Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 286 (2012) 273–289

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Graphical Foundation for Schedules

Guy McCusker[1], John Power[2] and Cai Wingfield[3,4]

Department of Computer Science
University of Bath
Bath BA2 7AY, United Kingdom

Abstract

In 2007, Harmer, Hyland and Melliès gave a formal mathematical foundation for game semantics using a notion they called a schedule. Their definition was combinatorial in nature, but researchers often draw pictures when describing schedules in practice. Moreover, a proof that the composition of schedules is associative involves 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 schedule, prove that it is equivalent to Harmer et al.'s definition, and illustrate its value by giving a proof of associativity of composition.

Keywords: Game semantics, geometry, schedules, composites, associativity.

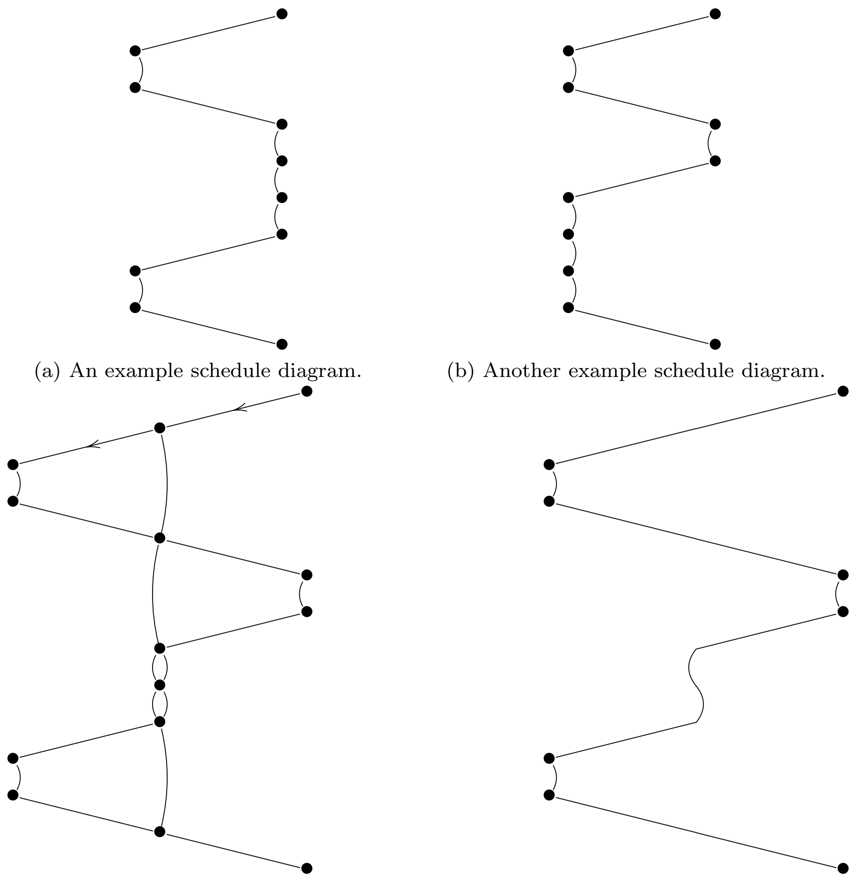
1 Introduction

Over recent decades, game semantics has become one of the standard forms of semantics for programming languages [2,3,4,12,16]. In 2007, Harmer, Hyland and Melliès gave a formal mathematical foundation for game semantics [9]. 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. They then defined a composite of schedules.

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). For examples, `1001111001` and `1001100001` are schedules $\{1, \ldots, 10\} \to \{0, 1\}$.

[1] Email: `g.a.mccusker@bath.ac.uk`
[2] Email: `a.j.power@bath.ac.uk`
[3] Email: `c.a.j.wingfield@bath.ac.uk`
[4] Supported by a PhD grant from the EPSRC.

doi:10.1016/j.entcs.2012.08.018



(a) An example schedule diagram.

(b) Another example schedule diagram.

(c) To compose, deform both schedules so that the two images of the *internal nodes* are identified, forming a *composition diagram.*

(d) The composite schedule is achieved by tracing along edges, starting from the first node on the right, swapping sides whenever an *internal node* is reached.

Fig. 1. Example schedules and composition.

Researchers typically describe schedules on the page or blackboard using a graphical representation [5,7,11,12]. For examples, Figures 1(a) and 1(b) are graphical representations of the above two schedules. Composites are also typically described graphically, in a manner implied by the description of schedules as pairs of order relations in [9]. Figures 1(c) and 1(d) describe the composite of the two examples of schedules above. While many people draw precisely such diagrams as these, there is another common practice which is to omit the lines — i.e. 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 vertical position, but no actual lines drawn. Such pictorially laid-out plays are still really schedule diagrams and the graphical definition of schedules in this paper encompasses them; arguments involving their composition are essentially the same as those here. In a sense, it is the fact that lines *could* be drawn that means such pictures represent schedules.

This situation gives rise to several natural questions which we explore in this paper. First, in Section 3, we characterise those pictures that arise from schedules.

In Section 4, we formally prove that Harmer et al.'s combinatorial definition and our geometric definition agree. We also define a composite of schedules in geometric terms and show that it agrees with Harmer et al.'s. In Section 5 we prove associativity of composition, give left and right identities, and thereby exhibit the 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 [9]. 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*.

Our graphical definition of schedule is set in the framework of Joyal and Street's string diagrams for monoidal categories [14]. 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 3 and 4, and in [14]. It is also possible to characterise schedules using the free adjunction $\mathcal{Adj}$ [19], cf. Melliès' 2-categorical string diagrams for adjunctions [18]. We plan to extend this graphical approach to encompass pointer diagrams [5,8,12], and in this way reformulate all of Harmer et al.'s paper in geometric terms.

2 Combinatorial foundation for schedules

In this section we recall the combinatorial definition of *schedules* and of composition of schedules from [9].

Definition 2.1 (as in Harmer et al. [9]) A **$\multimap$-scheduling function** is a function $e : \{1, \dots, n\} \to \{0, 1\}$ satisfying $e(1) = 1$ and $e(2k + 1) = e(2k)$.

Schedules $e : \{1, \dots, n\} \to \{0, 1\}$ 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 $|e| = n = |e|_0 + |e|_1$.

$\multimap$-scheduling functions are also called *schedules* in [9], but we will take care not to confuse this with Definition 3.4 of *schedule*, to follow. When necessary for disambiguation, we will call the latter a "graphical schedule". In Theorem 5.6 we will show that the definitions are equivalent.

Example 2.2 The following are scheduling functions:

(i) `1001111001` and `1001100001` are the examples we have already seen illustrated in Figures 1(a) and 1(b).

(ii) Any nonempty prefix (or *restriction* [9]) of a schedule is a schedule.

Definition 2.3 (as in Harmer et al. [9]) We will use the notation $e : p \to q$ when e is a schedule $e : \{1, \dots, p + q\} \to \{0, 1\}$ with $|e|_0 = p$ and $|e|_1 = q$.

Let $e : p \to q$ be a such a schedule. Writing $[n]$ to denote the set $\{1, \dots, n\}$, we will also write $[n]^+$ for the set of even elements of $[n]$ and $[n]^-$ for the set of odd elements of $[n]$. The schedule e corresponds to a pair of order-preserving, collectively surjective embeddings $e_L : [p] \hookrightarrow [p + q]$ and $e_R : [q] \hookrightarrow [p + q]$, where e_L is the order-preserving surjection to $e^{-1}(0) \subset [p+q]$ and e_R is likewise a surjection to $e^{-1}(1)$. These in turn correspond to order relations $e_L(x) < e_R(y)$ from $[p]^+$ to $[q]^+$, and $e_R(y) < e_L(x)$ from $[q]^-$ to $[p]^-$.

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 2.2(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 2.4 [9] 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 2.5 *[9] Positive natural numbers and schedules* $e : p \to q$ *form a category,* Υ*, with composition as in Definition 2.3, and with copycat scheduling functions as identities.*

A proof of Theorem 2.5 does not appear explicitly in [9], though for associativity of composition, reference is made to the merges of sketches from [13]. The theorem is certainly true, but a proof of associativity seems combinatorially cumbersome.

3 Graphical foundation for 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 [14]. 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 must work on the geometry of the plane graphs themselves [6]. The *compactness* in the definitions ensures that 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 3.1 [14] 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 which has fewer than two endpoints in G_0.

Definition 3.2 [14] 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 3.3 Consider Figure 1(a). One way to characterise this as a progressive plane graph 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(b), 1(c) and 1(d) are progressive plane graphs.

In this paper, our primary interest is in the progressive plane graphs that are given by directed paths, since schedule diagrams are paths (see Figures 1(a), 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).

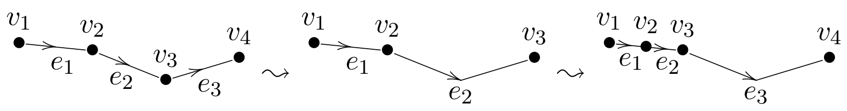


Fig. 2. A node is removed, nodes and edges are relabelled; a node is added, nodes and edges are relabelled.

Definition 3.4 A **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:

$$p_1 = v_1 \tag{S1}$$

$$\text{for each } k, \text{ either } \{p_{2k}, p_{2k+1}\} \subset U \text{ or } \{p_{2k}, p_{2k+1}\} \subset V \tag{S2}$$

- 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 inner 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.

For examples, Figure 1(a) shows a schedule $4 \to 6$, Figure 1(b) shows a schedule $6 \to 4$ 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* [14] which we will slightly adapt here:

Definition 3.5 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' 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

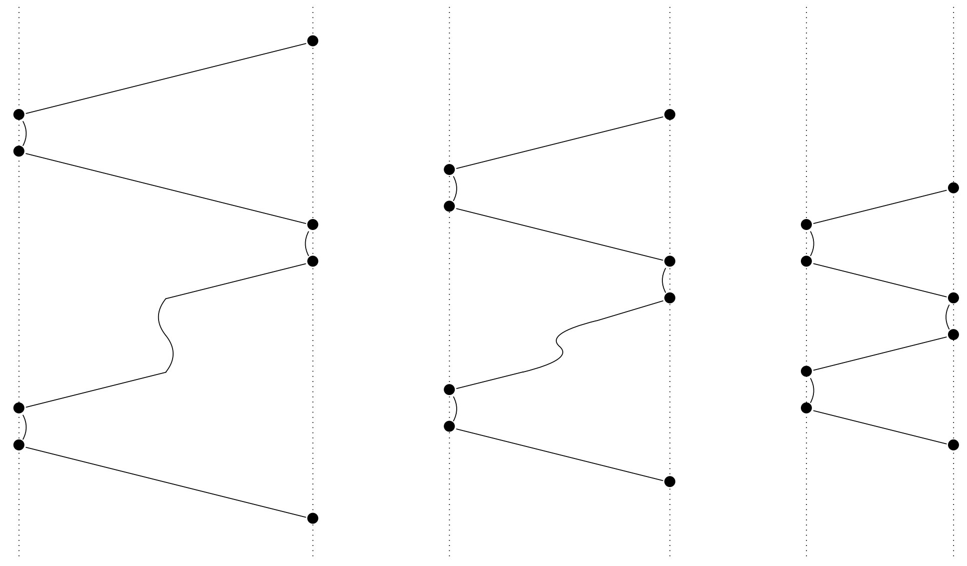


Fig. 3. A "time-lapse" view of a deformation of the schedule in Figure 1(b). Arrowheads used to indicate directions have been omitted for clarity.

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 3.6 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.
- 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.

4 Composition of 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

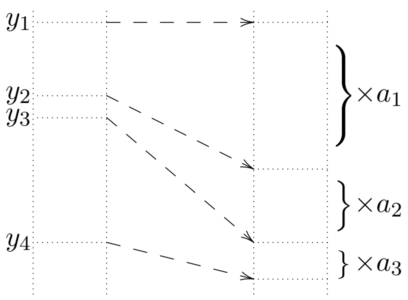


Fig. 4. An illustration of a piecewise vertical scale. The strip on the left has levels y_i marked on it and the strip on the right is the result of applying the scale with factors a_i to each piece, with the dashed straight lines indicating the linear scaling of the segments.

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.

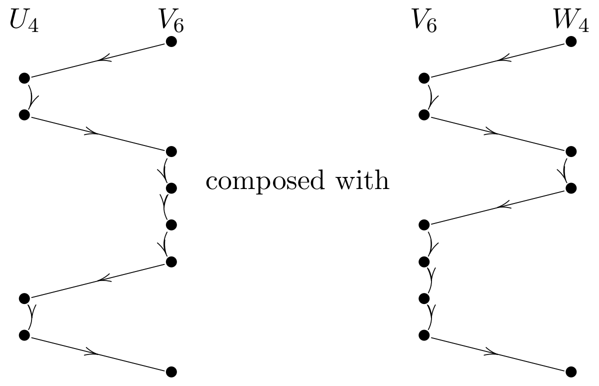
Definition 4.1 Let $S_{m,n} = (U, V, \Sigma, \iota)$ and $S'_{n,r} = (V, W, \Sigma', \iota')$ be two schedules (which we will refer to as S and S' for brevity). We first observe that a pair of translations and of piecewise vertical scalings allow us to assume that $\iota(V) = \iota'(V)$. We call a progressive plane graph formed in this way a (**2-fold**) **composition diagram**; 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 schedules and all edges are progressive, $U + V + W$ may be unambiguously ordered top-to-bottom in the composition diagram, with order-adjacent nodes connected by at least one edge. 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).

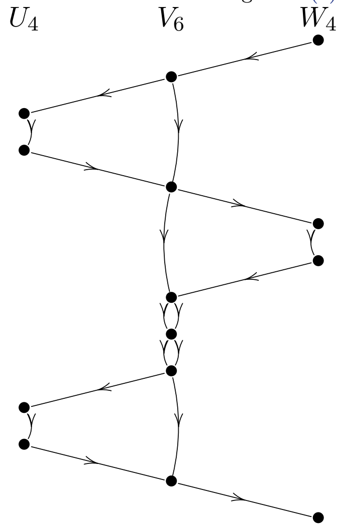
Notice that the edges removed are those comprising an extended "S" shape which begins at the first node of V, continues right with an edge in S' before reaching the next node of V, where it continues left with an edge in S, and continues to alternate in this way.

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.

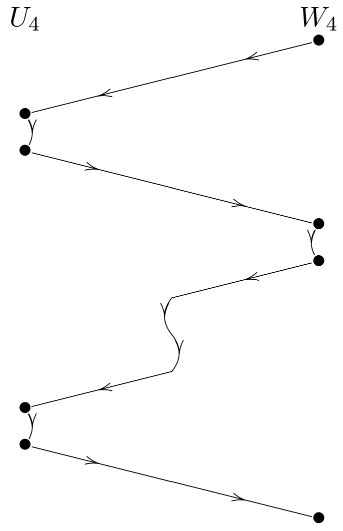
Lemma 4.2 *The path chosen in Definition 4.1 is the unique path (up to deformation) through all nodes of the composition diagram.*



(a) Since they share V_6, these schedules may be composed. These are the schedules from Figures 1(a) and 1(b).



(b) We piecewise vertical scale and translate both schedules so that the two images of V_6 are identified, forming a composition diagram.



(c) We trace along the path in the manner specified in Definition 4.1 and finally declassify the points of V_6.

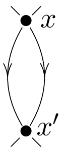
Fig. 5. Composition of schedules.

Proof. Let schedules S and S' be as in Definition 4.1. Consider the composition diagram. 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 6(a) and 6(b).

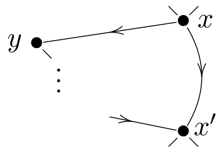
(I) *As in Figure 6(a). Both outward edges from x lead directly to another internal node.* In this case, selecting either edge will yield the same result up to deformation.

(II) *As in Figure 6(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'.



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



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

Fig. 6. Local pictures around internal nodes in composition diagrams

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

Based on Lemma 4.2, 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].

For a full example of composition, we may compose our original example schedule from Figure 1(b) on the left with a schedule $4 \to 6$. This is illustrated again by examples in Figures 5(a), 5(b) and 5(c).

Proposition 4.3 *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 4.1, as does (S1).

(S2) 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. □

Remark 4.4 In the proof of Lemma 4.2, 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 7(b) and 7(c), though in fact they can never occur.

While the reason for this may be derived from (S1) and (S2) by induction, there is also a "local" proof inspired by the colouring (or O/P-labelling) of nodes found in the literature [5,12]. Observe that an arbitrary schedule $S_{m,n} : U \to V$ with path order $U + V = \{p_1, \dots, p_{m+n}\}$ 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 (S1) and (S2), 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 7(a) shows our original schedule from Figure 5(b) decorated in this way. Observe that (S2) 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 $\multimap$ [1]). 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 ◑ and ◐ for the internal nodes of the composition diagram, such as the one in Figure 7(f). Were we to have two cross-schedule edges from the same internal node, it is not the case that both of them could 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 7(d) and 7(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, [1,7].

5 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 5.1 *Composition of schedules is associative.*

Proof. It suffices to show that both possible three-fold compositions are equal. 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 8.

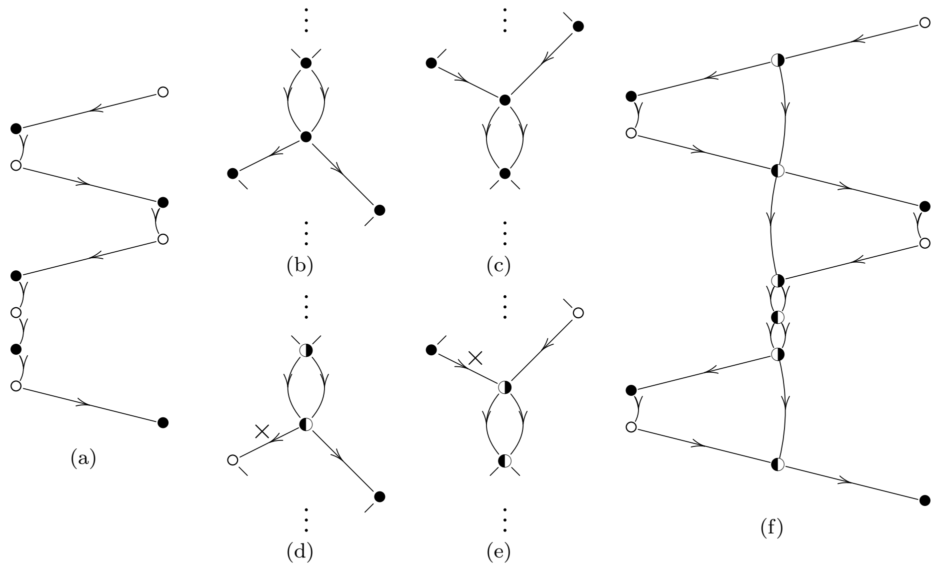


Fig. 7. Colouring of nodes.

$(S \| S') \| S''$ is achieved by first removing the extended "S" 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 "S" 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 4.2, 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. □

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 5.2 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 $P_{2n} = U'_n + U_n$.

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

Graphically, this can be seen in Figure 9. Alternatively, these copycat schedules may be characterised by saying that also $\{p_{2k+1}, p_{2k+2}\} \not\subset U_n$ and $\not\subset U'_n$.

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

Lemma 5.3 *Copycat schedules $I_{n,n}$ are the identities of schedule composition.*

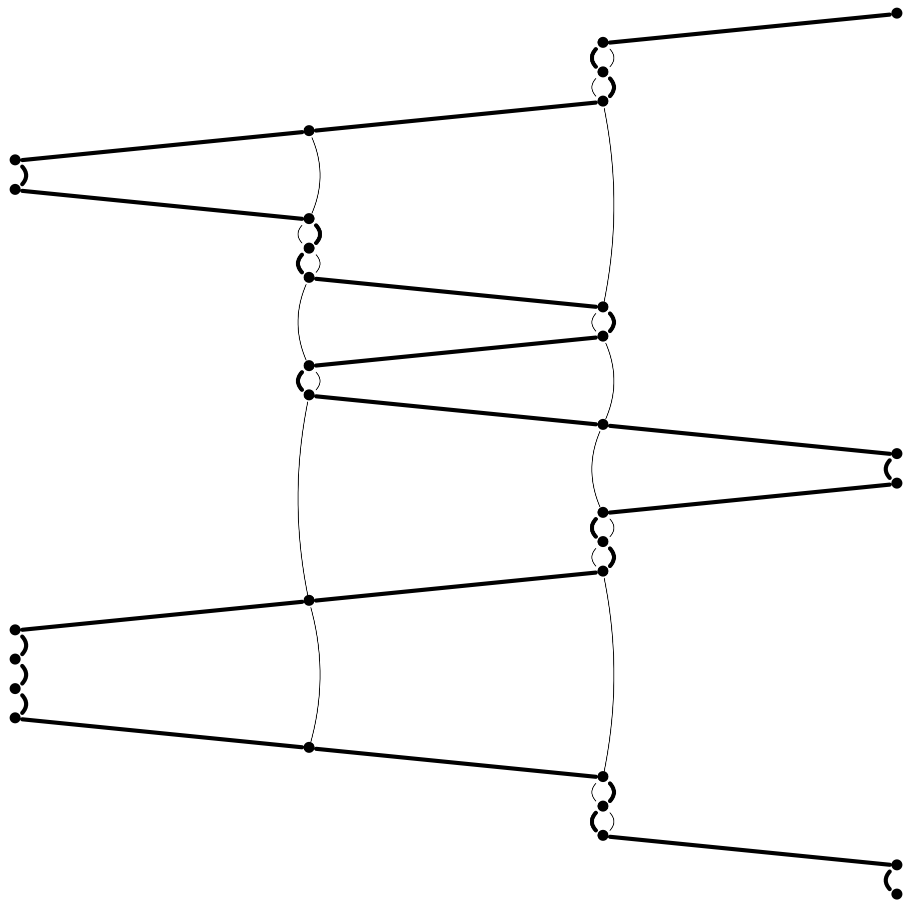


Fig. 8. A three-way composition diagram with composite path highlighted. Note that, since we must always cross between schedules on reaching an internal node, there are no choices to be made.

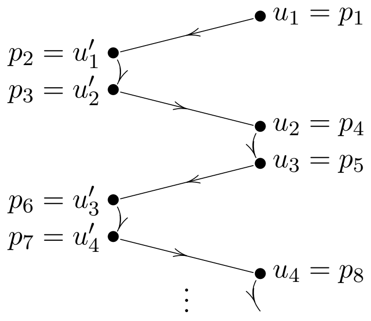


Fig. 9. A prefix fragment of an identity schedule.

Theorem 5.4 *Positive natural numbers, together with the graphical schedules form a category, called $\mathcal{Sched}$, where composition is defined by Definition 4.1 and identities are copycat schedules.*

We will demonstrate that $\mathcal{Sched}$ is equivalent to Υ by exhibiting a functor $\mathcal{Sched} \to \Upsilon$ giving the equivalence. 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. [9], 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 5.5 *C is a functor $\mathcal{Sched} \to \Upsilon$.*

Theorem 5.6 *$C : \mathcal{Sched} \to \Upsilon$ is an equivalence 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 $\pi_1 p_i \neq pi_1 p_{i+1}$) and with a circular arc (of angle less than π) if their first ordinates agree (i.e., if $\pi_1 p_i = pi_1 p_{i+1}$). This manner is similar to the explicit construction of identity schedules in Appendix A.

$CG = \mathrm{id}$ by construction. To see that $GC \cong \mathrm{id}$, 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. $\square$

6 Future work

Following these results it seems natural to generalise this approach to account for other ideas in [9]; to *strategies*, and to *$\otimes$-scheduling functions*. Capturing these similar notions in the diagrams used to represent them will present no challenge. It also seems appropriate to investigate the relationship with the the 2-categorical string diagrams for dialogue games in [18]. Further, we will examine the *pointer functions* and *heaps* — also ubiquitously diagrammatically represented — as their representation also seems well captured by their diagrams [9,10]. In future work we plan to give an account of all of Harmer et al.'s paper in these geometric terms.

Our arguments for key properties have been rendered far simpler through careful definition of the graphical objects under consideration. It may also be the case that these kinds of argument make it possible to define an associative composition for more relaxed notions of scheduling. Some refined notion of type, more sophisticated than just a number, may support a broader class of schedule.

As well as Joyal and Street's framework providing a "realistic" foundation for schedule diagrams, it is also extensible into other classes of planar diagrams [15,17,20,21] and we hope choosing it will provide common ground for future work, perhaps contributing new categories of games and strategies.

References

[1] Abramsky, S., *Semantics of interaction*, in: H. Kirchner, editor, *Trees in Algebra and Programming — CAAP*, Lecture Notes in Computer Science **1059**, Springer Berlin/Heidelberg, 1996 .

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

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

[4] Abramsky, S. and G. McCusker, *Linearity, sharing and state: a fully abstract game semantics for Idealised Algol with active expressions*, Algol-like languages **2** (1997), pp. 297–330.

[5] Abramsky, S. and G. McCusker, *Game semantics*, in: U. Berger and H. Schwichtenberg, editors, *Computational Logic*, NATO ASI Series **165**, Springer Berlin/Heidelberg, 1999 pp. 1–55.

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

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

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

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

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

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

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

[13] Hyland, M. and A. Schalk, *Graphs on games and sequentially realizable functionals. extended abstract*, Proceedings of Seventeenth Annual IEEE Symposium on Logic in Computer Science (2002), pp. 257–264.

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

[15] Joyal, A. and R. Street, *Braided tensor categories*, Advances in Mathematics **102** (1993), pp. 20–78.

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

[17] Melliès, P.-A., *Functorial boxes in string diagrams*, in: Z. Ésik, editor, *Computer Science Logic*, Lecture Notes in Computer Science **4207**, Springer BerlinHeidelberg, 2006 .

[18] Melliès, P.-A., *Game semantics in string diagrams*, Unpublished manuscript, viewed February 16 2012, available at author's website: `http://pps.jussieu.fr/~mellies/tensorial-logic/1-game-semantics-in-string-diagrams.pdf` (2012).

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

[20] Selinger, P., *A survey of graphical languages for monoidal categories*, Lecture Notes in Physics **813**, Springer Berlin/Heidelberg, 2011 pp. 289–355.

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

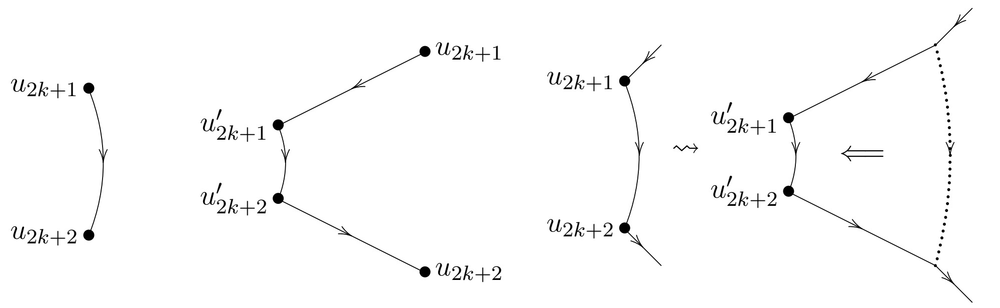
A Appendix: Identity schedules

$I_{n,n}$ 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:

- $U'_n = \{u'_i \mid i \text{ odd} \implies u'_i = (0, -2i),\ i \text{ even} \implies u'_i = (0, 1-2i)\}$
- $U_n = \{u_i \mid i \text{ odd} \implies u_i = (1, -2i),\ i \text{ even} \implies u_i = (1, 1-2i)\}$
- $\Sigma = (\Sigma, U'_n + U_n)$ where

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

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.

Fig. A.1.

Lemma A.1 *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.1(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.1(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.1(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 $S_{m,n} \cong S_{m,n} \| I_{n,n}$. $\square$