intermediate *composition diagram.*



(d) The result of the composition.

Figure 1: Example schedules and schedule composition.

relate to schedules and, in Section 5, how this relationship easily demonstrates the symmetric monoidal closed structure on $\mathcal{G}ame$.

Our graphical definitions are set in the framework of Joyal and Street's string diagrams for monoidal categories [2]. One might be tempted to use an algebraic definition of graph; however in order to properly consider the diagrams drawn by researchers, one would need to consider precisely what is meant by an embedding of an algebraic graph in the plane, entailing a similar discussion to that in Sections 2.2 and 2.3, and in [2]. It is also possible to characterise schedules using the free adjunction $\mathcal{A}dj$ [17], cf. Melliès' 2-categorical string diagrams for adjunctions [18]. We plan to extend this graphical approach to encompass pointer diagrams [13, 19, 10], and in this way reformulate all of Harmer et al.'s paper in geometric terms.

2. Graphical foundations

Material in this section was originally presented in [20].

In what follows we will recall the definition of $\multimap$-schedule from [12]. We will then give a formal account of the graphical setting in which we will characterise the diagrams of $\multimap$-schedules informally introduced in Section 1. The setting we choose to work in is derived from the thorough treatment of string diagrams for monoidal categories given in [2]. Having characterised graphical $\multimap$-schedules,

we will show that this notion of schedule is equivalent to the combinatorial definition in [12], insofar as both structures form categories under composition which are shown to be isomorphic.

An understanding of $\multimap$-schedules is central to the categorical approach to game semantics of [12], as strategies for games consist of sequences of moves together with schedules describing their interleaving, and composition of strategies is based on the compositions of the underlying schedules.

2.1. Combinatorial $\multimap$-schedules

We recall the combinatorial definition of *schedules* and of composition of schedules from [12].

Definition 1 (as in Harmer et al. [12])**.** A **$\multimap$-scheduling function** is a function $e : \{1, \ldots, n\} \to \{0, 1\}$ satisfying $e(1) = 1$ and $e(2k + 1) = e(2k)$.

Schedules $e : \{1, \ldots, n\} \to \{0, 1\}$ are sequences of `0`s and `1`s. We write $|e|$ for the length n of e. We also write $|e|_0$ for the number of `0`s and $|e|_1$ for the number of `1`s in the sequence; so $|e| = n = |e|_0 + |e|_1$.

We will use the notation $e : p \to q$ when e is a schedule $\{1, \ldots, p + q\} \to \{0, 1\}$ with $|e|_0 = p$ and $|e|_1 = q$.

$\multimap$-scheduling functions are also called *schedules* in [12], but we will take care not to confuse this with Definition 9 of *schedule*, to follow. When necessary for disambiguation, we will call the latter a "graphical schedule". In Theorem 25 we will show that the notions are equivalent.

Example 2. The following are scheduling functions:

(i) `1001111001` and `1001100001` are examples illustrated in Figure 1(b).

(ii) Any nonempty prefix (or *restriction* [12]) of a schedule is a schedule.

The following definition is taken more-or-less verbatim from Harmer et al.'s paper.

Definition 3 (as in Harmer et al. [12], p. 4.)**.** We will write $\{1, \ldots, n\}^+$ and $\{1, \ldots, n\}^-$ for the sets of even and odd elements of $\{1, \ldots, n\}$ respectively. Let $e : p \to q$ be a schedule. The schedule e corresponds to a pair of order-preserving, collectively surjective embeddings

$$e_L : \{1, \ldots, p\} \hookrightarrow \{1, \ldots, p + q\}$$

$$e_R : \{1, \ldots, q\} \hookrightarrow \{1, \ldots, p + q\}$$

which can be thought of as inserting the subsequences of p `0`s and q `1`s into a sequence of length $p + q$. These in turn correspond to order relations $e_L(x) < e_R(y)$ from $\{1, \ldots, p\}^+$ to $\{1, \ldots, q\}^+$, and $e_R(y) < e_L(x)$ from $\{1, \ldots, q\}^-$ to $\{1, \ldots, p\}^-$.

We may **compose** $e : p \to q$ with a schedule $f : q \to r$, to get a schedule $f.e : p \to r$, by taking the corresponding order relations, composing them as relations and then reconstructing the $\multimap$-scheduling function on $[p + r]$.

For instance, observe that the two schedules $e =$ `1001111001` and $f =$ `1001100001` from example 2(i) may be composed since $|e|_1 = |f|_0$. Their composite is $f.e =$ `10011001`. The graphical representation of this composition can be seen in Figures 1(c) and 1(d).

Definition 4 ([12])**.** A schedule $c : p \to p$ such that $c(2k + 1) \neq c(2k + 2)$ is called a **copycat function**.

A copycat function is of the form 10011001100..., and in this sense it is the "most alternating" schedule of its length. Any nonempty prefix of a copycat function is also a copycat function.

Theorem 5 ([12])**.** *Positive natural numbers and schedules $e : p \to q$ form a category, Υ, with composition as in Definition 3, and with copycat scheduling functions as identities.*

A proof of Theorem 5 does not appear explicitly in [12], though for associativity of composition, reference is made to the merges of sketches from [21]. The theorem is certainly true, but a proof of associativity seems combinatorially cumbersome.

2.2. Graphical $\multimap$-schedules

There are several possible ways to formalise the schedule diagrams we have drawn. The framework we choose to work in is inspired by that of Joyal and Street's treatment of the graphical calculus for monoidal categories [2]. We have chosen this framework as it resembles the pictures in the literature. It is possible to give an equivalent definition to this in terms of the algebraic definition of a graph, i.e., in terms of sets V, E and functions $\mathrm{dom}, \mathrm{cod} : E \to V$. But by definition, to give an embedding of an edge, together with its domain and codomain, in the plane is exactly to give an injective continuous function from the unit interval $[0, 1]$ into the plane. When the graph has more than one edge, one needs to add conditions that ensure that the images of the edges are disjoint except at endpoints. Ultimately, we see no way to express this in simpler terms than those given by Joyal and Street, even when we restrict ourselves to graphs generated by paths.

Proofs in this framework work on the geometry of the plane graphs themselves [22]. The definition of *progressive embedding* to follow ensures that a $\multimap$-schedule graph is *compact* as a subset of $\mathbb{R}^2$. As such, all diagrams and deformations may be finitely decomposed into elementary fragments, and larger constructions may be described in terms of these fragments and their arrangement.

Definition 6 ([2])**.** A **progressive graph**, $\Gamma = (G, G_0)$, is given by:

- G, a Hausdorff space.
- $G_0 \subset G$, a finite subset such that $G \setminus G_0$ is a finite collection of **edges** e_i, each homeomorphic to the open interval $(0, 1)$. G_0 is the set of (**inner**) **nodes**. We equip each edge with a direction and disallow directed cycles.

From a progressive graph $\Gamma = (G, G_0)$ we may form $\hat{\Gamma}$, the **endpoint compactification** of Γ. $\hat{\Gamma}$ is the compactification of G achieved by affixing distinct endpoints to each edge that has fewer than two endpoints in G_0.

Definition 7. [2] For a progressive graph $\Gamma = (G, G_0)$, a **progressive embedding of Γ in the plane** is given by a continuous injection $\iota : \hat{\Gamma} \hookrightarrow \mathbb{R}^2$ such that:

- ι respects direction on edges: the "source" of an edge is "higher" than its "target" (with these words given a naïve interpretation).
- The second projection $\pi_2 : \mathbb{R}^2 \to \mathbb{R}$ is injective on each edge.

We will call a progressive graph together with such an embedding a **progressive plane graph**.

Example 8. Consider the left-hand graph of Figure 1(b). One way to characterise this graph as progressive plane graphs would be with $G = [1, 10] \subset \mathbb{R}$, $G_0 = \{1, 2, \ldots, 10\}$, and ι the obvious embedding on the page with $\iota(1)$ the node in the top right and $\iota(10)$ the node in the bottom right. (The direction on edges is not explicitly shown in this figure but may be recovered since sources are higher than targets.) Similarly, Figures 1(c), 1(d) and 24(b) are progressive plane graphs.

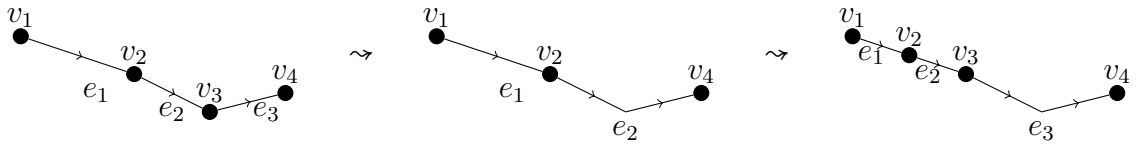


Figure 2: A node is removed, nodes and edges are relabelled; a node is added, nodes and edges are relabelled.

In this paper, our primary interest is in the progressive plane graphs that are given by directed paths, since schedule diagrams are paths (see for example Figures 1(b) and 1(d)). We will rely on a number of elementary observations about paths. First, since our paths are directed, there is an implicit *path order* on both the nodes and the edges, which we shall denote on nodes by indices on the set of nodes $\{p_1, \ldots, p_n\}$, and similarly by indices on edges: $e_i : p_i \to p_{i+1}$ is the edge with source p_i and target p_{i+1}.

Broadly speaking, composition of schedule diagrams involves the extraction of a path from a more complicated graph. One observation we will use in its definition is that graphs which are paths remain paths when we remove nodes (and glue adjoining edges) or add nodes (and split adjoining edges). In each case the relabelling of nodes by order is required (see Figure 2).

Definition 9. A **$\multimap$-schedule**, $S_{m,n} = (U, V, \Sigma, \iota)$ consists of the following data:

- Positive natural numbers m and n, identified with chosen totally ordered sets $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$. (If we wish to emphasise size, we may write these as U_m and V_n, though these sizes can be recovered from the subscripts on $S_{m,n}$.)
- A graph $\Sigma = (S, U + V)$ such that S is a path and the implicit path-ordering of nodes $U + V = \{p_1, \ldots, p_{m+n}\}$ respects the ordering of both U and V, and such that the following two conditions hold:

$$p_1 = v_1 \tag{1}$$

$$\text{for each } k, \text{ either } \{p_{2k}, p_{2k+1}\} \subset U \text{ or } \{p_{2k}, p_{2k+1}\} \subset V \tag{2}$$

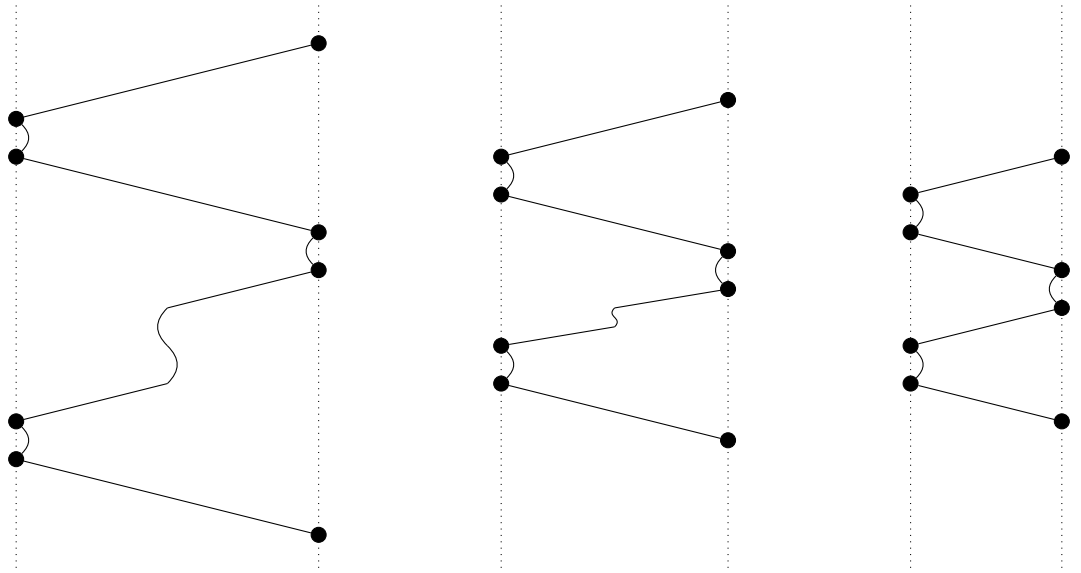
- Real numbers $u < v$ and chosen progressive embedding ι of Σ in the vertical strip of plane $[u, v] \times \mathbb{R}$ such that, (using notation $L_x := \{x\} \times \mathbb{R}$)
  - *U embeds in the left-hand edge:* $\iota(U) \subset L_u$
  - *V embeds in the right-hand edge:* $\iota(V) \subset L_v$
  - *Downwards ordering:* $j < k \implies \pi_2(\iota(p_j)) > \pi_2(\iota(p_k))$
  - *Only nodes touch edges:* $\iota(\Sigma) \cap (\{u, v\} \times \mathbb{R}) = \iota(U + V)$. Note that this condition implies that $\Sigma \setminus \Sigma_0$ is strictly contained within $(u, v) \times \mathbb{R}$.

We may write $S_{m,n} : U \to V$ when a schedule $S_{m,n}$ has sets of nodes U and V. Since the direction on the edges can always be recovered, we may safely omit the arrowheads when drawing schedules by hand.

Observe that a $\multimap$-schedule is compact as a subset of $\mathbb{R}^2$.

For examples, Figure 1(b) shows a schedule $4 \to 6$ on the left and a schedule $6 \to 4$ on the right, and Figure 1(d) shows a schedule $4 \to 4$.

We next need a notion of two schedules being "the same". Joyal and Street's framework provides a notion of *deformation* [2] which we will slightly adapt here:



(a) $t = 0$. (b) $t = 1/2$. (c) $t = 1$.

Figure 3: A "time-lapse" view of a deformation of the $\multimap$-schedule in Figure 1(d). Arrowheads used to indicate directions have been omitted for clarity.

Definition 10. Let $S_{m,n} = (U, V, \Sigma, \iota)$ and $S'_{m,n} = (U', V', \Sigma', \iota')$ be schedules with embeddings $\iota : \hat{\Sigma} \hookrightarrow [u, v] \times \mathbb{R}$ and $\iota' : \hat{\Sigma}' \hookrightarrow [u', v'] \times \mathbb{R}$ respectively.

We say that Σ **is deformable into** Σ' (**as a $\multimap$-schedule**) if there is a continuous function $h : \hat{\Sigma} \times [0, 1] \to \mathbb{R}^2$ such that

- For each $t \in [0, 1]$, $h(-, t)$ is an embedding $\hat{\Sigma} \hookrightarrow [u_t, v_t] \times \mathbb{R}$ of Σ as a schedule in the plane such that $h(U, t) \subset L_{u_t}$ and $h(V, t) \subset L_{v_t}$.
- $h(\hat{\Sigma}, 0) = \iota(\hat{\Sigma})$ is an embedding of Σ as a schedule with $h(U, 0) \subset L_u$ and $h(V, 0) \subset L_v$ and $u_0 = u$ and $v_0 = v$.
- $h(\hat{\Sigma}, 1) = \iota'(\hat{\Sigma}')$ is an embedding of Σ (also of Σ') as a schedule such that $h(U, 1) \subset L_{u'}$ and $h(V, 1) \subset L_{v'}$ and $u_1 = u'$ and $v_1 = v'$.

Then we may also say that the schedule $S_{m,n}$ is a **deformation of** the schedule $S'_{m,n}$. We call h the **deformation** and write "$S_{m,n} \sim S'_{m,n}$".

Since the deformation implies that the sets of nodes and edges in each schedule are in bijection, we may automatically associate them to give a notion of *node* and *edge* for a deformation class.

For example, looking again at the schedule in Figure 1(d), we may deform this by smoothly manipulating it in the plane, ensuring that the vertical order of nodes is not disturbed, and such that at each point in time it remains a schedule. Figure 3 shows an example of this. One might use a deformation specifically like this in the "cleaning up" of composite schedules before reuse.

Since a plane graph Γ with its plane embedding ι is trivially deformable into the graph-in-the-plane $\iota(\Gamma)$ with the identity embedding, we often identify a graph with a chosen (or arbitrary) embedding where the distinction is unnecessary. Similarly, we will often take a deformation class representative to be a graph chosen as a subset of the plane with the identity embedding.

Example 11. For any schedule, the following are examples of deformations which we will use a number of times in this paper:

- A translation of that schedule in the plane.

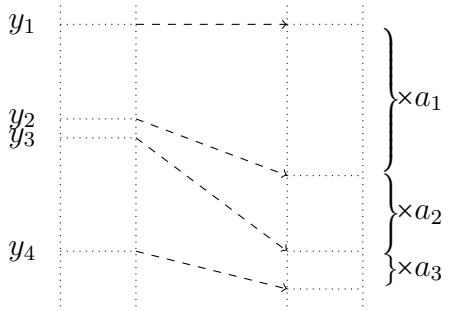


Figure 4: An illustration of a piecewise vertical scale. The strip on the left has levels y_i marked on it and the strop on the right is the result of applying the scale with factors a_i to each piece, with the dashed straight lines indicating the linear scaling of the segments.

- A horizontal or vertical scaling in the plane.
- A "piecewise" vertical scaling, achieved by dividing the plane by a finite number of horizontal lines and then applying a different scaling factor to each, as illustrated in Figure 4. This will allow us to place the nodes of a schedule wherever required without altering their order or left–right arrangement.

2.3. Composition of $\multimap$-schedules

In order to examine a category of schedules in analogue to Υ, we need a concrete description of composition of schedules. Composition of two schedules will be performed by constructing a larger progressive graph in the plane from the two components and then extracting a path from it. Essentially, the strips in which each schedule is embedded will be positioned in the plane to meet at a single vertical line. We will begin to trace a path in the right-hand component schedule, but switch to the other schedule whenever we meet it, and continue to swap back and forth whenever possible. In fact, this will give us the unique up-to-deformation path through all the nodes of both schedules, and such a path will itself be a schedule.

Definition 12. Let $S_{m,n} = (U, V, \Sigma, \iota)$ and $S'_{n,r} = (V, W, \Sigma', \iota')$ be two schedules (which we will refer to as S and S' for brevity). We first observe that a pair of translations and of piecewise vertical scalings allow us to assume that $\iota(V) = \iota'(V)$. We call a progressive plane graph formed in this way a (**2-fold**) **composition diagram** and denote it $S \cdot S'$; it has nodes $U + V + W$ and an edge for each edge in S and in S'. (We will now not differentiate between vertex sets U, V, W and their chosen embeddings $\iota(U), \iota(V) = \iota'(V), \iota'(W)$ where the context makes it clear what "$\in$" means.) Let us call any nodes not on the outside edges of a composition diagram **internal** and all other nodes **external**.

To form the **composite** of S and S', written $S \| S'$, we will extract a path from the composition diagram. Eventually our composite will have only nodes $U + W$. For now we consider nodes in V as well so that all edges have endpoints. Since S and S' are $\multimap$-schedules, all edges are progressively oriented downwards in the composition diagram. This induces an order on $U + V + W$ (their order taken top-to-bottom in the composition diagram) in which each pair of order-adjacent nodes are connected by at least one edge in S or S'. Starting from the first edge in S' we trace a path comprised of edges in S and S'. Upon reaching each external node, we take the unique outward edge from it. Upon reaching each internal node, we take the outward edge from it that lies in the other schedule from the inward edge we took. We stop when we reach a node with no outward edges. To complete the composite, we discard any edges we did not select and declassify all internal nodes, glueing together adjoining edges (as in Figure 2).

Notice that the edges removed are those comprising an extended "$\}$" shape which begins at the first node of V, continues right with an edge in S' before reaching the next node of V, where it continues left with an edge in S, and continues to alternate in this way.

This gives us $S \| S'$ as the data (U, W, P, κ) for a schedule, where P is the path formed of edges in this way and κ is the inclusion map of this path in the plane.

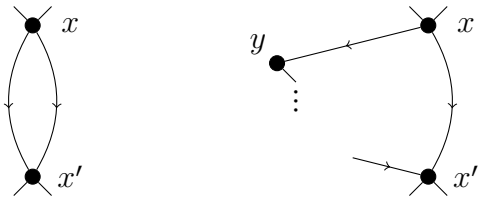
Lemma 13. *The path chosen in Definition 12 is the unique path (up to deformation) through all nodes of the composition diagram.*

Proof. Let schedules S and S' be as in Definition 12. Consider the composition diagram $S \cdot S'$. Since each component schedule is itself a path, the only nodes where we may have a choice of outward edges are the internal nodes — those shared between S and S'. At an internal node x with more than one outward edge in the composition diagram, there are two possible cases of "local picture", examples of which are shown in Figures 5(a) and 5(b).

(I) *As in Figure 5(a). Both outward edges from x lead directly to another internal node.* In this case, selecting either edge will yield the same result up to deformation.

(II) *As in Figure 5(b). One edge leads directly to another internal node x', and the other directly to an external node, y.* Suppose y is in S. We must take the edge to the external node y (the "cross-schedule" edge). To see why this is necessary, suppose we take the edge to the next internal node, x'. Since x' is an internal node, it is a node of S, and since S is itself a path through all its nodes, it will eventually reach x' from x. However, since the next node after x in S is y, y is before x' in the path order of S, and so y is above x'. Therefore, since all edges are progressive, if we take the edge directly to x' we will end up below y and so can never reach it. A similar argument applies if y is in S'.

This gives us a unique path in the composition diagram through $U + V + W$. □



(a) Order-adjacent internal nodes are connected by two edges.

(b) Order-adjacent internal nodes are connected by one edge.

Figure 5: Local pictures around internal nodes in composition diagrams

It is worthy of remark that the proofs of Lemma 13 is rather short and fundamentally graphical in nature. The well-definedness of composition of $-\circ$-schedules is a consequence and does not require any further demonstration, as are the statements of Remark 15.

Based on Lemma 13, we could have defined the composite simply as the unique (up to deformation) path through every node in the composition diagram. In case (I), where we have two edges from an internal node to another internal node, the proof of the lemma allows us to select either. However, if we decide always to select the outgoing edge on the opposite side to the incoming edge (so that we pass "through" the node), we have the property that we approach internal nodes from directions alternating right and left. This also constructs our composites in such a way that they resemble the string diagrams for adjunctions in [18].

Proposition 14. *Given schedules $S : U \to V$ and $S' : V \to W$, their composite $S \| S'$ is a schedule $U \to W$.*

Proof. The data of a schedule and conditions on the embedding follow easily from Definition 12, as does (1).

(2) simply says that once the path of a schedule diagram reaches one of the two sides, it remains for an even number of nodes before swapping. During composition, all that happens is that some internal nodes are removed, which may result in consecutive sequences of nodes on the same side being concatenated. At the start of the path, in W, this can only be a concatenation of an odd number with an even number, resulting in an odd number as required. Once the path reaches U, concatenations will be of an even number with an even number, as required. The result therefore follows by induction on the length of the schedule. □

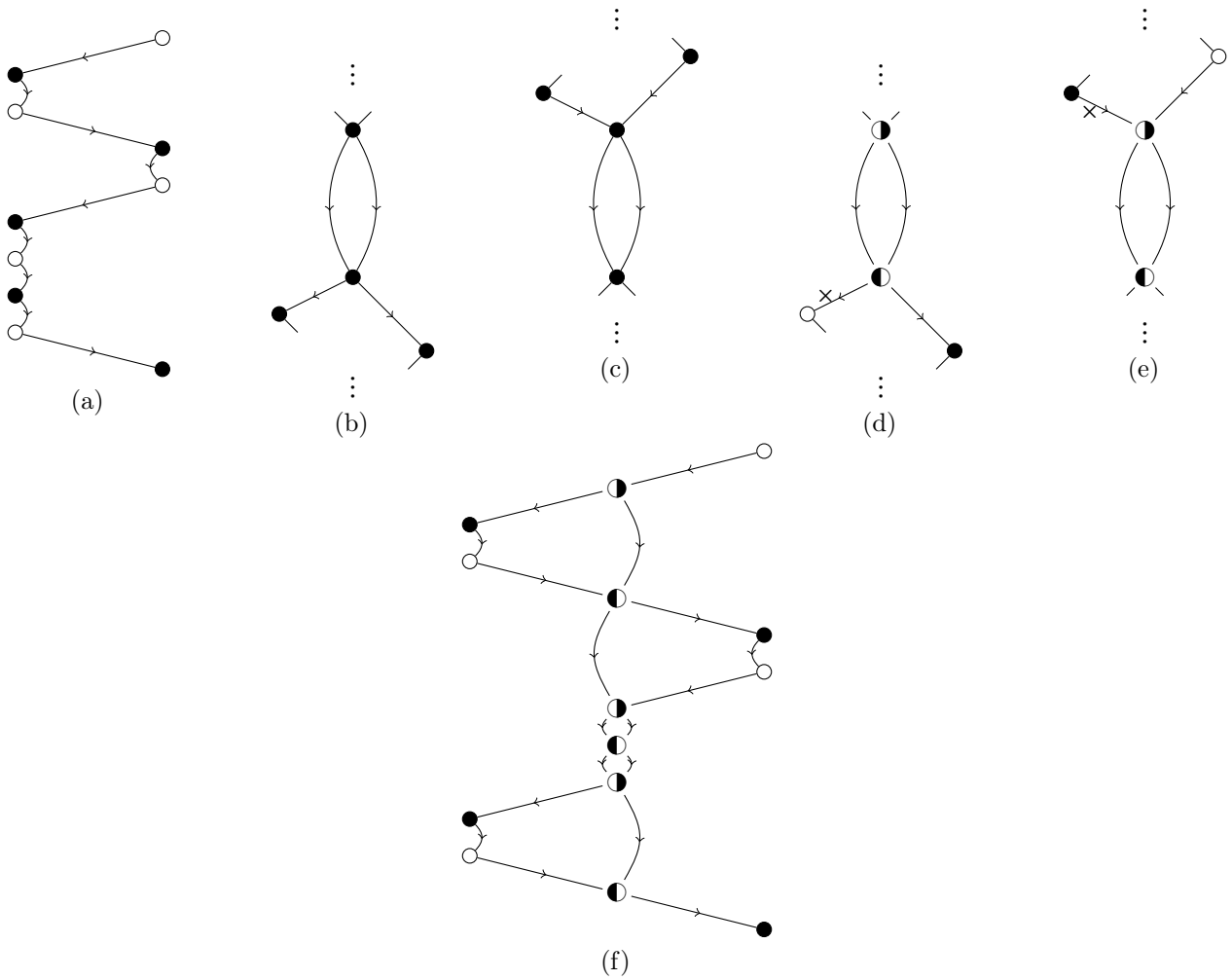


Figure 6: Colouring of nodes.

Remark 15. In the proof of Lemma 13, one might wonder why we can never have two outward edges from the same internal node, both to external nodes; or two inward edges from external nodes, both to the same internal node. Such hypothetical fragments of composition diagrams are shown in Figures 6(b) and 6(c), though in fact they can never occur.

While the reason for this may be derived from (1) and (2) by induction, there is also a "local" proof inspired by the colouring (or O/P-labelling) of nodes found in the literature [13, 10]. Observe that an arbitrary schedule $S_{m,n} : U \to V$ with path order $U + V = \{p_1, \ldots, p_{m+n}\}$ may be coloured as follows:

- v_1 coloured white (drawn as $\circ$) and u_1 black (drawn as $\bullet$).
- Nodes alternate white and black along the path order.
- Nodes in U alternate black–white taken top-to-bottom, as do those in V.

In fact, it is the case that any progressive path with nodes on either side of a vertical strip of $\mathbb{R}^2$ which is coloured in this way is a schedule. The colouring scheme encodes the "dynamics" of a schedule, as an alternative to (1) and (2), locally and in terms of colours on the nodes rather than by the explicit odd–evenness of distance from the first node. By colouring, we attach to each node its parity in its schedule.

Figure 6(a) shows our original schedule from Figure 1(b) decorated in this way. Observe that (2) is satisfied if and only if this colour scheme is followed.

Note that edges are always directed $\circ \to \bullet$ if they move from one side to the other (this is the *switching condition* for $\multimap$ [23]). Thus, if some p_i is black and p_{i+1} is white, then $\{p_i, p_{i+1}\} \subset U$ or $\subset V$. When composing schedules, the colours in the two copies of the internal nodes will be precisely reversed in each schedule. We can show this using $\circlearrowright$ and $\circlearrowleft$ for the internal nodes of the composition diagram, such as the one in Figure 6(f). Were we to have two cross-schedule edges from the same internal node, they could not both be $\circ \to \bullet$, since the internal node is different colours in both component schedules; hence such a scenario is impossible. Similarly for two cross-schedule edges to the same internal node. Figures 6(d) and 6(e) show the hypothetical fragments with a choice of colours, and the illegal edges marked with a $\times$. An analogous arguments using state diagrams exist elsewhere in the game semantics literature; for example, [23, 14].

Definition 16. Let $S : U_m \to V_n$ be a $\multimap$-schedule. The **truncation to** $j < k$ of S is the $\multimap$-schedule $S \restriction_j$ obtained by removing all parts of S strictly below the horizontal line through the j-th node along the path order of S.

Example 17. Figure 7 shows a truncated $\multimap$-schedule with the original $\multimap$-schedule in grey. Calling the original $\multimap$-schedule $S : 4 \to 8$, the truncated $\multimap$-schedules (in black) is then $S \restriction_7 : 2 \to 5$.

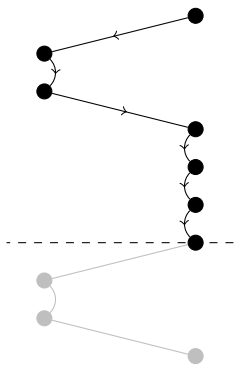


Figure 7: A truncation $S \restriction_7$ of a $\multimap$-schedule S. The dashed line shows the horizontal line through the node p_7 below which all nodes and edges are removed.

Truncation of $\multimap$-schedules interacts with composition of schedules in a convenient way. This is easily demonstrated by considering their graphs.

Proposition 18. *The truncation of a composite $\multimap$-schedule $(S \| T) \restriction_k$ is the composite of suitable truncations of S and T.*

Proof. Consider the composition diagram $S \cdot T$ and the procedure for extracting the composite. A truncation $(S \| T) \restriction_k$ is given by removing all parts of $S \| T$ below a horizontal line through its k-th node. On $S \cdot T$, this line intersects exactly one node of either S or T (whichever contains the node of $S \| T$ which it intersects) and exactly one edge of the other. Furthermore, no parts of S or T below this line contribute to the extracted path above the line, and so may be removed (along with any edges which were cut), yielding truncations of S and T which compose to give $(S \| T) \restriction_k$. □

An example of this scenario is shown in Figure 8.

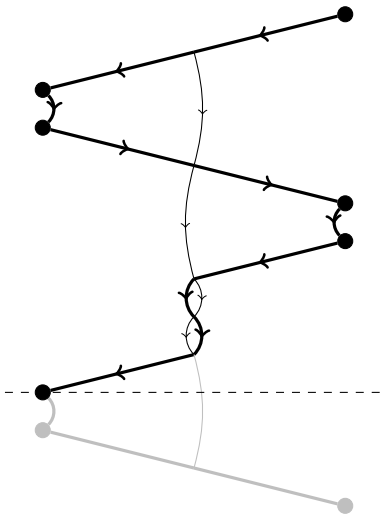


Figure 8: Truncation of a composite $\multimap$-schedule. The thick lines are edges of $S \| T$, the thin lines are edges of S and T which are not taken in the composite, the grey lines and nodes are those removed in the truncation.

2.4. The category $\mathcal{Sched}$

We now come to the key result, that of the associativity of composition. This, along with a definition of identities, will yield a description of the category of schedules.

Proposition 19. *Composition of schedules is associative.*

Proof. Suppose we are composing schedules

$$U \xrightarrow{S_{l,m}} V \xrightarrow{S'_{m,n}} W \xrightarrow{S''_{n,r}} X$$

(which we will refer to as S, S' and S'' for readability). We wish to show that $(S \| S') \| S''$ is deformable into $S \| (S' \| S'')$. Without loss of generality, we may position S, S' and S'' so that the two copies of V are identified and the two copies of W are identified. This is the **3-fold composition diagram**, an example of which can be seen in Figure 9.

$(S \| S') \| S''$ is achieved by first removing the extended "$\}$" shape through nodes in V and then removing the one through nodes in W. Dually, $S \| (S' \| S'')$ is achieved by first removing the extended "$\}$" shape through nodes in W and then the one through the nodes in V. These removals are local operations on the composition diagram and thus the results are necessarily the same. □

Alternatively, by Lemma 13, both composites $(S \| S') \| S''$ and $S \| (S' \| S'')$ are given by the unique path (up to deformation) in the 3-fold composition diagram which passes through each node $U + V + W + X$. Thus the difference in bracketing between $(S \| S') \| S''$ and $S \| (S' \| S'')$ corresponds to whether we remove unselected edges and inner nodes from V or from W first; both choices must yield the same path. In essence, associativity is due to the natural associativity of juxtaposition in the plane.

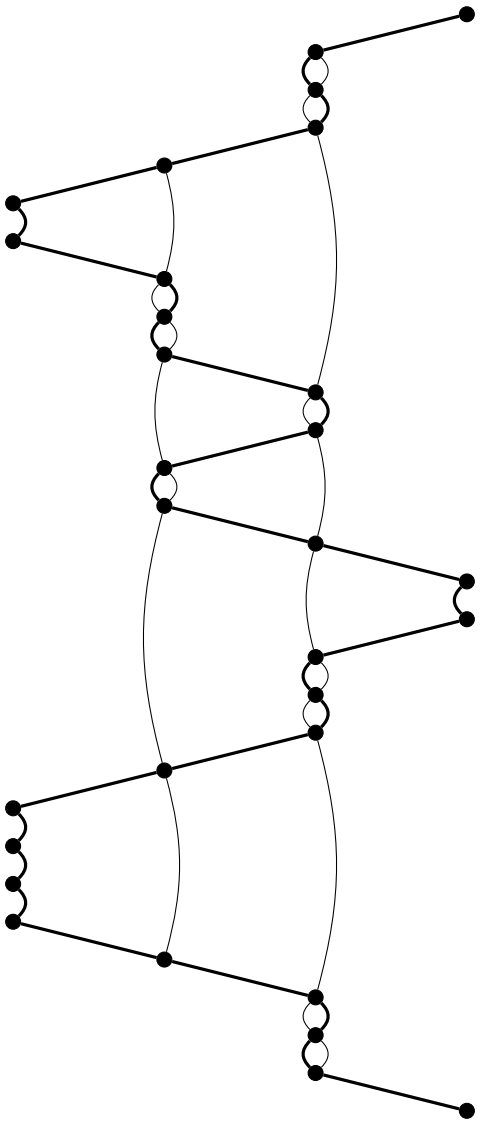


Figure 9: A three-way composition diagram with composite path highlighted. Note that, since we must always cross between schedules on reaching an internal node, there are no choices to be made.

Remark 20. The proof of Proposition 19 is not dissimilar to the proof of associativity of the graphical composition and tensor product described in [2]. In both cases, the associativity of a graphical construction follows essentially from the natural associativity inherent in the geometry of plane graphs. By contrast with a more combinatorial setting, where many reindexing lemmata would need to be proved, by using the setting of the plane, we get these ancillary facts (such as "left of left is left") for free.

We now proceed to examine the category of schedules. The objects of this category are natural numbers $m \in \mathbb{N}^+$, realised as finite indexed sets $U = \{u_1, \ldots, u_m\}$. A morphism $m \to n$ is a deformation-class of schedules $S_{m,n} : U \to V$.

Definition 21. Copycat schedules are the "most alternating" schedules possible subject to the schedule axioms. For $n \in \mathbb{N}^+$, the schedule $I_{n,n}$ may be given by its path description on vertex set $P_{2n} = U'_n + U_n$.

$$p_{4k+1} = u_{2k+1}, \quad p_{4k+2} = u'_{2k+1},$$
$$p_{4k+3} = u'_{2k+2}, \quad p_{4k+4} = u_{2k+2}$$

Graphically, this can be seen in Figure 10. Alternatively, these copycat schedules may be characterised by saying that also $\{p_{2k+1}, p_{2k+2}\}$ is not a subset of either U_n or U'_n.

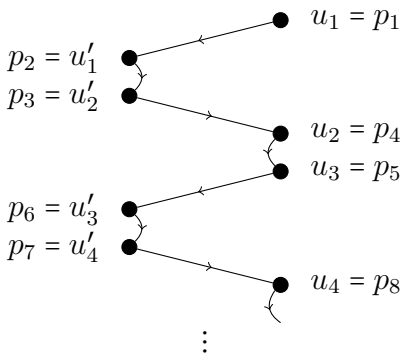


Figure 10: A prefix fragment of an identity schedule.

The following lemma is proved in Appendix Appendix A.1.

Lemma 22. *Copycat schedules $I_{n,n}$ are the identities of schedule composition.*

Theorem 23. *Positive natural numbers, together with the graphical schedules up to deformation form a category, called $\mathcal{Sched}$, where composition is defined by Definition 12 and identities are copycat schedules.*

We will demonstrate that $\mathcal{Sched}$ is isomorphic to Υ by exhibiting a functor $\mathcal{Sched} \to \Upsilon$ giving the isomorphism. Let $S_{m,n} : U \to V$ be a schedule in $[u, v] \times \mathbb{R}$; that is, an arrow of $\mathcal{Sched}$. We construct a functor C which acts on objects as the identity and which assigns to $S_{m,n}$ a $\multimap$-schedule function $e : [m+n] \to \{0, 1\}$ with

$$e : i \mapsto \begin{cases} 0 & \text{if } p_i \in L_u \\ 1 & \text{if } p_i \in L_v \end{cases}$$

In the combinatorial terms of Harmer et al. [12], a schedule $e : m \to n$ corresponds to injections $e_L : [m] \hookrightarrow [m+n]$ and $e_R : [n] \hookrightarrow [m+n]$, which in turn correspond to order relations $e_L(x) < e_R(y)$ from $[m]^+$ to $[n]^+$ and $e_R(y) < e_L(x)$ from $[n]^-$ to $[m]^-$. Thinking in terms of diagrams, the decorations + and − correspond to the parity down each edge. Then the order relation $e_R(y) < e_L(x)$ is depicted by edges right-to-left in the diagram and the order relation from $e_L(x) < e_R(y)$ is

depicted by edges left-to-right. The parity is indicated by the colours on nodes (though they are reversed on the left side). Composition of the order relation from two schedules is exactly what is performed during the composition on diagrams. Hence, we have the following proposition:

Proposition 24. *C is a functor $\mathcal{Sched} \to \Upsilon$.*

Theorem 25. *$C : \mathcal{Sched} \to \Upsilon$ is an isomorphism of categories.*

Proof. We exhibit an identity-on-objects functor $G : \Upsilon \to \mathcal{Sched}$. G assigns to a $\multimap$-scheduling function $e : [m+n] \to \{0,1\}$ with $|e|_0 = m$ and $|e|_1 = n$, a schedule $S_{m,n} : U_m \to V_n$ in the following manner:

Nodes $p_1, \ldots, p_{m+n}$ are arranged in the vertical strip $[0,1] \times \mathbb{R}$ with coordinates $p_i = (e(i), -i)$. Order-adjacent nodes p_i, p_{i+1} are joined by a straight line if their first ordinates disagree (i.e., if $\pi_1 p_i \neq p i_1 p_{i+1}$) and with a circular arc (of angle less than π) if their first ordinates agree (i.e., if $\pi_1 p_i = p i_1 p_{i+1}$). This manner is similar to the explicit construction of identity schedules in Appendix Appendix A.1.

$CG = \mathrm{id}$ by construction. To see that $GC = \mathrm{id}$, we need to show that schedule is determined up-to-deformation by the vertical order and left–right arrangement of nodes. By an appropriate piecewise vertical scale, translation and horizontal scale, we may assume that nodes are arranged according to their path-order at integer heights (as would be the case in the image of GC). So, by looking at the simply connected rectangles $[0,1] \times [i, i+1]$, we see that endpoint-preserving homotopies allow edges within these rectangles to be deformed into each other. □

3. Games and strategies

The core notion in game semantics is that of a *game*, a forest of *moves* which may be played alternately by two players, O ("opponent") and P ("player"), with O and P often being seen to model a computational environment and program respectively [16, 13, 24, 19].

In this section we will describe a category of games and strategies, $\mathcal{Game}$, which is analogous to the category $\mathcal{G}$ of [12] in that a game $A \multimap B$ is built on a collection of *labelled* $\multimap$-schedules, and strategies on such games are given by prefix-closed, *deterministic* sub-collections. The difference being that $\mathcal{Game}$ uses graphical $\multimap$-schedules where $\mathcal{G}$ uses combinatorial $\multimap$-scheduling functions. We will further show that $\mathcal{Game}$ is isomorphic to $\mathcal{G}$.

3.1. Games

We adapt our definition of *game* from Definition 3 of [12] in order that it more naturally fit with later use of $\multimap$-schedules.

Definition 26. A **game** A is given by a graded set with **predecessor function** (or **parent function**) π_A as in the diagram

$$A(1) \xleftarrow{\pi} A(2) \xleftarrow{\pi} \cdots$$

(with subscript dropped unless necessary to disambiguate). We may also use A to refer to the union of all the $A(i)$ (which we assume to be disjoint), and call the elements of A **positions** or **moves**. A move in $A(n)$ is called an **O-position** (indicating that O has just moved) if n is odd and a **P-position** if n is even. Elements of $A(1)$ are called **initial** positions and elements of $A \setminus \pi(A)$ are called **leaf** positions.

For positions a and $\pi(a)$, we say that $\pi(a)$ is the position **preceding** a and that a is a **successor** of $\pi(a)$.

A position $a \in A(n)$ determines a **(partial) play** of A, which is given by a sequence

$$\underline{a} = (\pi^{n-1}(a), \pi^{n-2}(a), \ldots, \pi(a), a) \tag{3}$$

A play $\underline{a}$ may be **restricted** or **truncated** to a play $\underline{a} \restriction_m$ consisting of the first m positions of $\underline{a}$. A play $\underline{a}$ is called **complete** if a is a leaf position.

In game semantics modelling the interaction of a program in its environment, games can be thought of as modelling the types of the interaction [23].

Remark 27. A game A can be thought of as a forest of directed, rooted trees, with the directed tree structure given by π and initial positions as roots. (Cf. alternative definitions of *game* as explicitly tree-like, e.g. [23] pp. 3–4.) A play $\underline{a}$ is a path from a root to the node a. The necessary alternation of parity in the sequence $\underline{a}$ reflects the alternating of opponent/player moves.

As can be seen in Example 28, we may completely recover a game A from its labelled forest. Initial positions are roots, elements of $A(k)$ are those nodes at depth k and π assigns to each node its parent.

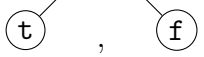
It is also worth noting that a game's forest is completely determined by its set of complete plays. There is an obvious isomorphism between a game's forest and the forest given by prefixes of complete plays. The isomorphism is the one that identifies a node with the list of moves along the path to it from an initial position. This characterisation comes more clearly into line with other definitions of game trees [23, 16].

Example 28. The game $\mathbb{B}$, which can be thought of as modelling the type of boolean truth values, is given by

$$\mathbb{B}(1) = \{\mathtt{q}\} \qquad \mathbb{B}(2) = \{\mathtt{t}, \mathtt{f}\}$$

with $\pi : \mathtt{t} \mapsto \mathtt{q}$ and $\pi : \mathtt{f} \mapsto \mathtt{q}$. We can draw explicitly as

$\mathbb{B}(1) = \{$  $\}$

$\mathbb{B}(2) = \{$  $\}$

The grade indications in this figure are superfluous (see Remark 27) and will not in general be drawn.

3.2. Strategies

We similarly take our definition of *strategy* to be similar to Definition 4 of [12].

Definition 29. A **strategy** σ for a game A is given by a graded subset $\sigma(2k) \subseteq A(2k)$ satisfying

$$\pi^2(\sigma(2k+2)) \subseteq \sigma(2k) \tag{4}$$

$$x, y \in \sigma(2k) \text{ and } \pi(x) = \pi(y) \text{ then } x = y \tag{5}$$

When σ is a strategy for A, we will write $\sigma : A$.

Note that, since all positions of σ are in the even-grades of A, they are all P-moves. We can think of a strategy $\sigma : A$ as a prescriptive walk-through of A. To each O-position with which P is presented, σ provides at most one response. We can think of (4) as saying that σ is closed under double predecessor, guaranteeing that every position of σ is reachable. We can think of (5) as saying that σ is deterministic.

Remark 30. As was noted in Remark 27, a game A is given by its set of complete plays with forest structure given by prefix. In the same way, a strategy $\sigma : A$ is given by a set of even-length prefixes of complete plays with the condition that the longest common prefix of any two is of even length, with forest structure again given by prefix.

Since π is given by prefixing of a play, (4) is equivalent to the plays being of even length. Since the longest common prefix would be determined by $\pi(x) = \pi(y)$ for some moves x and y, the requirement that this only occurs on the final position of a prefix of even length is equivalent to (5).

Just as games are thought of as modelling types, strategies can be thought of as modelling terms [23].

Example 31. The term $\mathtt{t}$ of type $\mathbb{B}$ is modelled by the strategy σ given by

$$\sigma(2) = \{\mathtt{t}\}$$

As described in Remark 30, Figure 11 exhibits $\mathtt{t} : \mathbb{B}$ as a subtree of the tree for $\mathbb{B}$ given in Example 28. We show not only the nodes in σ, but also all nodes in $\pi(\sigma)$, so that the rooted paths are still plays of the game.

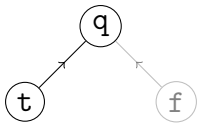


Figure 11: The tree for the strategy $\mathtt{t} : \mathbb{B}$. This tree is a subtree of the game tree for $\mathbb{B}$, which is shown in grey.

3.3. $\multimap$-schedules and linear function space

When we describe plays in an *arrow game*, we will use $\multimap$-schedules with the nodes labelled by positions from the component games. While we could follow the detailed route of [2], Definition 1.3, this is unnecessary at this point. Therefore we give the following simple definition.

Definition 32. A $\multimap$-schedule $S : U_m \to V_n$ is **labelled** in games A and B by a **labelling function**

$$l_S : U_m + V_n \to A + B$$

such that $l_S(u_i) \in A(i)$ and $l_S(v_j) \in B(j)$ for each $u_i \in U_m$ and each $v_j \in V_n$. We call $l_S(x)$ the **label** of x.

Observe that the notion of $\multimap$-schedule truncation extends naturally to the notion of labelled $\multimap$-schedule truncation: we simply truncate the underlying $\multimap$-schedule and restrict l_S to its new domain.

We indicate a labelling on a picture of a $\multimap$-schedule by writing the labels of a node inside or next to that node, as in Figure 12.

Definition 33. Given positions $a \in A(m)$ and $b \in B(n)$ and $\multimap$-schedule $S : U_m \to V_n$, the labelling l_S given by

$$l_S : u_i \mapsto \pi_A^{m-i}(a)$$
$$l_S : v_j \mapsto \pi_B^{n-j}(b)$$

is called a **play labelling**.

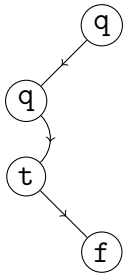


Figure 12: A $\multimap$-schedule $S : 2 \to 2$ play-labelled in the games $\mathbb{B}$ and $\mathbb{B}$.

In other words, in a play labelling we label U_m with the unique play $\underline{a}$ of A ending in a and label V_n with the unique play $\underline{b}$ of B ending in b.

Definition 34. A $\multimap$-schedule $S : U_m \to V_n$ with labelling l_S may be **composed** with a schedule $T : V'_{n'} \to W_r$ with labelling l_T to give a labelled $\multimap$-schedule $S \| T : U_m \to W_r$ with labelling $l_{S\|T}$ if the following hold:

(i) S and T are composable as $\multimap$-schedules; i.e. $n = n'$

(ii) $l_S(v_i) = l_T(v'_i)$ for all $i \leq n$

Note that if l_S and l_T are play labellings, for (ii) it suffices to show that $l_S(v_n) = l_T(v'_n)$.

The $S \| T$ is given by Definition 12, and the labelling $l_{S\|T}$, which we call the **composite labelling**, is given by

$$l_{S\|T}(u_i) = l_S(u_i) \qquad l_{S\|T}(w_j) = l_T(w_j)$$

We next consider a constructor on games, the *linear function space*, $\multimap$.

Definition 35 (Cf. *lifting of arenas* in [25].)**.** Let A be a game. The graded set $\hat{A}$ is given by the diagram

$$\hat{A}(0) \xleftarrow{\pi_{\hat{A}}} \hat{A}(1) \xleftarrow{\pi_{\hat{A}}} \hat{A}(2) \xleftarrow{\pi_{\hat{A}}} \cdots$$

with $\hat{A}(0) = \{*\}$, $\hat{A}(k) = A(k)$ for all $k > 0$ and $\pi_{\hat{A}}$ extended to send all elements of $\hat{A}(1)$ to $*$.

Definition 36. Given games A and B, we construct the **arrow game** $A \multimap B$ given by the diagram

$$(A \multimap B)(1) \xleftarrow{\pi_{A\multimap B}} (A \multimap B)(2) \xleftarrow{\pi_{A\multimap B}} \cdots$$

where $(A \multimap B)(k)$ is the set of all triples (S, a, b) with $a \in \hat{A}(m)$, $b \in B(n)$ and $S : U_m \to V_n$ is a $\multimap$-schedule with $m + n = k$ and path order $(p_1, \ldots, p_k)$. The predecessor function $\pi_{A\multimap B}$ is given by

$$\pi_{A\multimap B} : (S, a, b) \mapsto \begin{cases} (S \upharpoonright_{k-1}, \pi_A(a), b) & \text{if } p_k \in U_m \\ (S \upharpoonright_{k-1}, a, \pi_B(b)) & \text{if } p_k \in V_n \end{cases}$$

Remark 37. We will often use the notation (S, a, b) to refer to a schedule $S : m \to n$ play labelled with $a \in A(m)$ and $b \in B(n)$. We will also use the notation $(S \cdot T, a, b, c)$ to refer to the labelled composition diagram for the composition $(S, a, b) \| (T, b, c)$.

In $A \multimap B$, the positions are given by a $\multimap$-schedule and the most recent moves in the component games A and B. The triple (S, a, b) lets us draw a $\multimap$-schedule with a play labelling of U_m with $\underline{a}$ and V_n with $\underline{b}$ (both in order). Then, the predecessor function $\pi_{A\multimap B}$ maps (S, a, b) to $(S \upharpoonright_{k-1}, a', b')$, where b' is the final label on the right hand of the truncated labelled $\multimap$-schedule and a' is the final label on the left if one exists, and else is $*$. The role of the $\hat{-}$ operator is to account for the case where a move has not yet been played in one of the components. Observe that a' and b' are guaranteed to be in the correct grades of A and B respectively.

Example 38. We may give the game $\mathbb{B} \multimap \mathbb{B}$ by giving its graded sets. Rather than writing the positions as the triples (S, a, b), we rather give the labelled $\multimap$-schedules.

$$(\mathbb{B} \multimap \mathbb{B})(1) = \{\,\mathtt{q}\,\}$$

$$(\mathbb{B} \multimap \mathbb{B})(2) = \left\{ \begin{array}{ccc} \mathtt{q} & \mathtt{q} & \mathtt{q} \\ \mathtt{t} & \mathtt{f} & \mathtt{q} \end{array} \right\}$$

$$(\mathbb{B} \multimap \mathbb{B})(3) = \left\{ \begin{array}{cc} \mathtt{q} & \mathtt{q} \\ \mathtt{q} & \mathtt{q} \\ \mathtt{t} & \mathtt{f} \end{array} \right\}$$

$$(\mathbb{B} \multimap \mathbb{B})(4) = \left\{ \begin{array}{ccc} \mathtt{q} & \mathtt{q} & \\ \mathtt{q} & \mathtt{q} & \\ \mathtt{t} & \mathtt{f} & \\ \mathtt{t} & & \end{array} \right., \quad \left. \begin{array}{cc} \mathtt{q} & \mathtt{q} \\ \mathtt{q} & \mathtt{q} \\ \mathtt{f} & \mathtt{f} \\ \mathtt{t} & \mathtt{f} \end{array} \right\}$$

with $\pi_{\mathbb{B} \multimap \mathbb{B}}$ given by truncation.

Notice that, as with any game, we may completely specify the game $\mathbb{B} \multimap \mathbb{B}$ in terms of its leap positions. The labelled $\multimap$-schedules of Figure 13, together with all possible truncations of those $\multimap$-schedules, provide a complete list of the positions of $\mathbb{B} \multimap \mathbb{B}$, with grading given by the length of the $\multimap$-schedule and π given by truncation.

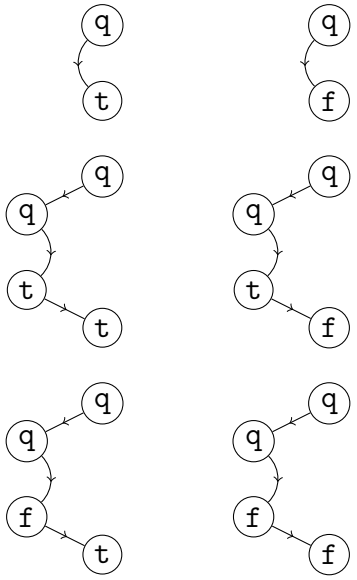


Figure 13: All complete plays for $\mathbb{B} \multimap \mathbb{B}$.

Remark 39. As described above, the entire tree of an arrow game is determined by the labelled $\multimap$-schedules for its leaf positions, with other positions and π given by prefixing. This also gives us the natural tree structure of the game.

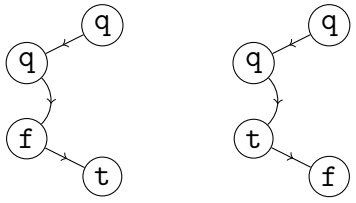
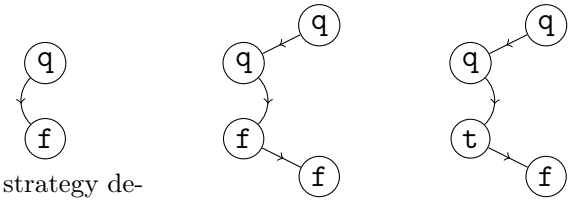


Figure 14: The $\multimap$-schedules giving all complete plays of the strategy "$\neg$".



(a) A strategy denoting the function which evaluates to `f` without examining its argument.

(b) A strategy denoting the function which evaluates to `f` after examining its argument.

Figure 15: Two strategies for the game $\mathbb{B} \multimap \mathbb{B}$.

We can therefore give a strategy σ on an arrow game by specifying a set of "maximal" even-length labelled $\multimap$-schedules whose even-length truncations give the remaining positions in σ. As in Remark 30, this set of maximal $\multimap$-schedules will subject to the condition that the longest common truncation is of any two members is of even length.

Example 40. Consider the strategy given by the $\multimap$-schedules in Figure 14. The tree given by these strategies under truncation is the subtree of the full tree for the game $\mathbb{B} \multimap \mathbb{B}$ which denotes the function $x \mapsto \neg x$.

Example 41. There are two strategies that could be considered to model the function which is constantly `f`. One ignores its argument and is shown in Figure 15(a). The other considers its arguments and is shown in Figure 15(b).

3.4. Composition of strategies

Strategies on arrow games are the morphisms of the category of games. Here, we adapt the description of strategy composition from Definition 7 of [12].

Definition 42. A strategy $\sigma : A \multimap B$ may be **composed** with a strategy $\tau : B \multimap C$ to give a strategy $\sigma \| \tau : A \multimap C$ where

$$\sigma \| \tau = \{(S \| T, a, c) \mid (S, a, b) \in \sigma \text{ and } (T, b, c) \in \tau\}$$

Lemma 43. *Let $\sigma : A \multimap B$ and $\tau : B \multimap C$ be strategies and let*

$$(S, a, b), (S', a', b') \in \sigma \qquad (T, b, c), (T', b', c') \in \tau$$

be labelled schedules. Then one of the following is true:

(i) One of $(S \| T, a, c)$ and $(S' \| T', a', c')$ is a prefix of the other.

(ii) Considered top-to-bottom, the composition diagrams $(S \cdot T, a, b, c)$ and $(S' \cdot T', a', b', c')$ first differ at some node, which we call x and x' in the respective diagrams, both A-nodes or both C-nodes, such that $S \| T \upharpoonright_x$ and $S' \| T' \upharpoonright_{x'}$ are of odd length.

Proof. Let's consider where the composition diagrams $(S \cdot T, a, b, c)$ and $(S' \cdot T', a', b', c')$ first differ. If they do not differ, or differ because one is a prefix of the other then (i) holds. If neither of these, the composition diagrams must first differ at some node which is present in both diagrams. We consider the following cases:

1. *x is an A-node and x' is a C-node, or vice versa.* This is impossible by switching condition in Remark 15.

2. *x is an A-node and x' is a B-node, or vice versa.* From the switching condition in Remark 15, we see that x, x' are both at an even-length position (they are P-moves) meaning that σ would violate (5), contrary to our assumption. A similar argument covers the case when x is a B-node and x' is a C-node, or vice versa.

3. *Both x, x' are in the same component.* We consider the following sub-cases:

   (a) *Both are in B.* Each node in B has a different parity in the left-hand and right-hand schedule. So if x, x' are both in B, one of σ and τ must violate (5).

   (b) *Both are in A or C.* Then x, x' are at an odd-length position (they are O-moves) else σ or τ would violate (5). In this case, (ii) holds.

□

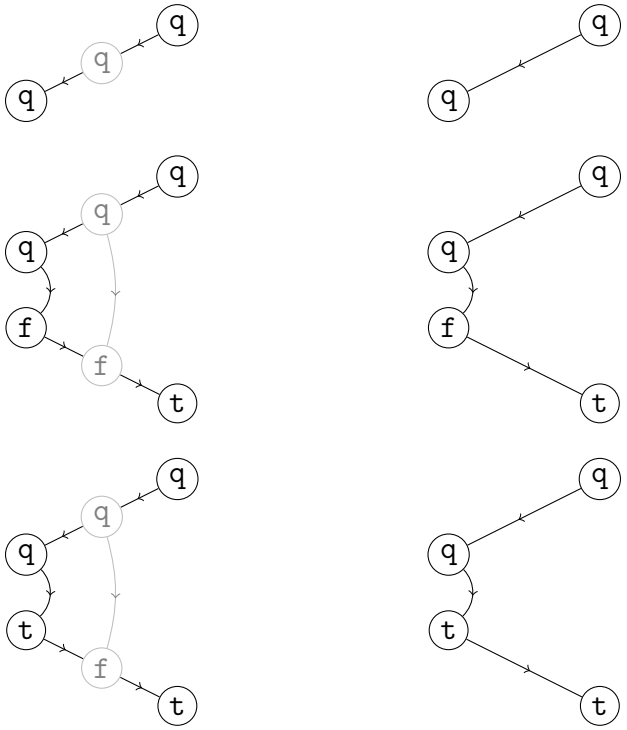
Proposition 44. *The composition of strategies in Definition 42 is well-defined. In other words, if $\sigma : A \multimap B$ and $\tau : B \multimap C$ then $\sigma \| \tau$ is a strategy on $A \multimap C$.*

Proof. $\sigma \| \tau$ consists of even-length $\multimap$-schedules as the composition of two even-length $\multimap$-schedules is of even length. It is closed under double predecessor (condition (4)) by Proposition 18, since the truncation of $S \| T$ can be seen to act on S and T, and σ and τ satisfy (4).

Lemma 43 guarantees that the longest common prefix of any two schedules in $\sigma \| \tau$ is of even length, and thus that $\sigma \| \tau$ satisfies (5). □

Example 45. The strategy denoting the function $x \mapsto \neg x$ shown in Example 40 may be composed with either of the strategies for functions which are constantly `f` given in Example 41. For example, let us compose the strategy for $x \mapsto \neg x$ with the strategy denoting the constant `f` function that examines its arguments, which we will write $x \mapsto$ `(if` x `then f else f)`. The strategy for $x \mapsto \neg x$ is given by the set of labelled $\multimap$-schedules of Figure 14 and the strategy for $x \mapsto$ `(if` x `then f else f)` is given by the set of labelled $\multimap$-schedules of Figure 15(b).

Figure 16(a) shows all possible labelled composition diagrams which may be formed from (even-length truncation of) a $\multimap$-schedule in the strategy for $x \mapsto \neg x$ on the right and a strategy for $x \mapsto$ `(if` x `then f else f)` on the left. Figure 16(b) shows the results of these compositions, and thus the labelled $\multimap$-schedules which give the strategy for $x \mapsto$ `(if` x `then t else t)`, the function which always evaluates to `t` after examining its arguments.



(a) All possible compositions of even-length truncations of $\multimap$-schedules in the strategies for $x \mapsto$ `(if` x `then f else f)` and $x \mapsto \neg x$.

(b) The results of the compositions.

Figure 16: Calculating the strategy for $x \mapsto$ `(if` x `then t else t)`

3.5. The category ***Game***

Our graphical setting, provides a straightforward proof of associativity of composition of $\multimap$-schedules, and so yields an equally immediate proof to the following proposition, for which there are notably few non-technical proofs elsewhere in the literature.

Proposition 46. *Composition of strategies is associative.*

Proof. This follows from the fact that composition of $\multimap$-schedules is associative; Proposition 19. □

Definition 47. For a game A, the **copycat** strategy $\kappa_A : A \multimap A$ is given by all positions (I, a, a) with $I : n \to n$, a copycat $\multimap$-schedule, and $a \in A(n)$.

As previously, a copycat strategy is determined by all triples (I, a, a) with $a \in A$ leaf positions and I the copycat $\multimap$-schedule of appropriate length.

Lemma 48. *Copycat strategies are identities of strategy composition.*

Proof. Suppose we compose copycat strategy $\kappa_A : A \multimap A$ with $\sigma : A \multimap B$ to form strategy $\kappa_A \| \sigma : A \multimap B$. If $\sigma : A \multimap B$ is given by positions (S, a, b), the composite $\kappa_A \| \sigma : A \multimap B$ is given by positions $(I \| S, a, b)$. By Lemma 22, copycat $\multimap$-schedules are identities of $\multimap$-schedule composition. □

Proposition 46 and Lemma 48 together provide us with the following result.

Theorem 49. *Games and strategies form a category.*

And thus we make the following definition (cf. [12], Definition 7.).

Definition 50. The **category of games** $\mathcal{G}ame$ from Theorem 49 is given as follows:

- The objects of $\mathcal{G}ame$ are games (graded sets with predecessor functions π)
- $\mathcal{G}ame(A, B) = \{\text{strategies } \sigma : A \multimap B\}$
- The identity map $A \to A$ is given by the copycat strategy $\kappa_A : A \multimap A$.
- Composition of $\sigma : A \to B$ with $\tau : B \to C$ to give $\sigma \| \tau : A \to C$ is given by composition of strategies.

Theorem 51. *The category $\mathcal{G}ame$ is isomorphic to the category of games and strategies $\mathscr{G}$ from [12], Definition 7.*

Proof. This follows from the fact that the category $\mathcal{S}ched$ of graphical $\multimap$-schedules is isomorphic to the category Υ of combinatorial $\multimap$-schedules (Theorem 25). □

3.6. $\otimes$-schedules and tensor games

Along the same lines as Definition 9, we make the following definition for a $\otimes$-*schedule*, which will describe the interleaving of plays in $\otimes$-composition of two games [23, 12]. This $\otimes$ will be shown in Proposition 83 to comprise a tensor product on $\mathcal{G}ame$.

Definition 52. A **$\otimes$-schedule**, $S^{\otimes}_{m,n} = (U, V, \Sigma, \iota)$, consists of the following data:

- Non-negative integers m and n, identified with chosen totally ordered (possibly empty) sets $U = \{u_1, \ldots, u_m\}$ and $V = \{v_1, \ldots, v_n\}$. This gives us $|U| = m$ and $|V| = n$, with U or V empty only when respectively $m = 0$ or $n = 0$.
- A progressive graph $\Sigma = (S, U + V)$ such that S is a path and the implicit path-ordering of nodes $U + V = \{p_1, \ldots, p_{m+n}\}$ respects the ordering of both U and V, and such that the following condition holds:

$$\begin{aligned} \text{for each } k \geq 0, \text{ either } \{p_{2k+1}, p_{2k+2}\} \subset U \\ \text{or } \{p_{2k+1}, p_{2k+2}\} \subset V \end{aligned} \tag{6}$$

- Real numbers $u < v$ and chosen progressive embedding ι of Σ in the vertical strip of plane $[u, v] \times \mathbb{R}$ such that, (using notation $L_x := \{x\} \times \mathbb{R}$)
  - *U embeds in the left-hand edge:* $\iota(U) \subset L_u$
  - *V embeds in the right-hand edge:* $\iota(V) \subset L_v$
  - *Downwards ordering:* $j < k \implies \pi_2(\iota(p_j)) > \pi_2(\iota(p_k))$
  - *Only nodes touch edges:* $\iota(\Sigma) \cap (\{u, v\} \times \mathbb{R}) = \iota(U + V)$. Note that this condition implies that $\Sigma \setminus \Sigma_0$ is strictly contained within $(u, v) \times \mathbb{R}$.

We may write $S : U \otimes V$ or $S : m \otimes n$ when S is such a $\otimes$-schedule.

Example 53. Figure 17 shows an example of a $\otimes$-schedule.

We may easily expand Definitions 16 and 32 to describe the *labelling* and *truncation* of $\otimes$-schedules.

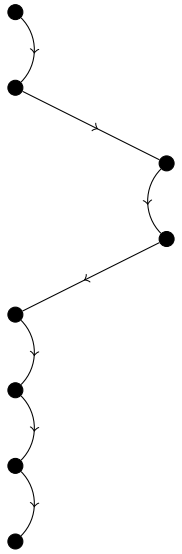


Figure 17: A $\otimes$-schedule $S^{\otimes}_{6,2}$.

Definition 54. Given games A and B, we construct the **tensor game** $A \otimes B$ given by the diagram

$$(A \otimes B)(1) \xleftarrow{\pi_{A\otimes B}} (A \otimes B)(2) \xleftarrow{\pi_{A\otimes B}} \cdots$$

where $(A \otimes B)(k)$ is the set of triples $(S^{\otimes}, a, b)$ with $a \in \hat{A}(m)$, $b \in \hat{B}(n)$ and $S^{\otimes}$ a $\otimes$-schedule with $m + n = k$ and path order $\{p_1, \ldots, p_k\}$. The predecessor function $\pi_{A\otimes B}$ is given by

$$\pi_{A\otimes B} : (S^{\otimes}, a, b) \mapsto \begin{cases} (S^{\otimes} \restriction_{k-1}, \pi_{\hat{A}}(a), b) & \text{if } p_k \in U_m \\ (S^{\otimes} \restriction_{k-1}, a, \pi_{\hat{B}}(b)) & \text{if } p_k \in V_n \end{cases}$$

Example 55. The game $\mathbb{B} \otimes \mathbb{B}$ is given by its graded sets which, in a similar manner to Example 38, we may specify entirely by the leaves of the game tree, which are labelled schedules. Figure 18 shows all complete plays for $\mathbb{B} \otimes \mathbb{B}$, with $\pi_{\mathbb{B}\otimes\mathbb{B}}$ given by truncation.

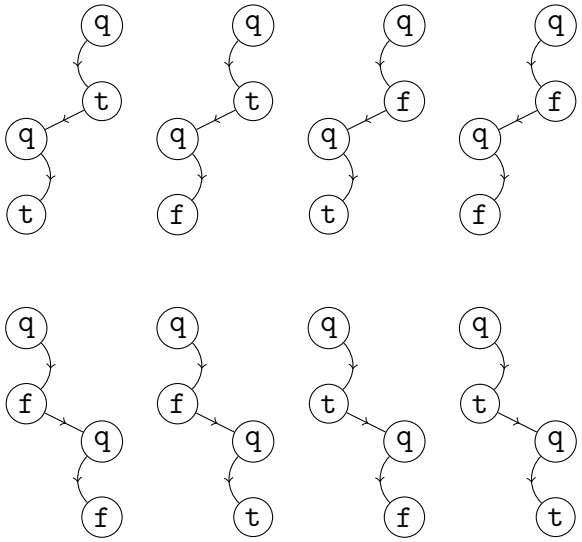


Figure 18: All complete plays for $\mathbb{B} \otimes \mathbb{B}$.

From Remark 30 we may give a strategy on a game A as a set of truncations of complete plays of A. From Remark 39 we see how to interpret this notion in a game $A \multimap B$ where plays are given by

labelled $\multimap$-schedules. There is a similar notion in a game $A \otimes B$ where plays are given by labelled $\otimes$-schedules which are even-length truncations of those labelled $\otimes$-schedules giving complete plays, subject to the condition that the longest common truncation of any two is of even length.

Example 56. The strategy $\mathtt{t} \otimes \mathtt{t} : B \otimes B$ is given by the labelled schedules in Figure 19.

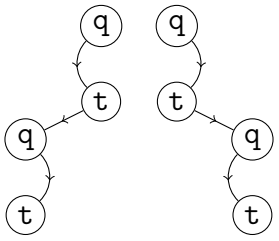


Figure 19: The schedules giving all complete plays of the strategy $\mathtt{t} \otimes \mathtt{t} : \mathbb{B} \otimes \mathbb{B}$.

4. Graphical representations of games with more than two components

In this section we examine a second class of graphs, describing plays of games with multiple components, which are already in informal use by researchers. This expanded class of graphs turns out to have use in proving certain categorical properties of $\mathcal{Game}$, and the work in this section sets up the infrastructure necessary for Section 5 and Theorem 86.

So far, the $\multimap$-schedules and $\otimes$-schedules used to describe the interleaving of plays in games only describe the interleaving across two components, such as in a game $A \multimap B$ or $A \otimes B$. If a game has a more complex structure, the plays are more complicated.

For example, consider the game $A \multimap (B \otimes C)$. A position of $A \multimap (B \otimes C)$ is a triple (S, a, x), where S is a $\multimap$-schedule, a is a position of $\hat{A}$ and x is a position of $B \otimes C$. Therefore, x is itself a triple (T, b, c), with T a $\otimes$-schedule, b a position of $\hat{B}$ and c a position of $\hat{C}$. One could express a position of $A \multimap (B \otimes C)$ as a $\multimap$-schedule whose nodes on the left are labelled $(\ldots, \pi_A(a), a)$ and whose nodes on the right are labelled with $(\ldots, \pi_{B \otimes C}(T, b, c), (T, b, c))$.

An example of such a position is shown in Figure 20(a). The enlarged nodes on the right are labelled with $\otimes$-schedules which show the play in $B \otimes C$.

However, this representation does not reflect the intuitive arguments used by researchers who frequently argue about multi-component games in similar ways to two-component games [16, 13, 10]. Rather than nested interleavings being recorded within nodes' labels, graphs can directly show play passing between several component games. In our example, this may be depicted by like Figure 20(b). These "unfolded" expressions are widely used and provide intuitive, informal arguments for structure on categories of games.

In this section we classify the collection of such graphs, explain how to use them to represent positions of games, and demonstrate that games described in this way are isomorphic to the games presented in the "folded" or "nested" style.

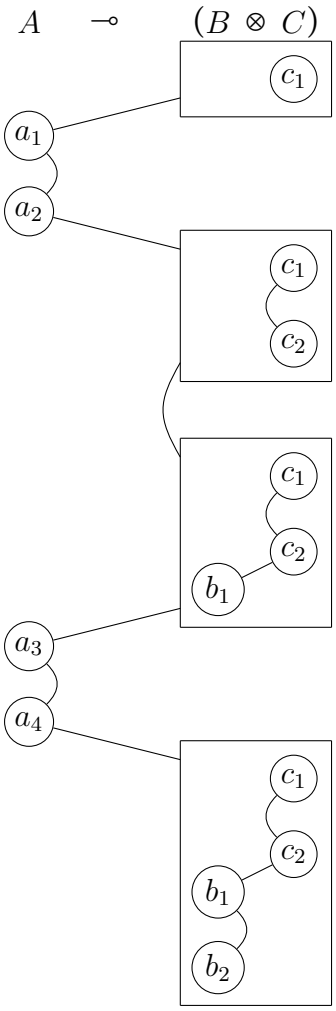
4.1. Interleaving graphs

We begin, as we did with defining schedules, with unlabelled graphs.

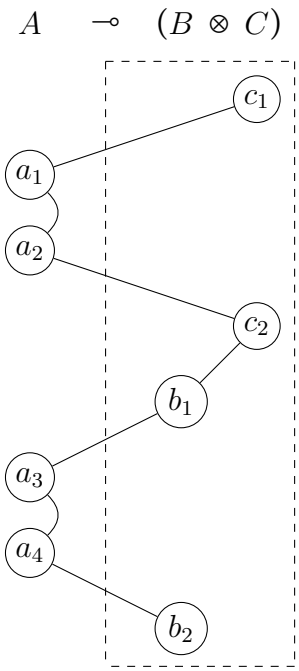
Definition 57. An n**-interleaving graph** is a list

$$S_{m_1,\ldots,m_n} = (U^{(1)}, \ldots, U^{(n)}, \Sigma, \iota)$$

consisting of the following data:



(a) An example of a play of some game $A \multimap (B \otimes C)$.



(b) The same play unfolded into a labelled 3-interleaving graph

Figure 20: Two representations of the same play.

- Each m_i is a non-negative integer associated with a chosen totally ordered (possibly empty) set $U^{(i)}$ of size m_i.
- A graph $\Sigma = (S, U^{(1)} + \cdots + U^{(n)})$ such that S is a path and the implicit path-ordering of nodes $U^{(1)} + \cdots + U^{(n)} = \{p_1, \ldots, p_{m_1+\cdots+m_n}\}$ respects the ordering of each of the $U^{(i)}$.
- Real numbers $u_1 < \cdots < u_n$ and a chosen progressive embedding of Σ in a horizontally bounded vertical strip of $\mathbb{R}^2$ such that
  - For each i, $\iota(U^{(i)}) \subset L_{u_i}$.
  - Downwards ordering: $j < k \implies \pi_2(\iota(p_j)) > \pi_2(\iota(p_k))$.

We may refer to an **interleaving graph** when we do not wish to specify n. The notions of *truncation* and *labelling* for schedules may easily be extended to interleaving graphs.

Example 58. The following are examples of n-interleaving graphs.

- Figure 21 shows an example of an 4-interleaving graph.
- All $\multimap$-schedules and $\otimes$-schedules are examples of 2-interleaving graphs.

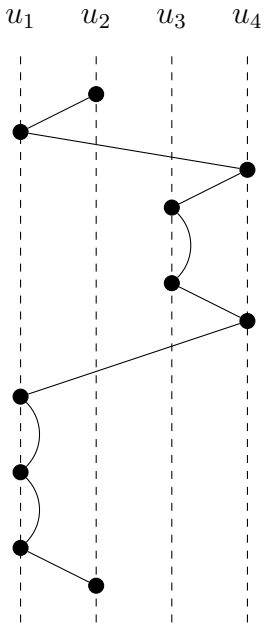


Figure 21: An example of a 4-interleaving graph.

We will frequently describe interleaving graphs "up to deformation" in a way similar to Definition 10. However, there are two distinct notions of *deformation* we will use, one more general than the other. In some cases we wish to refer to a deformation of interleaving graphs *as interleaving graphs*, so that the vertical order of the nodes remains unchanged and so that any nodes on the same vertical line remain on a vertical line. For this we will use the notion of deformation from Definition 60. In the more general case, when we are manipulating the structure of interleaving graphs in Section 4.3, we will wish to deform the graphs in such a way that nodes formerly on the same vertical line can move horizontally relative to each other. For this we will need a more generalised notion of *deformation*, as in Definition 59.

Definition 59 ([2], Definition 1.2.)**.** Let $\Gamma = (G, G_0)$ and $\Gamma' = (G', G'_0)$ be progressive graphs, together with progressive plane embeddings $\iota : \hat{\Gamma} \hookrightarrow \mathbb{R}^2$ and $\iota' : \hat{\Gamma'} \hookrightarrow \mathbb{R}^2$ respectively. We say that Γ with ι **is deformable into** Γ' with ι' (**as a progressive plane graph**) if there is a continuous function $h : \hat{\Gamma} \times [0, 1] \to \mathbb{R}^2$ such that

- For each $t \in [0,1]$, $h(-,t)$ is a progressive plane embedding $\Gamma \hookrightarrow \mathbb{R}^2$.
- $h(\Gamma, 0) = \iota(\Gamma)$ is a progressive plane embedding of Γ.
- $h(\Gamma, 1) = \iota'(\Gamma')$ is a progressive plane embedding of Γ (and of Γ').

In particular, we consider restricted notions of deformations, such as in Definition 10 and in the following:

Definition 60. Let $S_{m_1,\dots,m_n} = (U^{(1)}_{m_1}, \dots, U^{(n)}_{m_n}, \Sigma, \iota)$ and $S'_{m_1,\dots,m_n} = (U'^{(1)}_{m_1}, \dots, U'^{(n)}_{m_n}, \Sigma', \iota')$ be n-interleaving graphs in the plane. We say that S **is deformable into** S' (**as an n-interleaving graph**) if there is a continuous function $h : \Sigma \times [0,1] \to \mathbb{R}^2$) such that:

- For each $t \in [0,1]$, $h(-,t)$ is an embedding $\Sigma \hookrightarrow \mathbb{R}^2$ as an n-interleaving graph such that $h(U^{(i)}, t) \subset L_{u_{i,t}}$ for some $u_{1,t} < \cdots < u_{i,t} < \cdots < u_{n,t}$
- $h(\Sigma, 0) = \iota(\Sigma)$ is an embedding of Σ as an n-interleaving graph
- $h(\Sigma, 1) = \iota'(\Sigma')$ is an embedding of Σ (and of Σ') as an n-interleaving graph

Example 61.

- Translations and piecewise scalings are examples of deformations as interleaving graphs.
- Figure 22(a) shows an example of a deformation of an interleaving graph *as an interleaving graph.*
- Figure 22(b) shows an example of a deformation of an interleaving graph *as a progressive plane graph*, so that the result is also an interleaving graph.

Remark 62. In general when we speak of deformations of interleaving graph, and in particular interleaving graphs "up-to-deformation", we mean deformation *as interleaving graphs* in the sense of Definition 60, though this may go unsaid. Only in Section 4.3 and Proposition 84 will we use an expanded notion of deformation.

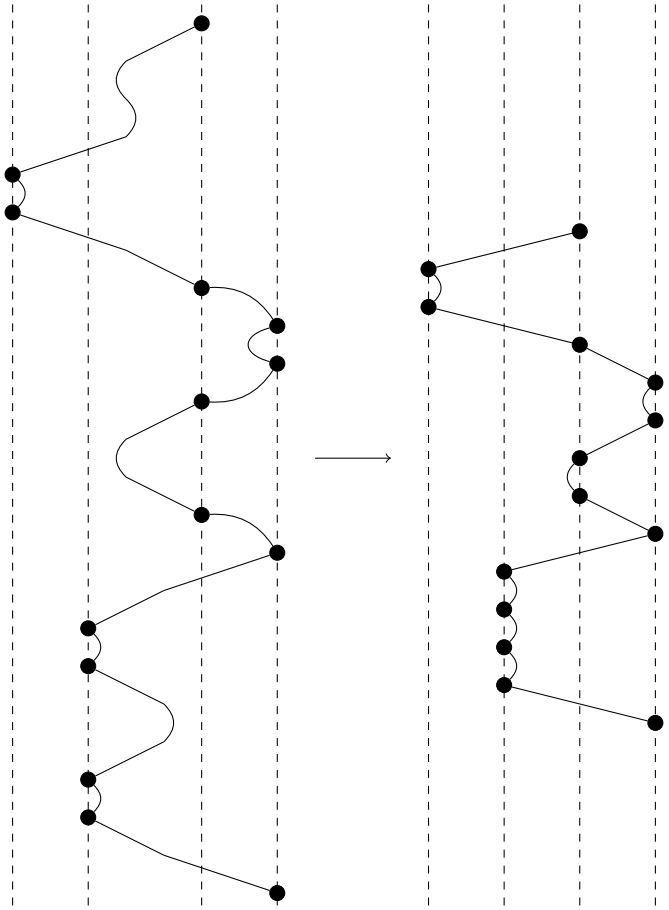
The difference is that deformation *as an n-interleaving graph* requires that at each value of t, the image is an n-interleaving graph, whereas deformation *as a progressive plane graph* permits isometries of the plane which leave an n-interleaving graph no longer an n-interleaving graph.

Thus defined, n-interleaving graphs describe diagrammatically all ways in which moves may be interleaved in an order-preserving way between n components. But not all such interleavings may occur in practice. We want to characterise those n-interleaving graphs which could describe the play of a game whose components are joined with $\otimes$ and $\multimap$. For this we first need some procedures for examining the structure of interleaving graphs.

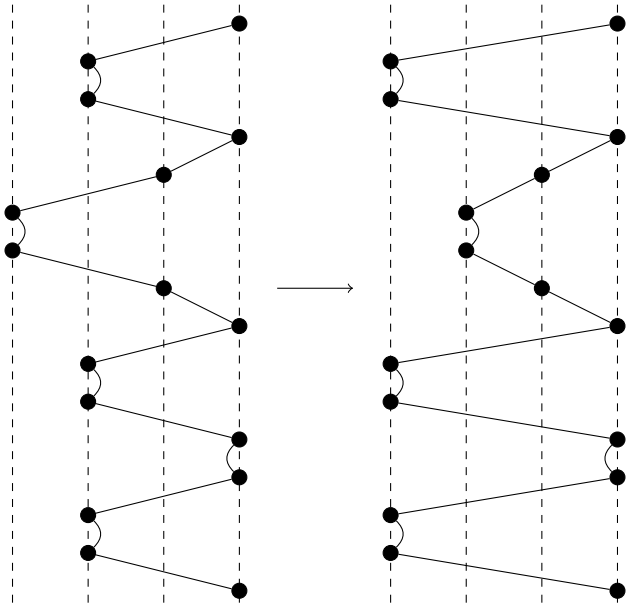
Definition 63. Let $S = (U^{(1)}, \dots, U^{(n)}, \Sigma, \iota)$ be an n-interleaving graph in $[u_1, u_n] \times \mathbb{R}$. We consider two fundamental operations on it.

1. **Collapse**: For a sequence $\{u_i, u_{i+1}, \dots, u_j\} \subset \{u_1, \dots, u_n\}$, we form an $(n-j+i+1)$-interleaving graph
$$[S]^{(u_i,u_j)} = (U^{(1)}, \dots, U^{(i)} + \cdots + U^{(j)}, \dots, U^{(n)}, \Sigma', \iota')$$
where Σ' is a deformation of Σ with embedding ι so that the nodes in $U^{(i)} + \cdots + U^{(j)}$ lie on some vertical line L_u with $u \in [u_i, u_j]$ and all other nodes coincide with their corresponding nodes in Σ.

   In essence, we pull all nodes which lie on the lines $L_{u_i}, \dots, L_{u_j}$ onto the same line L_u.



(a) Deformation as an interleaving graph. Example from Figure A.28(f).



(b) Deformation as a progressive plane graph. Example from Figure 25(a).

Figure 22: Two "deformations" of interleaving graphs.

2. **Restrict**: For a subset
$$\bar{U} = \{u_{i_1}, \ldots, u_{i_j}\} \subseteq \{u_1, \ldots, u_n\} = U$$
we form a j-interleaving graph
$$[S]_{i_1,\ldots,i_j} = (U^{(i_1)}, \ldots, U^{(i_j)}, \Sigma', \iota')$$
where Σ' is achieved from Σ by glueing edges at all nodes in $U \setminus \bar{U}$ (equivalently, declassifying nodes in $U \setminus \bar{U}$) and discarding any half-edges.

From these we derive an additional useful operation:

3. **Segregate**: Let (u_1, u_n) be the open interval of points $u_1 < x < u_n$. For a selected $u \in (u_1, u_n)$, find u_i such that $u_i < u < u_{i+1}$. Collapse nodes on $\{u_1, \ldots, u_i\}$ to u_1 and collapse nodes on $\{u_{i+1} \ldots, u_n\}$ to u_n.

Example 64. Figures 23(a), 23(b) and 23(c) show examples of the three operations on n-interleaving graphs.

We will proceed by considering the compositional structure of games, as described by a set of symbolic expressions:

Definition 65. Given a set $\{A_1, \ldots, A_n\}$ of **letter symbols**, the **binary $(\multimap, \otimes)$-words** with letter symbols $A_1, \ldots, A_n$ are those words generated by the following grammar:
$$\mathtt{W} \quad ::= \quad (\mathtt{W} \otimes \mathtt{W}) \quad | \quad (\mathtt{W} \multimap \mathtt{W}) \quad | \quad A_i$$

The symbols $\otimes$ and $\multimap$ are **connectives**. In a word $\mathtt{W}$ of the form $(\mathtt{W}_1 \multimap \mathtt{W}_2)$, the explicitly denoted occurrence of $\multimap$ is called the **major connective** (and similarly for $(\mathtt{W}_1 \otimes \mathtt{W}_2)$); $\mathtt{W}_1$ and $\mathtt{W}_2$ are **subwords** of $\mathtt{W}$; subwords of $\mathtt{W}_1$ and $\mathtt{W}_2$ are also regarded as subwords of $\mathtt{W}$. We frequently omit parentheses where no ambiguity arises.

We are now able to classify those interleaving graphs which may represent plays in games whose compositional structure is known. If a game has a structure defined by a binary $(\multimap, \otimes)$-word w, we wish our interleaving graph to exhibit the appropriate qualities when the interleaving is considered across each of w's connectives.

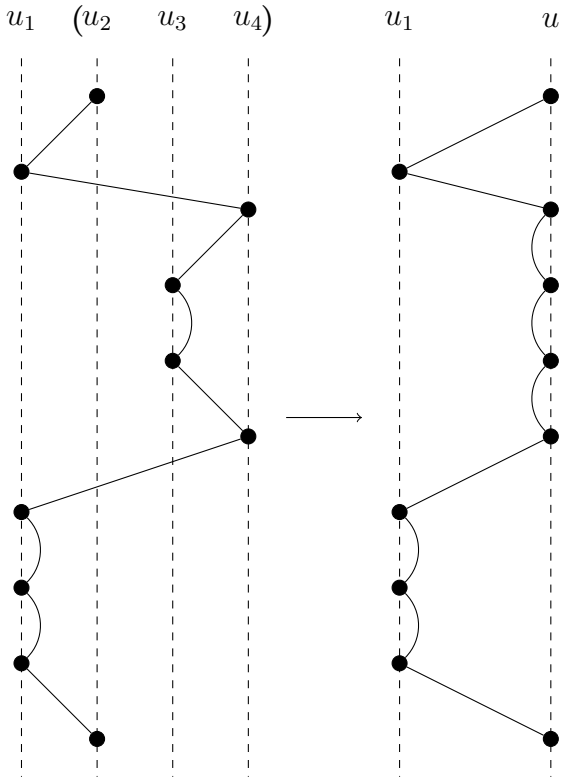
Definition 66. Let w be a binary $(\multimap, \otimes)$-word of length n with letter symbols $A_1, \ldots, A_n$, connectives $\chi_1, \ldots, \chi_{n-1}$ and parentheses. The n letter symbols of w may be identified, in order, with some points $u_1 < \cdots < u_n \in \mathbb{R}$. Let S be an n-interleaving graph on $\{u_1, \ldots, u_n\}$.

S is **suitable** for w if both:

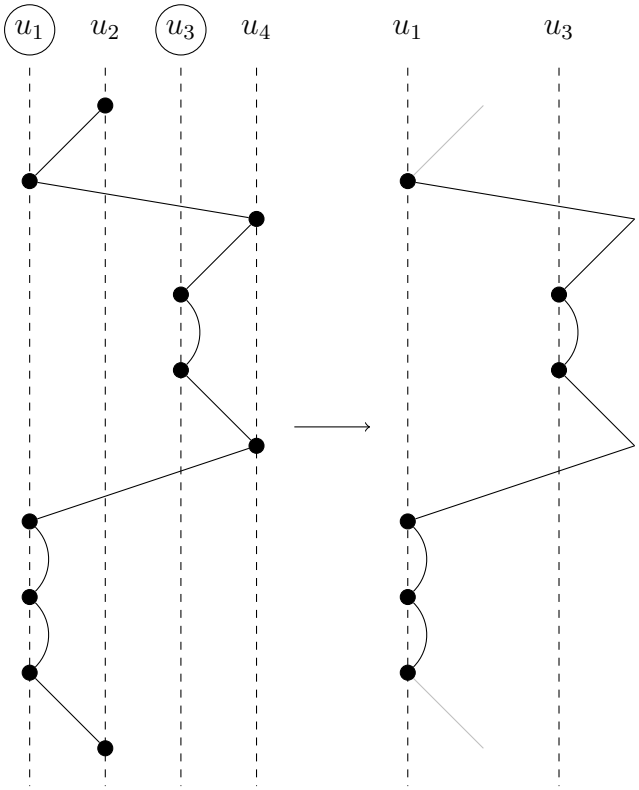
(i) For each subword of w, the corresponding restriction of S yields a graph suitable for that subword.

(ii) For each connective χ_i of w, restricting S to the subword whose major connective is χ_i and then segregating on some u strictly between the u_i, u_j corresponding to the closest letter symbols to that connective yields a 2-interleaving graph which is a schedule for that connective.

Definition 67. Let $S = (U^{(1)}, \ldots, U^{(n)}, \Sigma, \iota)$ be an n-interleaving graph suitable for a word w of length n. Let X_i be the i-th letter symbol of w. The **X_i-nodes** of S are those in $U^{(i)}$. For a subword v of w with letter symbols $X_i, \ldots, X_j$, the **v-nodes** of S are those in $U^{(i)} + \cdots U^{(j)}$.

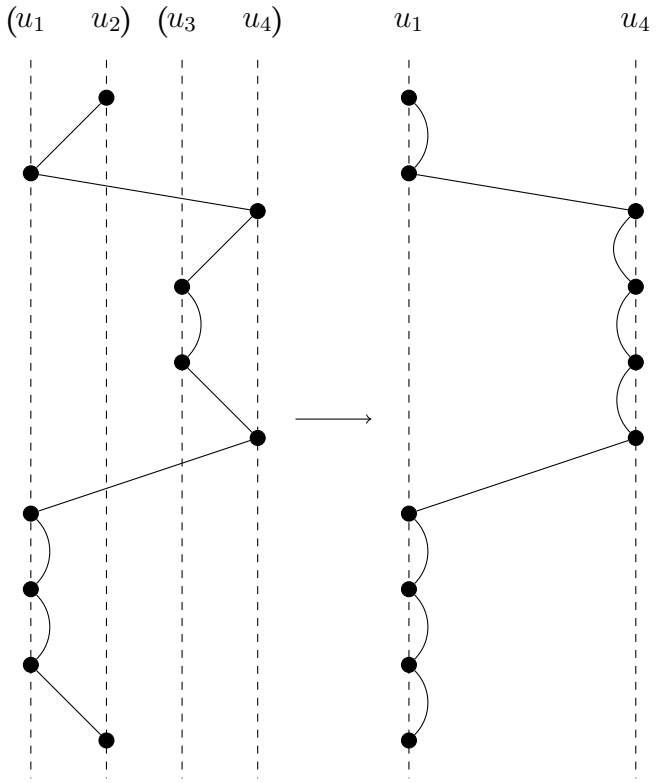


(a) An example of collapsing a 4-interleaving graph.



(b) An example of a restriction of a 4-interleaving graph.

Figure 23: Examples of operations on n-interleaving graphs.



(c) An example of a segregation of a 4-interleaving graph.

Figure 23: (cont.) Examples of operations on n-interleaving graphs.

4.2. Games whose moves are interleaving graphs

In order to describe plays of games, we need to extend notions of truncation and labelling to interleaving graphs. This provides us with a graphical representation of positions in a multi-component game.

At the beginning of Section 4 we saw an example of both the "folded" and "unfolded" plays of a game $A \multimap (B \otimes C)$.

In general, suppose a game A is built from games $A_1, \ldots, A_n$ using constructors $\otimes$ and $\multimap$, so that its structure is described by a binary $(\otimes, \multimap)$-word w. Let the main connective of w be $\square$, where $\square$ can stand for $\otimes$ or $\multimap$, so that $A = X \,\square\, Y$. We understand that each position of A is of the form (S, x, y), where S is a $\square$-schedule, x is a move of X and y is a move of Y, and where x and y themselves may take the form of labelled schedules.

To complete the representation of plays of compound games using interleaving graphs, we need to extend the notion of labelling and play labelling to interleaving graphs. We now can represent this with a single n-interleaving graph suitable for w, whose A_i-nodes are labelled with positions from the game A_i as appropriate.

In the structure of our game A, a connective $\square$ of w determines a subword of w for which it is the major connective. Within this subword, the interleaving of play across $\square$ is given by a $\square$-schedule. Therefore, the n-interleaving graph which represents the play must be such that, when restricted to the subword and segregated over $\square$, gives the same $\square$-schedule. This requirement is captured by the suitability of the n-interleaving graph for w.

When constructing games using $\otimes$ and $\multimap$, we generate moves by taking all possible $\otimes$- and $\multimap$-schedules of the right length, appropriately labelled. Therefore we represent plays using interleaving graphs in the following way:

Definition 68. Let A be a game formed using $\multimap$ and $\otimes$ from games $A_1, \ldots, A_n$ according to binary $(\otimes, \multimap)$-word w. The **unfolded form** of A is a game $\widetilde{A}$, given by the diagram

$$\widetilde{A}(1) \xleftarrow{\pi_{\widetilde{A}}} \widetilde{A}(2) \xleftarrow{\pi_{\widetilde{A}}} \cdots$$

where $\widetilde{A}(k)$ is the set of $(n+1)$-tuples $(S, a^{(1)}, \ldots, a^{(n)})$ with

- $S = (U_{k_1}^{(1)}, \ldots, U_{k_n}^{(n)}, \Sigma, \iota)$ is an n-interleaving graph suitable for w, and with path order of nodes $\{p_1, \ldots, p_k\}$
- $\sum_{i=1}^{n} k_i = k$
- $a^{(i)} \in \hat{A}_i(k_i)$

and with predecessor function $\pi_{\widetilde{A}}$ given by

$$\pi_{\widetilde{A}} : (S, a^{(1)}, \ldots, a^{(n)}) \mapsto (S \restriction_{k-1}, \ldots, \pi_{A_i}(a^{(i)}), \ldots)$$

for $p_k \in U^{(i)}$.

Example 69. Every game $A \otimes B$ or $A \multimap B$ whose positions are given by labelled $\otimes$- or $\multimap$-schedules are equal to their unfolded forms, if the schedules are viewed as 2-interleaving graphs.

4.3. Folding and unfolding interleaving graphs

In this section we describe an isomorphism between each game A and its unfolded form $\widetilde{A}$. This isomorphism is at the level of the game forests themselves. This isomorphism will take the form of a process of *unfolding* a position of A into an interleaving graph forming a position of $\widetilde{A}$.

In the following we let the symbol $\square$ stand for either $\otimes$ or $\multimap$.

Definition 70. Let w be a $(\otimes, \multimap)$-word w containing letter X. Let G be an n-interleaving graph suitable for w.

Suppose within w we wish to replace X by $X_1 \square X_2$, to get the word $w[X \rightsquigarrow X_1 \square X_2]$.

Given a $\square$-schedule S, we may **unfold** G **with** S to give the $(n+1)$-interleaving graph $G[X \overset{S}{\rightsquigarrow} X_1 \square X_2]$ suitable for $w[X \rightsquigarrow X_1 \square X_2]$:

- The X-nodes of G are arranged on a vertical line L_u.
- Since G is compact as a subset of $\mathbb{R}^2$, we may find a vertical strip $[u^-, u^+] \times \mathbb{R}$, with $u^- < u < u^+$, whose interior contains the X-nodes and whose closure contains no other nodes of G.
- Draw S so that its nodes lie on the boundary of $[u^-, u^+] \times \mathbb{R}$ and so that its i-th node is at the same vertical position as x_i.
- $G[X \rightsquigarrow X_1 \square X_2]$ is the deformation of G (as a progressive plane graph) so that G's X-nodes are coincident with the nodes of S and so that G's other nodes are unmoved.

It is clear from this construction that collapsing $G[X \overset{S}{\rightsquigarrow} X_1 \square X_2]$ on $X_1 \square X_2$ yields an n-interleaving graph which is equal to G up to deformation.

By viewing a $\square$-schedule as a 2-interleaving graph, the above process gives us a way to unfold a labelled $\square$-schedule, whose nodes are labelled by $\square$-schedules, into an n-interleaving graph. (We begin from the main connective and proceeding with subwords in a recursive fashion.)

Finally, we may extend unfolding to encompass labelled interleaving graphs.

Example 71. Figure 20(a) shows an example play of some game $A \multimap (B \otimes C)$ with

$$\pi_A : a_4 \mapsto a_3 \mapsto a_2 \mapsto a_1 \in A(1)$$
$$\pi_B : b_2 \mapsto b_1 \in B(1)$$
$$\pi_C : c_2 \mapsto c_1 \in C(1)$$

Since it is a $\multimap$-game, the plays of $A \multimap (B \otimes C)$ are given by labelled $\multimap$-schedules. In this case the labels for the nodes on the right are positions of the game $B \otimes C$, whose positions are themselves labelled $\otimes$-schedules.

Figure 20(b) shows the unfolding of the $\multimap$-schedule into a 3-interleaving graph. Notice how restricting to the dotted box leaves a labelled $\otimes$-schedule deformable into the final label on the right of the original $\multimap$-schedule.

Proposition 72. *Unfolding of interleaving graphs is confluent up to deformation.*

Proof. This follows from the observation that there are no critical pairs.

In other words, because we consider games with a binary compositional structure, the choice of order in unfolding corresponds to a traversal of the formula tree for the game's structure, where a child node must be visited after its parent. Therefore, all operations on the graph can be local; operating on the formula tree's right branch can have no effect on the portion of a graph corresponding to the left branch, and vice versa. □

The reverse of the unfolding process is the *folding* process, which takes an n-interleaving graph G suitable for some binary $(\otimes, \multimap)$ word w and encodes one subword $(X_1 \,\square\, X_2)$ as labels on the X nodes of an $(n-1)$ interleaving graph $G[X_1 \,\square\, X_2 \rightsquigarrow X]$, suitable for $w[X_1 \,\square\, X_2 \rightsquigarrow X]$.

Definition 73. Let w be a $(\otimes, \multimap)$-word with subword $(X_1 \,\square\, X_2)$. Let G be an n-interleaving graph suitable for w. We encode the interleaving of the $(X_1 \,\square\, X_2)$-nodes of G on the X-nodes of an $(n-1)$-interleaving graph $G[X_1 \,\square\, X_2 \rightsquigarrow X]$ suitable for $w[X_1 \,\square\, X_2 \rightsquigarrow X]$ using the following **folding process**:

- The underlying graph of $G[X_1 \,\square\, X_2 \rightsquigarrow X]$ is that of G with its $(X_1 \,\square\, X_2)$-nodes collapsed.
- The label on the final X-node of $G[X_1 \,\square\, X_2 \rightsquigarrow X]$ is the restriction of G to its $(X_1 \,\square\, X_2)$-nodes. Previous X-node labels are found by truncation. All other labels are as in G.

It is clear from this construction that these two operations of unfolding and folding are mutual inverses up to deformation (and deformation of schedules inside labels). Furthermore, an n-interleaving graph suitable for a binary $(\otimes, \multimap)$-word w can be folded into a labelled $\square$-schedule, where $\square$ stands for w's main connective.

Proposition 74. *A game of the form $A \,\square\, B$ consists of labelled $\square$-schedules. If A is of the form $A_1 \otimes A_2$ or $A_1 \multimap A_2$, then the game which consists of the unfoldings of each of the $\square$-schedules of $A \,\square\, B$ into 3-interleaving graphs is isomorphic to $A \,\square\, B$. Similarly if B is of the form $B_1 \otimes B_2$ or $B_1 \multimap B_2$.*

Proof. Viewing a game A as its set of complete plays, with π_A given by truncation, that unfolding is an isomorphism follows from the fact that it respects truncation. □

Corollary 75. *A game A of the form $A_1 \,\square\, A_2$ is isomorphic to its unfolded form $\tilde{A}$.*

Corollary 76. *If games A and B are such that $\widetilde{A}$ and $\widetilde{B}$ have the same positions (up to deformation as interleaving graphs), then A and B are isomorphic.*

Now we can understand composition of interleaving graphs: Though we can always consider plays, and hence strategies, of games as consisting of interleaving graphs, when we compose strategies, we follow the definition of composition of labelled $\multimap$-schedules, and so use folded plays, only unfolding after composition.

From now on we will frequently refer to a compound game such as $(A \otimes A') \multimap (B \otimes B')$ as having plays consisting of interleaving graphs, with the understanding that we are working with the unfolded representation.

5. Symmetric monoidal closed structure on *Game*

We now extend the notion of $\otimes$ from an operation on the objects of *Game* to the morphisms, providing a monoidal structure on *Game* [2, 26]. We then proceed to apply the techniques of Section 4 to give a proof of some categorical properties of *Game*.

In a game $A \multimap B$, O must start in B after which P has the choice of whether to answer in B or play as O in A. At each position, only P has the option to switch games, whereas O must remain in the same component as the previous move. In a tensor game $A \otimes B$, O may choose which component to play in and P must respond in the same component.

Given two strategies $\sigma : A \multimap B$ and $\tau : A' \multimap B'$, the strategy $\sigma \otimes \tau : (A \otimes A') \multimap (B \otimes B')$ allows O to switch components in $B \otimes B'$ but requires P to respond in accordance with σ if O's last move was in B or τ if O's last move was in B'. When in $A \otimes A'$, both O and P must respond in accordance with the strategy σ or τ currently being played.

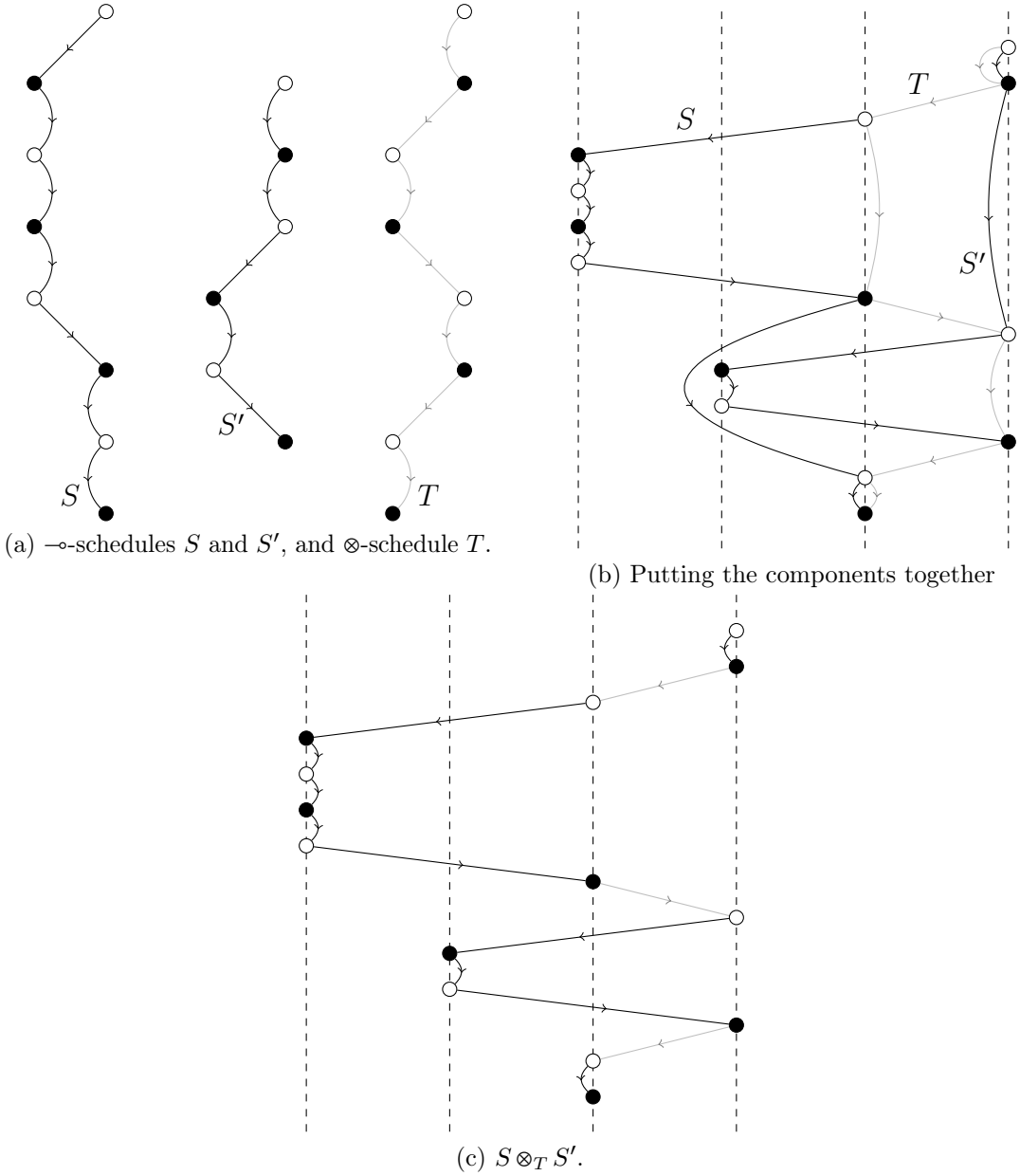
Along these lines we describe a method of combining the schedules which comprise plays in σ and τ into 4-interleaving graphs describing plays in $\sigma \otimes \tau$.

Definition 77. Let $S : m \to n$ and $S' : m' \to n'$ be $\multimap$-schedules. Let $T : n \otimes n'$ be a $\otimes$-schedule.

- Select points $u < u' < v < v' \in \mathbb{R}$ and embed T in the vertical strip $[v, v'] \times \mathbb{R}$. Colour the nodes of T black ($\bullet$) and white ($\circ$) so that the first node is white and nodes alternate white–black along the path T.
- Embed S' in $[u', v'] \times \mathbb{R}$ so that the nodes of S' which lie in $L_{v'}$ coincide with the corresponding nodes of T which lie in $L_{v'}$. Note that if we similarly colour the nodes of S', these colours agree where the nodes of S' and T coincide.
- Embed S in $[u, v] \times \mathbb{R}$ so that the nodes of S which lie in L_v coincide with the corresponding nodes of T which lie in L_v, and so that S does not touch S'. Note that if we similarly colour the nodes of S, these colours agree where the nodes of S and T coincide.

This gives us a kind of composition graph from which the appropriate 4-interleaving graph, which we call $S \otimes_T S'$ may be extracted.

- The nodes of $S \otimes_T S'$ are the the nodes of S and S' (which together include the nodes of T).
- The edges of $S \otimes_T S'$ are selected by the following procedure, starting at the first node of T:
  - From a white node in $[v, v'] \times \mathbb{R}$, take the outward edge in S or S' (there will be only one such option, by construction) and continue to take edges in sequence until reaching a further node in $[v, v'] \times \mathbb{R}$.
  - From a black node in $[v, v'] \times \mathbb{R}$, take the outward edge in T and continue to take edges in sequence until reaching a further node in $[v, v'] \times \mathbb{R}$.



(a) $\multimap$-schedules S and S', and $\otimes$-schedule T.

(b) Putting the components together

(c) $S \otimes_T S'$.

Figure 24: Example of construction $S \otimes_T S'$.

– All unselected edges are discarded.

Example 78. Figure 24 shows an example of this construction.

The construction of Definition 77 may be straightforwardly extended to the construction of a labelled 4-interleaving graph from two labelled $\multimap$-schedules and a $\otimes$ schedule.

The following follow from the recursive definition of suitability.

Proposition 79. *With S suitable for $A \multimap B$, S' suitable for $A' \multimap B'$ and T suitable for $B \otimes B'$, $S \otimes_T S'$ is suitable for $(A \otimes A') \multimap (B \otimes B')$.*

Corollary 80. *The 4-interleaving graph $S \otimes_T S'$, viewed as suitable for $(A \otimes B) \multimap (C \otimes D)$, uniquely determines a $\otimes$-schedule, found by restricting to its $(A \otimes B)$-nodes. We call this schedule $\check{T}$.*

Remark 81. The following are true of $S \otimes_T S'$:

1. We may recover T from $S \otimes_T S'$ by restricting to $[v, v'] \times R$.

2. We may recover S or S' from $S \otimes_T S'$ by restricting to $\{u, v\} \times \mathbb{R}$ and $\{u', v'\} \times \mathbb{R}$ respectively.

We now use this construction to give an explicit description of the action of $\otimes$ on the morphisms of *Game*.

Definition 82. Recall that a strategy $\sigma : A \multimap B$ is a collection of even-length labelled $\multimap$-schedules so that the longest common truncation of any two is of even length. Likewise for a strategy $\tau : A' \multimap B'$.

The **tensor** product $\sigma \otimes \tau$ is the strategy given by the collection of all possible (up to deformation) labelled 4-interleaving graphs achieved by the following

- Take a labelled $\multimap$-schedule $S : m \to n$ from σ.
- Take a labelled $\multimap$-schedule $S' : m' \to n'$ from τ.
- Take a $\otimes$-schedule $T : n \otimes n'$.
- Form the labelled 4-interleaving graph $S \otimes_T S'$.

These interleaving graphs may of course be considered as $\multimap$-schedules, in accordance with the definition of a strategy for a game of the form $X \multimap Y$.

Proposition 83. *$\otimes$ is a tensor product on* *Game*.

Proof. We require:

(i) $\otimes$ has a unit.

(ii) $\otimes$ is bifunctorial.

(iii) $\otimes$ is associative.

Unit. The *empty game*, I, given by $I(k) = \varnothing$ for each k, is the unit of the tensor product $\otimes$.

Bifunctoriality. The following is a quick précis of the proof of bifunctoriality. For full details see Appendix Appendix A.2, with illustrations in Figure A.28.

Plays in $(\sigma \otimes \sigma') \| (\tau \otimes \tau')$ are given by

$$(S \otimes_{\check{Z}} S') \| (T \otimes_Z T')$$

appropriately labelled. Since $\check{Z}$ is determined by T, T' and Z, we find such a play is equal to

$$(S \| T) \otimes_Z (S' \| T')$$

up to deformation, and thus is a play of $(\sigma \| \tau) \otimes (\sigma' \| \tau')$.

The opposite direction follows by reversing the argument, together with the observation that if a $\multimap$-schedule is of the form $S \| T$, its diagram may be deformed to exhibit its composite structure.

Associativity. This follows in a similar manner from Corollary 76. □

Proposition 84. *There is a well-defined symmetry for* $\otimes$.

Proof. For each $\otimes$-schedule $S^{\otimes}_{p,q}$ underlying a play in $Y \otimes X$ we can construct a unique 4-interleaving graph G suitable for $(X \otimes Y) \multimap (Y \otimes X)$ such that the restriction of G to $Y \otimes X$ is $S^{\otimes}_{p,q}$, the restriction of G to $X \otimes Y$ is (up to deformation) the $\otimes$-schedule $\bar{S}_{q,p}$, the reflection of $S^{\otimes}_{p,q}$ in the vertical axis, and the segregation of G over the $\multimap$ gives a copycat $\multimap$-schedule.

This 4-interleaving graph is constructed by starting with $I_{p,p} \otimes_S I_{q,q}$ and then deformed as a progressive plane graph so as to exchange the leftmost two vertical lines on which nodes lie. This is illustrated by example in Figure 25(a).

Now we have that $\bar{S} \| G = S$, as desired. This is illustrated by example in Figure 25(b).

The strategy $b_{X,Y} : (X \otimes Y) \multimap (Y \otimes X)$ is then the collection of all such labelled 4-interleaving graphs, for each labelled $\otimes$-schedule in $Y \otimes X$. The strategy $b_{X,Y}$ is self-inverse by construction. □

Proposition 85. *For games* A, B, C*, we have*

$$(A \otimes B) \multimap C \cong A \multimap (B \multimap C)$$

Proof. This follows from the observation that a labelled 3-interleaving graph suitable for $(A \otimes B) \multimap C$ is also suitable for $A \multimap (B \multimap C)$ and vice versa.

In a 3-interleaving graph describing a play of $(A \otimes B) \multimap C$ first node in the graph must be a C-node, and all subsequent nodes come in pairs of A-, B- or C-nodes. Similarly for a play of $A \multimap (B \multimap C)$. Corollary 76 allows us to make such an argument. □

Proposition 85 is illustrated by example in Figure 26.

With Propositions 83, 84 and 85 providing a structure on $\mathcal{G}ame$, we get the following important results [27, 12].

Theorem 86. $\mathcal{G}ame$ *has a symmetric monoidal closed structure.*

Corollary 87. $\mathcal{G}ame$ *models multiplicative intuitionistic linear logic.*

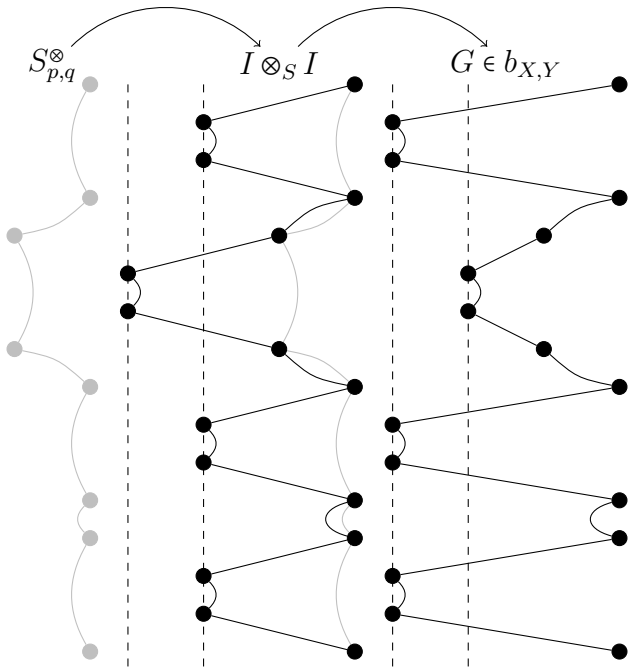
6. Concluding remarks and future work

We have given a detailed and formal account of a graphical foundation for interleaving in plays of dialogue games and strategies. We have seen in particular how diagrams drawn by researchers, informed by their intuitions, can provide intuitive and rigorous proofs when formalised.

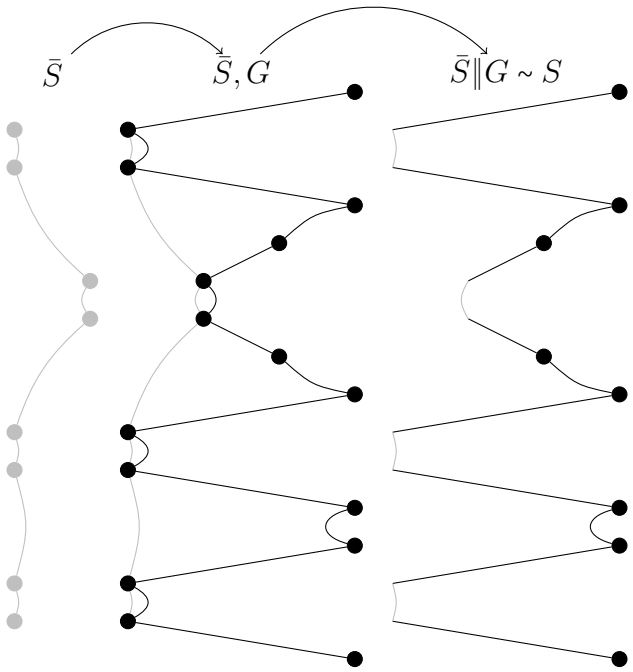
Following these results it seems appropriate to investigate the relationship with the 2-categorical string diagrams for dialogue games in [11]. Further, the ubiquitous diagrammatic representations of *heaps* and *pointer functions* which seem to capture structure also sit well within the framework we have used here [28, 29, 30, 10].

The use of progressive plane graphs in formal accounts of graphical methods in algebra is established in the literature [2, 31, 6]. As well as Joyal and Street's framework [2] providing a "realistic" foundation for diagrams in games, it is also directly extensible to other classes of plane diagrams [32, 31, 33, 6] and we hope choosing it as our foundation will provide common ground for future work, perhaps contributing new categories of games and strategies.

Further investigations in this direction have been conducted in [34], which gives an account of Harmer et al.'s results on the category of games and innocent strategies via the exponential monad and comonad ? and ! [12] in the graphical terms of this paper.



(a) Construction of G from $I \otimes_S I$.



(b) $\bar{S} \| G \sim S$.

Figure 25: Symmetric structure on $\mathcal{G}ame$.

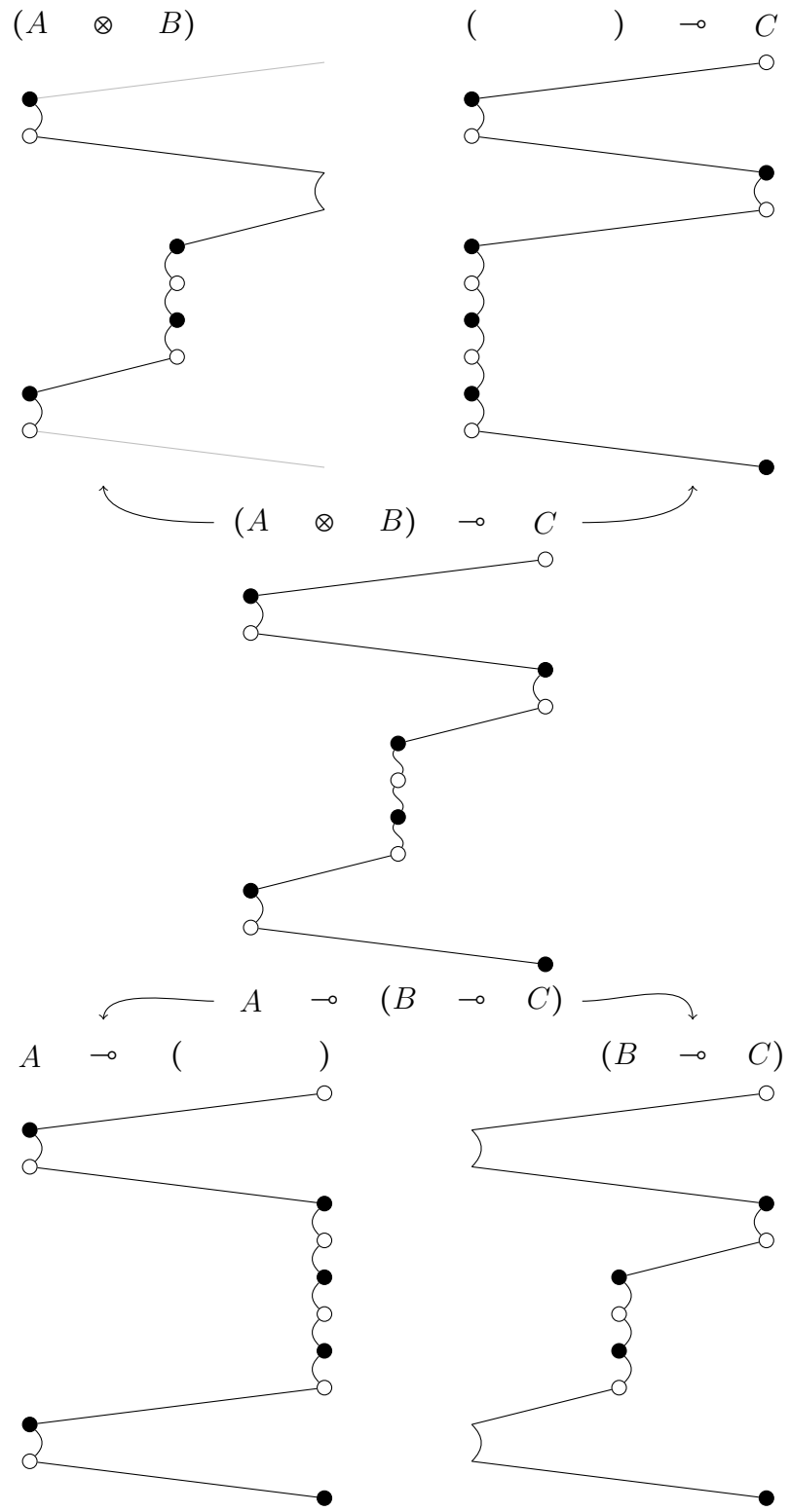


Figure 26: Below the central figure we see that it is suitable for $A \multimap (B \multimap C)$. Above the central figure we see that it is suitable for $(A \otimes B) \multimap C$.

7. Bibliography

[1] J.-Y. Girard, Linear logic, Theoretical Computer Science.

[2] A. Joyal, R. Street, The geometry of tensor calculus I, Advances in Mathematics 88 (1) (1991) 55–112.

[3] A. Guglielmi, T. Gundersen, Normalisation control in deep inference via atomic flows, Logical methods in computer science.

[4] J. Baez, M. Stay, Physics, topology, logic and computation: a Rosetta stone, in: B. Coecke (Ed.), New Structures in physics, Lecture Notes in Physics, 2011, pp. 95–172.

[5] J. Lambek, Compact monoidal categories from linguistics to physics, in: New Structures for Physics, Lecture notes in physics, Springer, 2011, pp. 467–487.

[6] P. Selinger, A survey of graphical languages for monoidal categories, Lecture Notes in Physics, Springer Berlin/Heidelberg, 2011, pp. 289–355.

[7] S. Abramsky, R. Jagadeesan, Games and full completeness for multiplicative linear logic, in: R. Shyamasundar (Ed.), Foundations of Software Technology and Theoretical Computer Science, Vol. 652 of Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 1994, pp. 291–301.

[8] S. Abramsky, R. Jagadeesan, P. Malacaria, Full abstraction for PCF, Information and Computation 163 (2) (2000) 409–470.

[9] S. Abramsky, G. McCusker, Linearity, sharing and state: a fully abstract game semantics for Idealised Algol with active expressions, Algol-like languages, Progress in theoretical computer science 2 (1997) 297–330.

[10] M. Hyland, L. Ong, On full abstraction for PCF: I, II, and III, Information and computation 163 (2) (2000) 285–408.

[11] P.-A. Melliès, Categorical models for linear logic revisited, To appear, Theoretical Computer Science.

[12] R. Harmer, M. Hyland, P.-A. Melliès, Categorical combinatorics for innocent strategies, Logic in Computer Science. LICS 2007. 22nd Annual IEEE Symposium on (2007) 379–388.

[13] S. Abramsky, G. McCusker, Game semantics, in: U. Berger, H. Schwichtenberg (Eds.), Computational Logic, Vol. 165 of NATO ASI Series, Springer Berlin/Heidelberg, 1999, pp. 1–55.

[14] R. Harmer, Games and full abstraction for non-deterministic languages, Ph.D. thesis, University of London (2000).

[15] M. Hyland, Combinatorics of proofs, `http://dpmms.cam.ac.uk/~martin/Research/Slides/licsasl07.pdf` (2007).

[16] M. Hyland, Game semantics, in: A. Pitts, P. Dybjer (Eds.), Semantics of Logics and Computation, Publications of the Newton Institute, Cambridge University, 1997, pp. 131–184.

[17] S. Schanuel, R. Street, The free adjunction, Cahiers de topologie et géométrie différentielle catégoriques 27 (1) (1986) 81–83.

[18] P.-A. Melliès, Game semantics in string diagrams, in: Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science, IEEE Computer Society, 2012, pp. 481–490.

[19] R. Harmer, Innocent game semantics, lecture Notes. Available on author's website: `http://pps.jussieu.fr/~russ/GS.pdf` (2004).

[20] G. McCusker, J. Power, C. Wingfield, A graphical foundation for schedules, Electronic Notes in Theoretical Computer Science 286 (2012) 273–289, proceedings of the 28th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXVIII). doi:10.1016/j.entcs.2012.08.018.

[21] M. Hyland, A. Schalk, Games on graphs and sequentially realizable functionals. extended abstract, Proceedings of Seventeenth Annual IEEE Symposium on Logic in Computer Science (2002) 257–264.

[22] M. A. Armstrong, Basic Topology, Springer-Verlang, 1983.

[23] S. Abramsky, Semantics of interaction, in: H. Kirchner (Ed.), Trees in Algebra and Programming — CAAP, Vol. 1059 of Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 1996.

[24] A. Schalk, Games for semantics: an introduction, Unpublished notes, viewed February 2013, available at author's website: `http://www.cs.man.ac.uk/~schalk/notes/intgam.ps.gz`.

[25] O. Laurent, Polarized games, Annals of Pure and Applied Logic 130 (1–3) (2004) 79–123.

[26] S. MacLane, Categories for the Working Mathematician. Second Edition., Springer, 1997.

[27] R. A. G. Seely, Linear logic, *-autonomous categories and cofree algebras, Proceedings, AMS conference on categories in computer science and logic.

[28] P. Clairambault, R. Harmer, Totality in arena games, Annals of pure and applied logic 5 (161) (2010) 673–689.

[29] R. Harmer, O. Laurent, The anatomy of innocence revisited, in: S. Arun-Kumar, N. Garg (Eds.), Foundations of Software Technology and Theoretical Computer Science, Vol. 4337 of Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2006, pp. 224–235.

[30] V. Danos, R. Harmer, The anatomy of innocence, in: L. Fribourg (Ed.), Computer Science Logic, Vol. 2142 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2001, pp. 188–202.

[31] M. C. Shum, Tortile tensor categories, Journal of Pure and Applied Algebra 93 (1) (1994) 57–110.

[32] A. Joyal, R. Street, Braided tensor categories, Advances in Mathematics 102 (1) (1993) 20–78.

[33] P.-A. Melliès, Functorial boxes in string diagrams, in: Z. Ésik (Ed.), Computer Science Logic, Vol. 4207 of Lecture Notes in Computer Science, Springer BerlinHeidelberg, 2006.

[34] C. Wingfield, Graphical foundations for dialogue games, Ph.D. thesis (2013).

Appendix A. Appendices

Appendix A.1. Identity schedules

$I_{n,n}$ may be explicitly defined, for example as

$$I_{n,n} = (U'_n, U_n, \Sigma_{n,n}, \mathrm{id} : \Sigma \to [0,1] \times \mathbb{R})$$

where:

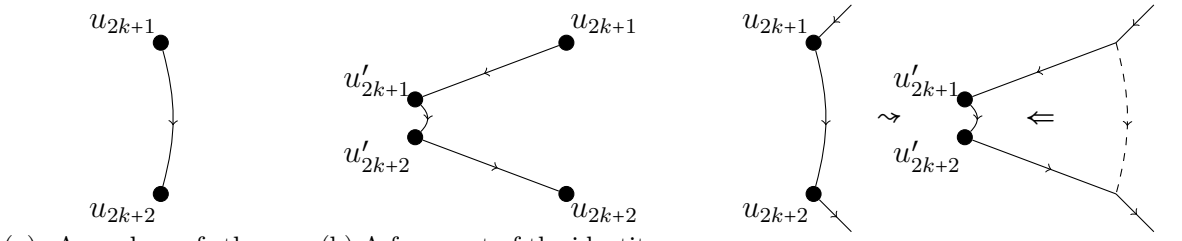
$$\begin{aligned} U'_n = \{u'_i \,|\, & i \text{ odd} \implies u'_i = (0, -2i), \\ & i \text{ even} \implies u'_i = (0, 1-2i)\} \\ U_n = \{u_i \,|\, & i \text{ odd} \implies u_i = (1, -2i), \\ & i \text{ even} \implies u_i = (1, 1-2i)\} \end{aligned}$$

$$\Sigma = (\Sigma, U'_n + U_n)$$

where:

$$\begin{aligned} \Sigma = & \bigcup\{\text{line segments } [u_{2k-1}, u'_{2k-1}]\} \cup \cdots \\ & \bigcup\{\text{line segments } [u'_{2k}, u_{2k}]\} \cup \cdots \\ & \bigcup\{\text{circular arc, endpoints } \{u'_{2k-1}, u'_{2k}\}, \\ & \quad \text{angle } \alpha < \pi\} \cup \cdots \\ & \bigcup\{\text{circular arc, endpoints } \{u_{2k}, u_{2k+1}\}, \\ & \quad \text{angle } \alpha < \pi\} \end{aligned}$$

Of course, we consider any deformation of this to be an identity schedule.



(a) An edge of the schedule $S_{m,n}$.

(b) A fragment of the identity schedule.

(c) A fragment of the deformation demonstrating that copycat schedules are identities of schedule composition.

Figure A.27: Composition with the identity schedule.

Lemma 88. *Left and right composition with $I_{n,n}$ satisfies identity axioms.*

Proof. First, for composition on the left, let $S_{m,n} : U \to V$ be a schedule and let $I_{m,m} : U' \to U$ be the identity schedule. We want to show that $I_{m,m} \| S_{m,n} \cong S_{m,n}$.

Since $S_{m,n}$ is a schedule, we have that u_{2k+1} and u_{2k+2} are joined by an edge, as in Figure A.27(a). Since $I_{m,m}$ is a copycat, we know that u_{2k+1} and u_{2k+2} are joined in $I_{m,m}$by the identity schedule fragment in Figure A.27(b).

We know that in $I_{m,m} \| S_{m,n}$, the edge into u_{2k+1} to be chosen will be the one from $S_{m,n}$, and the edge out of u_{2k+1} will be the one from $I_{m,m}$, after which u_{2k+1} will be "declassified" as a node. Similarly, the edge into u_{2k+2} to be chosen will be the one from $I_{m,m}$ and the edge out will be the one from $S_{m,n}$ before u_{2k+2} is declassified. Then the equality of the schedule fragment surrounding

u_{2k+1} and u_{2k+1} (which will become the schedule fragment surrounding u'_{2k+1} and u'_{2k+1} in the composite) holds up to the evident deformation in Figure A.27(c).

Between points u_{2k} and u_{2k+1} all activity in the composite will be the activity in $S_{m,n}$, as u_{2k} is approached from the right in the construction of $I_{m,m} \| S_{m,n}$ and so u_{2k+1} must be approached from the left.

A similar argument also demonstrates that for schedule $S_{m,n} : U \to V$ and copycat schedule $I_{n,n} : V \to V'$, we have $S_{m,n} \cong S_{m,n} \| I_{n,n}$. □

Appendix A.2. Bifuctoriality of $\otimes$

Lemma 89. $\otimes$ *is bifunctorial.*

Proof. Let A, A', B, B', C, C' be games and let $\sigma : A \multimap B$, $\sigma' : A' \multimap B'$, $\tau : B \multimap C$ and $\tau' : B' \multimap C'$ be strategies. We wish to show that $\otimes$ is bifunctorial, i.e. that

$$(\sigma \otimes \sigma') \| (\tau \otimes \tau') = (\sigma \| \tau) \otimes (\sigma' \| \tau') \tag{A.1}$$

We will illustrate the argument with a running example in Figure A.28.

Since both sides of (A.1) are sets of plays, we will proceed by showing set inclusions in both directions.

(⊆). Consider

$$(\sigma \otimes \sigma') \| (\tau \otimes \tau') : (A \otimes A') \multimap (C \otimes C') \tag{A.2}$$

This is a composite of strategies $\sigma \otimes \sigma' : (A \otimes A') \multimap (B \otimes B')$ and $\tau \otimes \tau' : (B \otimes B') \multimap (C \otimes C')$. So (by the definition of composition of strategies) the plays of (A.2) are given by all 4-interleaving graphs which are composites of 4-interleaving graphs in $\sigma \otimes \sigma'$ and $\tau \otimes \tau'$. Which ones compose?

A play of $\tau \otimes \tau'$ is given by a 4-interleaving graph $T \otimes_Z T'$ with $T : n \to q$ and $T' : p \to r$ two $\multimap$-schedules and $Z : q \otimes r$ a $\otimes$-schedule. Let $\check{Z} : n \otimes p$ be the $\otimes$-schedule induced by $T \otimes_Z T'$. Likewise a play of $\sigma \otimes \sigma'$ is a 4-interleaving graph $S \otimes_{Z'} S'$ with $S : j \to l$ and $S' : k \to m$ two $\multimap$-schedules and $Z' : l \otimes m$ a $\otimes$-schedule.

In our running example, S, S', T, T' are shown in Figure A.28(a), and Z and $\check{Z}$are shown in Figure A.28(b).

Such a $T \otimes_Z T'$ and $S \otimes_{Z'} S'$ are composable just when $\check{Z}$ is deformable into Z' with the corresponding labels in $B \otimes B'$ matching. Therefore we require

$$l = n \qquad m = p \qquad Z' \sim \check{Z}$$

So a play in (A.2) is given by a 4-interleaving graph

$$(S \otimes_{\check{Z}} S') \| (T \otimes_Z T') \tag{A.3}$$

which is completely determined by S, S', T, T' and Z.

$T \otimes_Z T'$ and $S \otimes_{\check{Z}} S'$ are shown by example in Figures A.28(c) and A.28(d).

Composing (A.3) hides activity in $B \otimes B'$, with play continuing in whichever (left or right) component we are in:

- If we enter a B-node from the right we are in an edge of T so must leave in S.
- If we enter a B-node from the left we are in an edge of S so must leave in T.

- If we enter a B'-node from the right we are in an edge of T' so must leave in S'.
- If we enter a B'-node from the left we are in an edge of S' so must leave in T'.

So the graph moves between A and C with edges from $S\|T$, between A' and C' with edges from $S'\|T'$, and between C and C' via Z; exactly as in the definition of $(S\|T)\otimes_Z(S'\|T')$, which is play of $(\sigma\|\tau)\otimes(\sigma'\|\tau')$. So we have the first inclusion

$$(\sigma\otimes\sigma')\|(\tau\otimes\tau')\subseteq(\sigma\|\tau)\otimes(\sigma'\|\tau') \tag{A.4}$$

In our example, the composition diagram (A.3) is shown in Figure A.28(e) with the final composite shown in Figure A.28(f).

(⊇). A play in $(\sigma\|\tau)\otimes(\sigma'\|\tau')$ is given by a 4-interleaving graph $(S\|T)\otimes_Z(S'\|T')$ for some appropriate $\multimap$-schedules $S\|T$ and $S'\|T'$, and $\otimes$-schedule Z. Examples are given in Figure A.28(g).

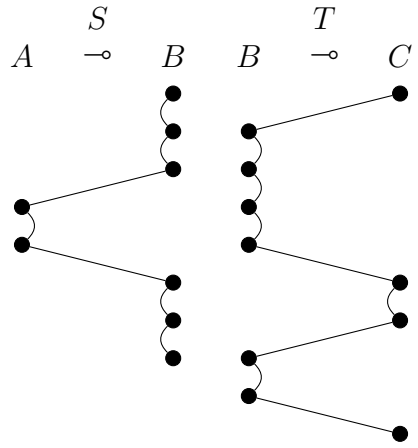
The key fact is that a $\multimap$-schedule $S\|T$ known to be a composition of $\multimap$-schedules S and T may be deformed so that a vertical line separates it into subgraphs of S and T. This may be achieved by reversing the procedure of Definition 12, and can be seen by example in Figure A.28(h). This provides positions in B and similar procedure with $S'\|T'$ gives positions in B'. This can be done simultaneously, as is shown in Figure A.28(i). Finally there's a uniquely determined $\otimes$-schedule interleaving positions in $B\otimes B'$ and it is $\check{Z}$ induced by $T\otimes_Z T'$.

Hence

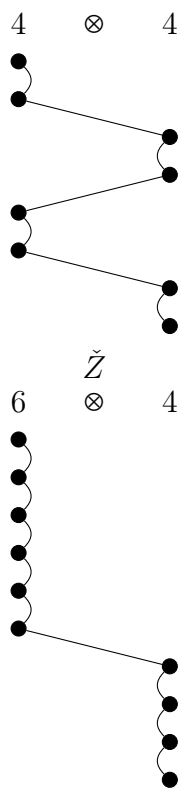
$$(\sigma\otimes\sigma')\|(\tau\otimes\tau')\supseteq(\sigma\|\tau)\otimes(\sigma'\|\tau') \tag{A.5}$$

And (A.4) and (A.5) together give (A.1), as required. □

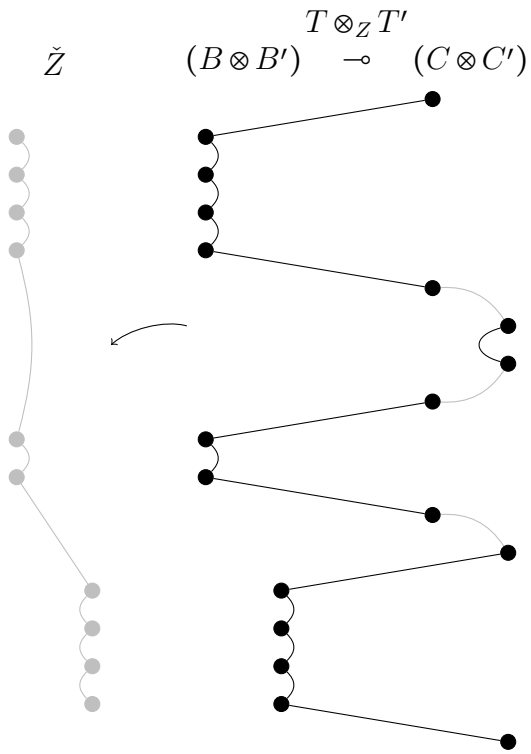
Remark 90. In the proof of Lemma 89, the fact that a play $(S\otimes_{\check{Z}}S')\|(T\otimes_Z T')$ is deformable into a play $(S\|T)\otimes_Z(S'\|T')$ can also be easily seen, since the composite $(S\otimes_{\check{Z}}S')\|(T\otimes_Z T')$ is the unique (up to deformation) Hamiltonian path through the composition diagram. This can be seen in Figures A.28(e) and A.28(f).



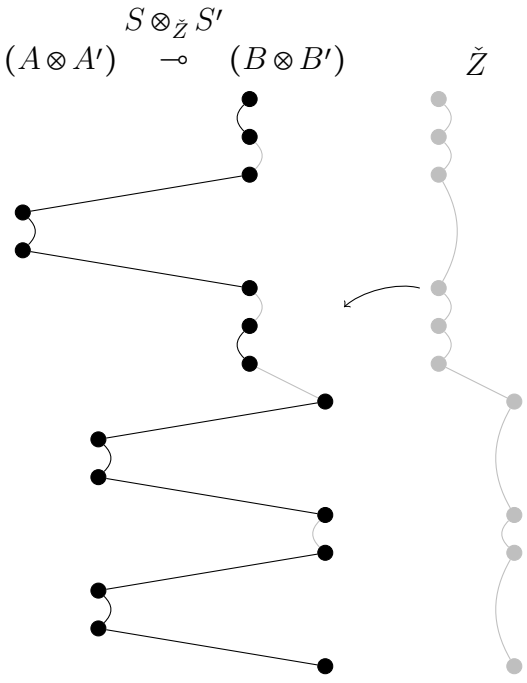
(a) $\multimap$-schedules S, T, S' and T'.



(b) The $\otimes$-schedules Z and the $\otimes$-schedule $\check{Z}$ induced by $T \otimes_Z T'$.

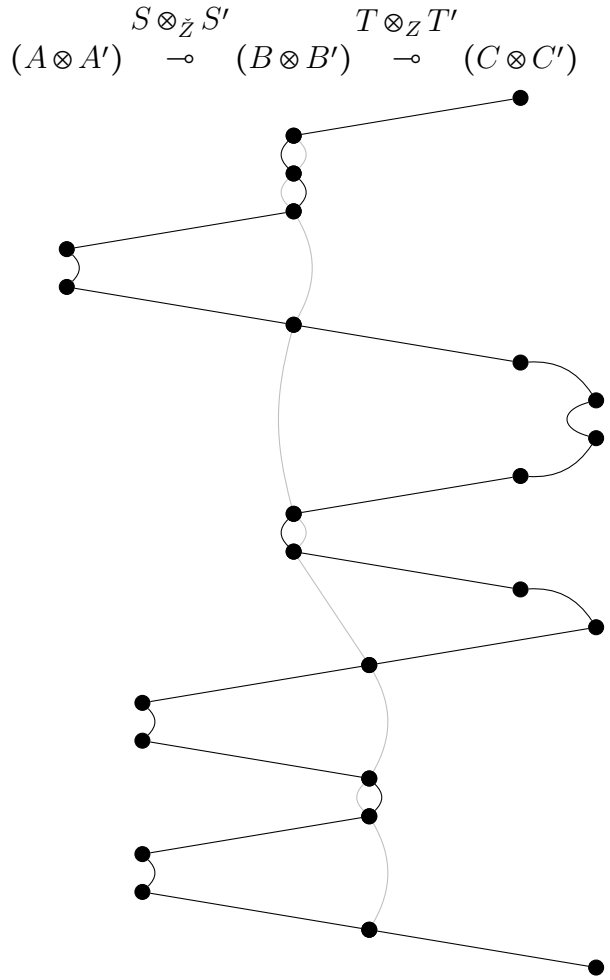


(c) The construction $T \otimes_Z T'$ and the induced $\check{Z}$.



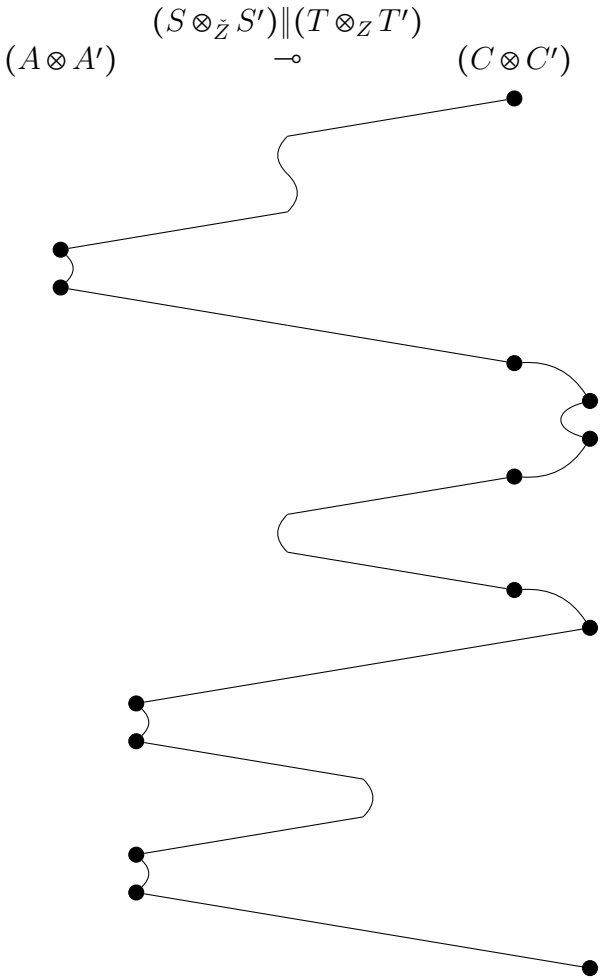
(d) The construction $S \otimes_{\check{Z}} S'$.

Figure A.28: Bifunctoriality

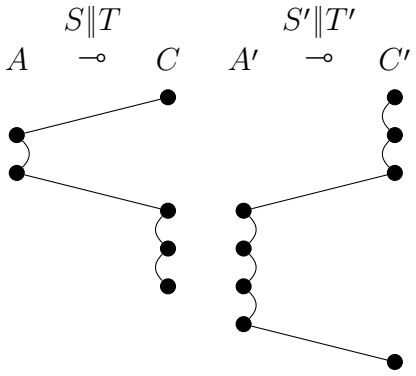


(e) Composition diagram for $(S \otimes_{\check{Z}} S') \| (T \otimes_Z T')$.

Figure A.28: (cont.) Bifunctoriality.

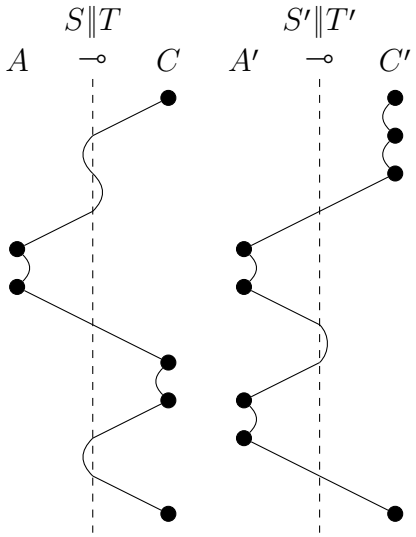


(f) 4-interleaving graph $(S \otimes_{\check{Z}} S') \| (T \otimes_Z T')$.

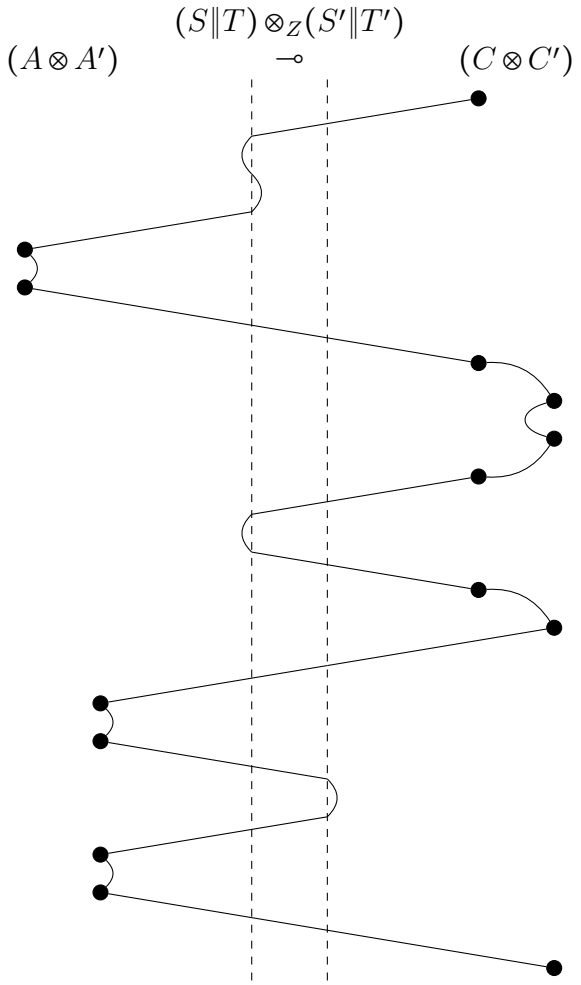


(g) Composites $S \| T$ and $S' \| T'$.

Figure A.28: (cont.) Bifunctoriality.



(h) Composites $S\|T$ and $S'\|T'$, deformed to show their composite structure.



(i) Deformation of $S\|T$ and $S'\|T'$ performed simultaneously on the graph of $(S\|T)\otimes_Z(S'\|T')$.

Figure A.28: (cont.) Bifunctoriality.