Graphical Foundations for Dialogue Games

submitted by

Cai Wingfield

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Computer Science

October 2013

Abstract

In the 1980s and 1990s, Joyal and Street developed a graphical notation for various flavours of monoidal category using graphs drawn in the plane, commonly known as string diagrams. In particular, their work comprised a rigorous topological foundation of the notation.

In 2007, Harmer, Hyland and Melliès gave a formal mathematical foundation for game semantics using a notions they called $\multimap$-schedules, $\otimes$-schedules and heaps. Schedules described interleavings of plays in games formed using $\multimap$ and $\otimes$, and heaps provided pointers used for backtracking. Their definitions were combinatorial in nature, but researchers often draw certain pictures when working in practice.

In this thesis, we extend the framework of Joyal and Street to give a formal account of the graphical methods already informally employed by researchers in game semantics. We give a geometric formulation of $\multimap$-schedules and $\otimes$-schedules, and prove that the games they describe are isomorphic to those described in Harmer et al.'s terms, and also those given by a more general graphical representation of interleaving across games of multiple components. We further illustrate the value of the geometric methods by demonstrating that several proofs of key properties (such as that the composition of $\multimap$-schedules is associative) can be made straightforward, reflecting the geometry of the plane, and overstepping some of the cumbersome combinatorial detail of proofs in Harmer et al.'s terms. We further extend the framework of formal plane diagrams to account for the heaps and pointer structures used in the backtracking functors for O and P.

Acknowledgements

I would very much like to thank my supervisors John Power and Guy McCusker for all their support, advice and encouragement throughout my PhD. They helped make my time at Bath thoroughly rewarding. This thesis would not have been possible without their generous time, support, pep talks, and sage advice.

I would also like to thank my examiners Martin Hyland and Nicolai Vorobjov for taking the time to read and comment on my thesis, and for the discussion and suggestions which helped improve it.

For their interest, patience and thoughtful questions, I am grateful to other members of the Mathematical Foundations group at the University of Bath: Paola Bruscoli, Ana Calderon, Martin Churchill, Pierre Clairambault, Anupam Das, Etienne Duchesne, Alessio Guglielmi, Willem Heijltjes, Jim Laird, Nicolai Vorobjov and David Wilson. From outside Bath, I would like to thank Dan Ghica, Martin Hyland, Alex Jironkin, Paul Levy, Paul-André Melliès, Michele Pagani, Pino Rosolini and Andrea Schalk for stimulating and enlightening conversations. I would also like to acknowledge my use of Aleks Kissinger's *TikZiT* software [Kis07] for drawing diagrams, which saved me a great deal of time in the preparation of this thesis.

Some of the material in Chapter 3 of this thesis has been presented in the following papers, coauthored with my supervisors: [MPW12, MPW]. I would like to thank the anonymous reviewers of [MPW12] for their comments.

I gratefully acknowledge the UK Engineering and Physical Sciences Research Council and the University of Bath, both of whom supported this work financially.

I cannot adequately express my gratitude to my parents, who have always supported and encouraged me, to my sister Jasmine, and of course to Eleanor, for her patience, sympathy, delight and everything.

Contents

1 Introduction **7**

1.1 Graphical notation schemes 7

1.2 Symbolic and graphical notation for monoidal categories 9

1.3 Symbolic and graphical notation for game semantics 13

1.4 Choosing a framework 19

1.5 Outline of thesis 20

2 Progressive plane graphs and string diagrams for monoidal categories **23**

2.1 A symbolic notation for monoidal categories 23

2.1.1 Monoidal signatures and interpretations 23

2.1.2 Free monoidal categories 26

2.2 A graphical notation for monoidal categories: Joyal and Street's string diagrams 27

2.2.1 Formal characterisation of plane graphs 27

2.2.2 Evaluation of graphs 33

2.2.3 Robustness of values 40

2.2.4 A category of diagrams 42

3 A graphical foundation for interleaving structures in games **49**

3.1 Graphical foundations for schedules 49

3.1.1 Combinatorial $\multimap$-schedules 49

3.1.2 Graphical $\multimap$-schedules 50

3.1.3 Composition of $\multimap$-schedules 53

3.1.4 The category $\mathcal{Sched}$ 59

3.2 Games and strategies 64

3.2.1 Games 64

3.2.2 Strategies . . . 65
3.2.3 $\multimap$-schedules and linear function space . . . 66
3.3 The category of games . . . 69
3.3.1 Composition of strategies . . . 69
3.3.2 The category *Game* . . . 71
3.3.3 $\otimes$-schedules and tensor games . . . 73
3.4 Graphical representations of games with more than two components . . . 76
3.4.1 Interleaving graphs . . . 76
3.4.2 Games whose moves are interleaving graphs . . . 85
3.4.3 Folding and unfolding interleaving graphs . . . 86
3.5 Symmetric monoidal closed structure on *Game* . . . 88

4 Pointer diagrams and backtracking **97**
4.1 Pointers and graphs . . . 97
4.1.1 Heaps . . . 97
4.1.2 Heap graphs . . . 98
4.1.3 A partial order on heaps . . . 101
4.1.4 Heap constructions . . . 104
4.2 Categories of heaps . . . 108
4.2.1 Heap functors Ohp and Php . . . 108
4.2.2 Composing and decomposing threads . . . 114
4.2.3 *Oheap*, the category of O-heaps . . . 116
4.3 Further categorical properties . . . 122
4.3.1 Aside: discrete fibrations . . . 122
4.3.2 O-heaps as discrete fibrations . . . 125
4.4 Exponentials . . . 126
4.4.1 The exponential functor ! . . . 126
4.4.2 Backtracking backtracks, ! as a comonad on *Game* . . . 130
4.4.3 Backtracking for P: the functor ? . . . 139
4.4.4 Games constructed with both ? and ! . . . 140
4.4.5 A distributive law of ! over ? . . . 142
4.4.6 The category *Innocent* of games and innocent strategies . . . 147

5 Further directions **149**

5.1 This thesis and future work 149

5.2 Appraisal 149

5.3 Some related work 150

Bibliography **153**

Appendices **161**

A.1 Geometric methods 161

A.2 Monoidal categories 161

A.2.1 Monoidal categories, monoidal functors and $\mathcal{MonCat}$ 161

A.2.2 Monoidal natural transformations and monoidal functor categories 166

A.2.3 A generalisation of Cayley's Theorem for monoids 167

A.3 Schedules and interleaving graphs 171

A.3.1 An example of suitability 171

A.3.2 An example of unfolding 171

A.3.3 Bifuctoriality of $\otimes$ 171

A.4 Pointers 187

A.4.1 Heap graphs in standard configuration 187

A.4.2 The strategy $!\neg : !\mathbb{B} \multimap !\mathbb{B}$ 188

A.4.3 Example of a game $!!G$ 191

Chapter 1

Introduction

1.1 Graphical notation schemes

Mathematicians frequently use sketches and pictures as representations of mathematical constructions that are defined in other terms. This is most frequently found in an informal context, such as a lecture or conversation, where an imprecise representation of an idea is all that is needed to successfully communicate a point. However, it is not uncommon to find diagrams (whether formally introduced or not) present in the definitions and proofs of textbooks and academic papers. This can be productive — what is drawn on the page or the blackboard frequently captures some important aspect of the structure or characteristic of the objects of study.

There are a number of motivations for examining this further. Pragmatically, such diagrammatic methods are useful, since the human brain is in many cases more adept at manipulating certain kinds of geometric objects than at manipulating linear strings of symbols whose precise positioning is crucial. It is often easier to "see" a simple deformation which connects two diagrams than where one can apply a symbolic axiom which connects two expressions.

Moreover, it may be the case that we can capture the essential structure of study while eliminating much of the "bureaucracy" of its notation. The use of graphical notations can make the proofs of some theorems "obvious", where use of purely symbolic, typographical methods may shed little light on the issue.

Diagrammatic arguments have long been used by mathematicians to provide easy communication and convincing arguments. Aspects of the geometry of the plane have been used to capture highly abstract structures in logic, category theory and computer science, as well as in physics, linguistics and elsewhere [Pen71, Str76, Gir87, Pow90, GG08, Mim09].

However, to good mathematics we need to be clear about exactly what we are using, how axioms apply, and precisely what proofs mean. A proof must either be formal (in the sense of logic), or be written in shorthand drawing on the reader's intuitions to sound convincingly like it stands for a formal argument. This can become problematic

when mathematicians employ diagrammatic methods without appropriately rigorous characterisations and theorems about their geometric constructions. The problem is that the full mathematical justifications for these drawings and diagrams are frequently elided. Diagrams may be defined to be "piecewise linear" or use "the obvious definitions", though both of these leave something to be desired. The first, as this is not what people actually draw when they draw diagrams, and so potentially the definition is not capturing the subtleties of the methods mathematicians actually use. The second, as the definitions them selves are frequently *not* obvious, and seemingly insignificant differences in definitions can have unexpected results. It is vitally important to know whether a particular diagram is or is not an example of the structure in question, or when two diagrams are to be considered the same. For instance, [Gol74] shows a set of examples demonstrating that two possible characterisations of braid deformation are not equivalent, a fact which was not known to Artin. Proofs of the deceptively "evident" Jordan curve theorem are far from obvious, and the generalisation to curves which are not finitely piecewise linear requires careful consideration [Hal07].

As we have mentioned, sketches have been employed informally in many mathematical settings, but there have also been successful formal treatments. One seminal such case was Joyal and Street's development in the 1980s and 1990s of a graphical notation for monoidal categories (also called tensor categories), now commonly referred to as *string diagrams* [JS88, JS91]. We will see this in greater detail later, and a full description is given in Chapter 2. Joyal and Street's graphical language for monoidal categories easily extends to braided monoidal categories and symmetric monoidal categories [JS93], and has since been extended further to graphical languages for compact closed categories and tortile tensor categories [FY89, Shu94], traced monoidal categories [JSV96], and beyond [JS92, Mel06, BS11]. Peter Selinger has produced an excellent survey of graphical languages for monoidal categories [Sel11].

Applications of diagrammatic methods derived from monoidal categories have also proved strong in particle physics and quantum mechanics [Abr08, BS11, CP11]. In these cases, a point worth highlighting is that the diagrams do not represent physical objects, but formal expressions. They make crucial use of the geometry of the plane, especially in the case of string diagrams for monoidal categories. This will be discussed further in Chapters 2 and 3.

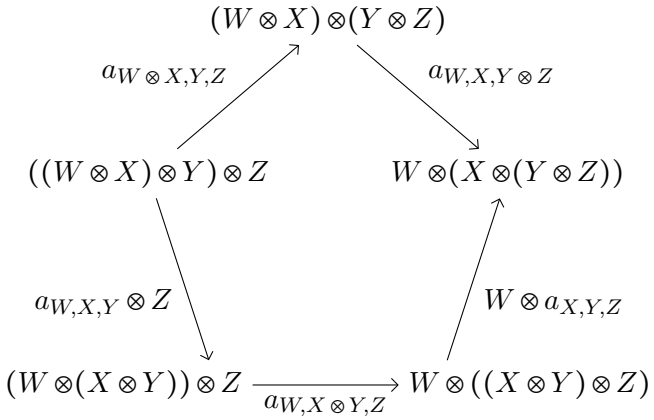
In this thesis we will see detailed accounts of two examples of diagrammatic reasoning employed in the mathematical foundations of computer science. We wish our definitions to be as close as possible to what researchers already use — and importantly, of what they think — when they draw diagrams. We mean to be precise, avoiding appeals to "obvious" yet unstated facts. The power of the methods we described will come from elementary facts about plane geometry. We will make heavy use of *compactness*. When graphs are compact subsets of the plane, they may be finitely decomposed in such a way as to avoid many of the potential pathological cases we might otherwise have to worry about. The facts we will use regarding compactness are elementary and can be found in any decent undergraduate textbook on topology, such as [Arm83], but we restate them in Appendix A.1.

1.2 Symbolic and graphical notation for monoidal categories

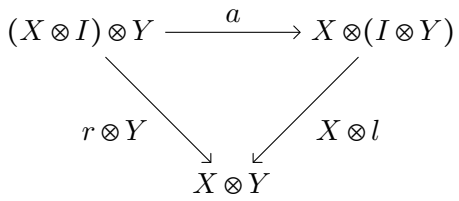
A monoidal category [Mac97] is a category $\mathscr{V}$, a functor $\otimes : \mathscr{V} \times \mathscr{V} \to \mathscr{V}$, a distinguished object $I \in \mathscr{V}$ and natural isomorphisms

$$\begin{aligned} a_{X,Y,Z} : (X \otimes Y) \otimes Z &\to X \otimes (Y \otimes X) \\ l_X : I \otimes X &\to X \\ r_X : X \otimes I &\to X \end{aligned}$$

such that



(the "pentagon axiom") and



(the "triangle axiom") commute as diagrams of natural transformations.

Important examples of monoidal categories include the category of sets together with the standard set product, the category of sets and relations together with the standard product of relations, and the category of endofunctors for a category $\mathscr{C}$ together with composition of functors. (A full description of monoidal categories and some important relevant theorems are given in Appendix A.2.)

Monoidal categories are a powerful concept, but calculating with them is far from trivial. Indeed, MacLane's early investigations [Mac63, Mac97] contained the further axiom

$$l_I = r_I : I \otimes I \to I$$

which was later shown by Kelly to be derivable from the other axioms [Kel64]. With an appropriate geometric foundation, some of these issues cease to be so problematic.

Before we can understand what is meant by the phrase "graphical notation", we must

first appreciate what a "symbolic notation" entails. Informally speaking, when we write down an expression such as "$f : X \to Y$" in a monoidal category, we may be using the symbols "X" and "Y" as "object variables", in the sense that they may represent arbitrary objects from the category which may have some unspecified $\otimes$-structure. In this case, the symbol "f" is understood to represent an arbitrary arrow in the category, which may or may not have a $\otimes$ or compositional structure, but which must at least have source X and target Y. In this way, "$f : X \to Y$" is used to describe generalised structure as well as specific instances of morphisms.

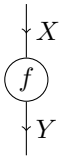
More specifically, we need a way of generating symbolic expressions, which for us comes in the form of a *monoidal signature* and a way of *interpreting* those expressions in a monoidal category. We will explain this in fuller detail in Chapter 2. Furthermore, when we say that an equation between two morphisms of a monoidal category holds *in general*, we mean that it holds up to a unique isomorphism which is composed only of components of a, l and r. In other words, it holds up to unique isomorphism in a *free monoidal category* (see Appendix A.2).

A graphical notation, then, is system for generating graphs which have some compositional structure, and way of interpreting those graphs as expressions in a monoidal category. We will also need an appropriate characterisation of what it means for two graphs to be "the same".

Joyal and Street's *string diagram* notation for monoidal categories uses graphs in the plane to represent morphisms in a monoidal category. Objects and morphisms of the category are represented by the edges and nodes of the graph, which is then built compositionally. Informally, we precis the notation here:

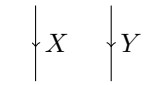
Objects X we denote by strings:

and morphisms $f : X \to Y$ by nodes connected by strings

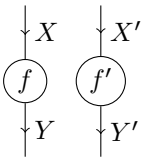


In this sense, our diagrams are to be read top-to-bottom, though different sources write them in different orientations.

The tensor product on objects and arrows is horizontal juxtaposition:

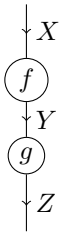


is $X \otimes Y$



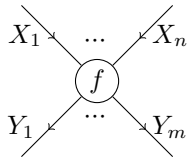
is $f \otimes f' : X \otimes X' \to Y \otimes Y'$

and composition is given by vertical glueing:



is $g \circ f : X \xrightarrow{f} Y \xrightarrow{g} Z$

General arrows $f : X_1 \otimes \cdots \otimes X_n \to Y_1 \otimes \cdots \otimes Y_m$ look like



We identify identity arrows id_X with their objects X, as is common practice with the symbolic notation. The object I and components of a, l and r are invisible.

An example of such a diagram taken from [JS91] is shown in Figure 1, which, for objects A, B, C, D and morphisms

$$a : A \to B \otimes B \qquad b : B \to C \otimes D \qquad c : B \otimes C \to C \qquad d : D \otimes C \to D$$

denotes the composition

$$(B \otimes C \otimes d) \circ (B \otimes c \otimes D \otimes C) \circ (a \otimes b \otimes C)$$

We consider that two diagrams are "the same" if one can be achieved from the other via a planar isotopy — we are allowed to move the nodes and deform the edges, so long as the connections are unchanged, no nodes or edges pass through each other, and at each stage we are left with a valid diagram.

This allows us to perform calculations using the diagrams in ways far easier than by applying the monoidal axioms to symbolic strings. Consider the following word problem

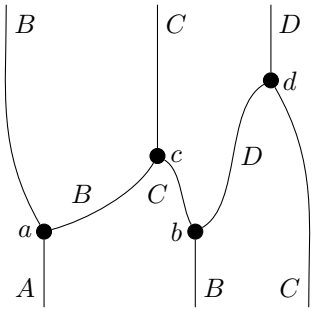


Figure 1: An example string diagram from [JS91].

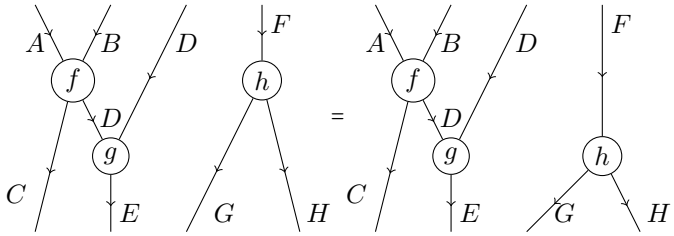


Figure 2: The word problem (1.1) reformulated in the graphical notation. The height of the nodes is important in each diagram's correspondence to its respective symbolic expression.

for arrows. Given

$$f : A \otimes B \to C \otimes D, \qquad g : D \otimes D \to E, \qquad h : F \to G \otimes H$$

how can we tell whether or not the equality of arrows

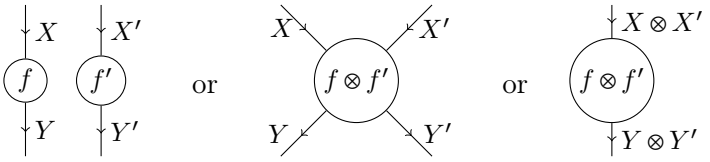
$$(f \otimes D \otimes h) \circ (C \otimes g \otimes G \otimes H) = (f \otimes D \otimes F) \circ (C \otimes g \otimes h) \tag{1.1}$$

holds *in general*? In fact it does, but it's not obvious.

Translating the problem into the graphical language, the word problem (1.1) is reformulated as in Figure 2, so that, if this is indeed a valid notation, the equality of these as arrows $A \otimes B \otimes D \otimes F \to C \otimes E \otimes G \otimes H$ holds as their expressions "look" the same in the way informally described above.

As mentioned above, the notation for (for instance) a morphism $f : X \to Y$ may or may not have a $\otimes$-decomposition and whether or not we wish to draw attention to this is reflected in whether or not we make this explicit. In this sense we may treat the symbols f, X, Y as representing larger tensor products. In our new graphical notation we also want to capture this versatility. Therefore, we have these three ways of writing an arrow $f \otimes f' : X \otimes X' \to Y \otimes Y'$, depending on whether the variables into which the values have

been substituted have the explicit tensor structure:



1.3 Symbolic and graphical notation for game semantics

A *denotational semantics* for a programming language is a way of modelling code as a mathematical object, in a compositional way. It facilitates certain styles of formal reasoning about programs; behavioural equivalence and property satisfaction, for example.

Game semantics is a style of denotational semantics for programming languages which, in the 1990s and 2000s, provided the first fully abstract model of the language PCF [Nic94, AJM00, HO00]. Over recent decades, game semantics has become one of the standard forms of semantics for programming languages [AJ94, AM97, Mel], and has allowed many structures to be brought to bear on a variety of problems in programming language semantics [AHM98, HM99, Lai01, Ghi09, CM10, Chu11].

Game semantics models the interaction of a program with its environment as a formal game played between two players. Roughly speaking, the *opponent*, O, acts as the environment and the *player* (or *proponent*), P, acts as the program. There are many excellent tutorials and introductions to game semantics — recommended are [Hyl97, AM99, Sch01, Har04, Cur06]. This thesis will aim to be self-contained, providing all required definitions, but we refer the reader to these sources to gain further understanding of the rudiments and full motivations for game semantics.

In 2007, Harmer, Hyland and Melliès gave a formal mathematical foundation for game semantics [HHM07]. 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. Harmer et al. then define 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 $e(2k+1) = e(2k)$ for $k > 0$. 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`. `1001111001` and `1001100001` are schedules $\{1, \ldots, 10\} \to \{0, 1\}$.

Researchers frequently describe schedules on the page or blackboard using a graphical representation [AM99, Har00, Hyl97, Hyl07, HO00]. In keeping with the intuitions of game semantics as involving a game which is *played*, diagrams are used to indicate the passing of control between O and P, and between different components of the game.

Figure 7(a) has graphical representations of the two schedules given above as binary strings. Composites are also typically described graphically, in a manner implied by the

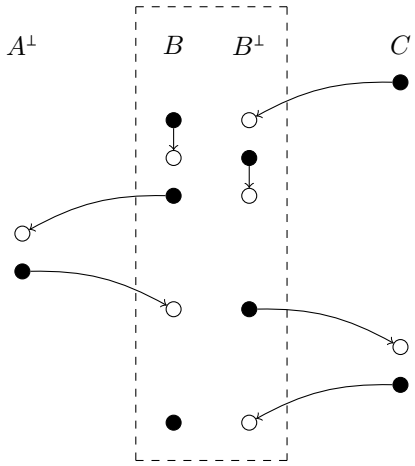


Figure 3: A diagram depicting composition of strategies, from [Hyl97].

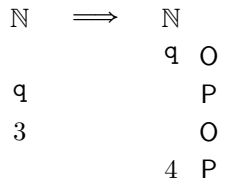


Figure 4: A pictoral representation of a play from [AM99].

description of schedules as pairs of order relations in [HHM07]. For example, Figure 3 is an example from the literature used to explain composition of strategies. Figures 7(b) and 7(c) describe the composite of the two example schedules above.

While many draw precisely such diagrams as these, another common practice 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 horizontal and vertical position, but no actual lines drawn. Consider, for example, Figure 4, which is a reproduction from [AM99], or Figure 5, depicting composition of schedules. Such pictorially laid-out plays are still really examples of the same class of schedule diagrams, and the graphical definitions of schedules in this thesis encompass 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.

Furthermore, plays of compound games with more than two components are often laid out in a similar style, using graphs we call *interleaving graphs*. For example, Figure 6 shows an example of a play for a game of a higher type taken from the literature. Further examples of such diagrams can be seen in Figures 45(b) and 46.

Graphical descriptions of schedules in game semantics are used because they seem in many cases to better capture researchers' intuitions about the structure of games than the

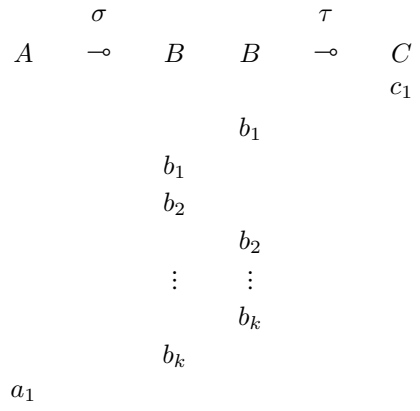


Figure 5: A diagram depicting composition of strategies, from [Abr96].

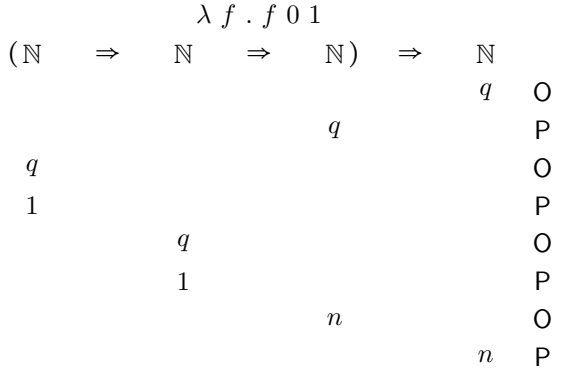
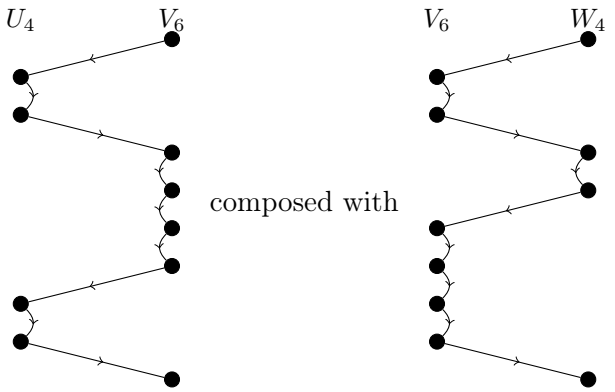
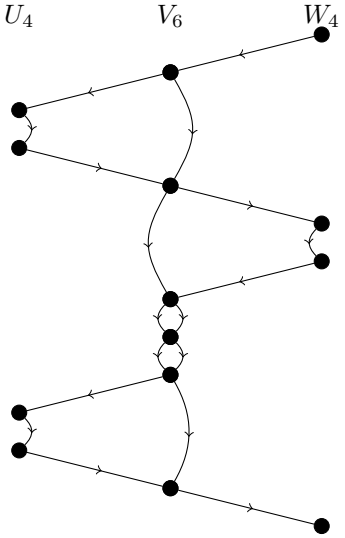


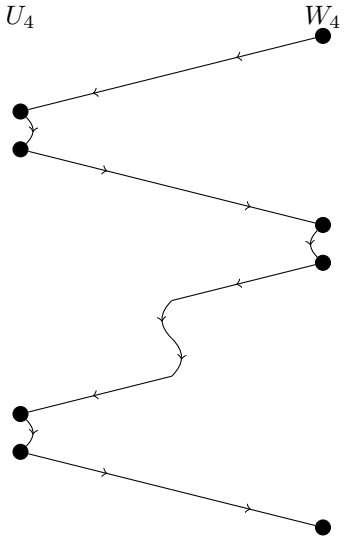
Figure 6: A diagram depicting a play of a higher type, from [AM99].



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



(b) The intermediate *composition diagram*.



(c) The result of the composition.

Figure 7: Example schedules and schedule composition.

combinatorial definitions. Categorical properties which relate to the compositional nature of schedules seem obvious or inescapable when graphical arguments are used, whereas in terms of the original definitions, proofs are often complicated and unenlightening, and are frequently therefore omitted from papers.

This situation gives rise to the natural question of whether the proofs of key properties of schedules can be made simpler and closer to intuitive arguments by redefining them as diagrams, in a similar way to string diagrams mentioned above. It is this which provides a starting point for the main work of this thesis.

We will begin by characterising those pictures which arise as schedule diagrams, formally prove that Harmer et al.'s combinatorial definition and our geometric definition agree, and define a graphical composition of schedules which also agrees. Our graphical definitions are set in the framework of Joyal and Street's string diagrams. It is also worth possible to characterise schedules using the free adjunction $\mathcal{A}dj$ [SS86], cf. Melliès' 2-categorical string diagrams for adjunctions [Mel12b].

Harmer et al. also assert that schedule composition yields a category, but they do not include a proof in [HHM07]. In geometric terms, the proof of associativity (the key property) will follow directly from the natural associativity of *juxtaposition in the plane*.

As well as schedules being used to represent plays in games with two "components", plays in games with more than two components are also pictorally represented in a similar way, using more complicated *interleaving graphs*. Consider Figure 8, for example, which is a reproduction of Figure 5 from [Hyl97]. It also seems natural to encompass these useful diagrams in our definitions, and describe how they are related to graphical schedules, and so how informal arguments using them can be made precise. As with schedules, equivalent examples without lines drawn are also present in the literature, and are equally covered by our discussions.

A *heap* is a way of structuring ordered data. It is defined to be a partial function $\phi : \{1, \cdots, n\} \to \{1, \cdots, n\}$ such that $\phi(i) < i$ wherever $\phi(i)$ is defined, and this notion can be generalised to a *heap structure* on any finite ordered set. In this way, a heap structure is equivalent to a forest of directed trees where paths down branches are strictly increasing.

Heaps provide a structure on games allowing us to equip $\mathcal{G}ame$, the category of games and strategies, with a linear exponential comonad called ! [HHM07, HO00]. A play of the game $!A$ is a forest whose rooted paths are plays of A.

Heaps and heap structures in game semantics are often denoted diagrammatically [DH01, CH10, HO00]. For example, Figure 9 is a reproduction of Figure 1 of [HO00]. The forest structure of a heap is inherently planar, however it is common practice to encode the ordering on the nodes of the forest with their position in the diagram, sometimes yielding pictures which do not exhibit the planarity of the heap structure, but do nonetheless use the geometry of the plane to effect.

It is these more general diagrams which we wish to consider, as they encode ordering in the same way as schedule diagrams, allowing us to describe certain important operations diagrammatically. Consider, for example, Figure 10, which is a reproduction of Figure 4 of [HO00].

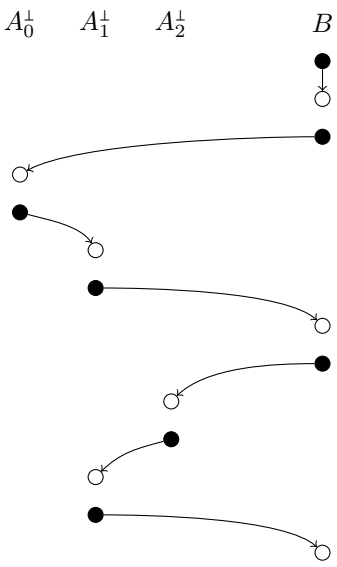


Figure 8: Figure 5 of [Hyl97] shows an interleaving graph; a schedule with more than two "sides".

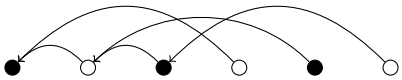


Figure 9: A reproduction of Figure 1 from [HO00] (some labels omitted) shows a graphical representation of a heap used to describe a pointer structure.

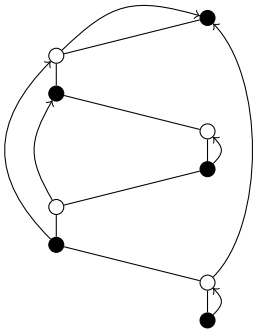


Figure 10: A reproduction of Figure 4 of [HO00] (some labels omitted).

We will characterise these diagrams as graphs in the plane which permit finitely many transverse crossings of edges. We will follow the examples set by knot diagrams [Rol76, Cro04] and see these as shadows from graphs in a higher-dimensional space.

1.4 Choosing a framework

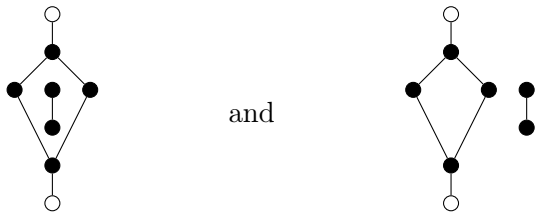
It may perhaps be unclear to some readers exactly why our chosen setting — embedding Hausdorff spaces in $\mathbb{R}^2$ — has been considered the most appropriate when other, less complicated notions of graph also exist. No particular choice in the definitions of graph is absolutely essential to the material of this thesis, but the choices we have made provide us some specific benefits.

We could rely merely on plan*ar* graphs, rather than graphs actually in the plane. After all, $\mathbb{R}^2$ has a fairly rich topology and we do not make use of all its nuances. There are a number of reasons, however, why we have used *plane* graphs.

We note that researchers, who are typically limited to writing on paper, boards or screens, have found their geometries sufficient to encode combinatorial structure. Syntactically complex expressions are represented, either precisely or up to unique isomorphism, in the simple arrangement of points and lines in the plane. There seems to be value in acknowledging and characterising *what researchers actually draw*.

In the plane, notions such as "left", "right", "inside", "outside" (and so on) have real meaning. One could deal with planar graphs, and indicate such properties by using labels, colours, or some additional decorations on the planar graph, but this would require us to perform some reindexing between combined components. We would also be required to prove numerous ancillary lemmata (such as that *something on-the-right of something on-the-right is still on-the-right*). Using the plane, these come for free. This convenience can be seen in action in this thesis.

In particular, the Jordan curve theorem [Veb05] means that in the plane, every closed loop separates an *inside* region from an *outside* region. In terms of the string diagrams for monoidal categories discussed in Chapter 2, this distinction matters. For example,



are homeomorphic *as graphs*, but not isotopic as subsets of the plane (i.e. not related via deformation in the sense of Definition 17). Expressions in monoidal categories formed by adding labels to these diagrams would *not* be isomorphic due only to the monoidal axioms. Again we could perhaps encode planar regions in a planar graph by some cyclic labelling on nodes or edges, but this would add significant complication, and our

representations would begin to move further away from what researchers would end up drawing. The plane captures this crucial information in an elegant way.

Defining the graphs themselves also presents us with some choies. The reader may be used to thinking of a graph as something resembling a set V of *vertices* together with a multiset E of *edges* consisting of ordered pairs of elements of V. However, if we are to have some notion of a graph *in the plane*, we would need to begin to associate each edge with a copy of the unit interval $(0, 1)$, together with an embedding of that interval in the plane satisfying certain conditions, or something similar. Some of the key proofs in this thesis involve cutting and glueing, which become more complicated in this setting.

Of course, each possible choice could be made to work, and we have opted here to put the conceptual difficulty into the early definitions of graph and planar embedding, so that the later arguments can be left cleaner and more intuitive.

It is also worth mentioning that this $\mathbb{R}^2$-based framework from [JS91] is also highly extensible, in the sense that it can be refined and augmented for other settings. The work done by Joyal and Street provides a firm foundation for much other work exploring graphical languages for monoidal categories. In [JS93], Joyal and Street extend their graphical language to *braided monoidal categories*, with the braiding isomorphisms represented by specially labelled nodes

The graphical language in the plane then directly becomes identical to the *braid diagrams* common in knot theory, and provide analogous theorems for deformations of braided strings, and operations on the braid diagrams, via a theorem of Reidemeister [Rei27]. The braid diagrams are specifically *in the plane*, and (though [JS93] does not do so) can easily be given by slight decorations of progressive plane graphs, as mentioned.

This can be developed further into graphical languages for symmetric monoidal categories, traced monoidal categories, monoidal categories with duals, twists, and so on [JS92, Shu94, Sel11].

1.5 Outline of thesis

The content and contributions of this thesis are as follows:

In Chapter 2 we give a detailed account of Joyal and Street's geometric foundation of string diagrams as certain compact Hausdorff spaces embedded in the plane. We show how these diagrams comprise a valid notation for expressions in a monoidal category, and reproduce the proof of [JS91] that the category of diagrams up to deformation is the free (strict) monoidal category on the labels.

In Chapter 3 we give a graphical foundation for the oft-drawn schedule diagrams and diagrammatically depicted plays. This foundation is in similar terms to Joyal and Street's, and similarly uses compactness to allow piecewise consideration of the diagrams. We give a novel proof that the composition of $\multimap$-schedules is associative and thus exhibit a category of schedules.

We use graphical $\multimap$-schedules to give a new description of the category $\mathcal{G}\mathit{ame}$ of games and strategies, where plays consists of labelled schedules. We describe a monoidal structure on $\mathcal{G}\mathit{ame}$ using a graphical $\otimes$-schedule. We give a full graphical characterisation of plays in games with more than two components, and precisely describe how this is related to the graphical representation of their binary interleaving structures. This allows us to give extremely succinct proof of several key categorical properties, for example of the symmetric monoidal closed structure on $\mathcal{G}\mathit{ame}$.

In Chapter 4 we give, from a similar graphical foundation, an account of heap graphs and the pointer diagrams used in game semantics. We discuss the category $\mathcal{O}\mathit{heap}$ of O-heaps and heap functors Ohp and Php. We use diagrams familiar to game semanticists to give a endofunctor ! on $\mathcal{G}\mathit{ame}$ which allows O to backtrack, and an endofunctor ? which allows P to backtrack. We show how !, ?, $\multimap$ and $\otimes$, described in terms of diagrams, interact. Following the example of [HHM07], we show how ! gives a comonad on $\mathcal{G}\mathit{ame}$ and how ? gives a monad, and give a distributive law λ of ! over ?, which is relevant to an understanding of games and innocent strategies.

In Chapter 5 we give a brief retrospective overview of the thesis. We comment on where it was successful and where less so. We end by comparing and contrasting it with other related work.

Chapter 2

Progressive plane graphs and string diagrams for monoidal categories

We begin by considering what is meant by a formal notation for monoidal categories. In what follows e will look in some detail at the graphical framework of progressive plane graphs. We will see how progressive plane graphs may be interpreted as formal expressions in a monoidal category, and how this may be used to aid calculations.

2.1 A symbolic notation for monoidal categories

Before developing a graphical notation for monoidal categories, we must first examine the more conventional symbolic notations, and see what form a notation must take.

The terms in which we discuss notation are essentially those from [JS91, JS93] and further explained in [Sel11], and are standard. Notes on the categorical background are in Appendix A.2.

2.1.1 Monoidal signatures and interpretations

Mathematical formulae are commonly represented with sequences of various symbols, including letters, connectives and brackets. The precise form of these will of course vary depending on the mathematical context, but we typically want the compositional rules governing well-formed formulae to bear some relationship to the structures at hand.

One way to characterise well-formed formulae is using a *signature*, a collection of sets of usable symbols, together with some functions describing how they can be combined. In the case of expressions in monoidal categories, we have a *monoidal signature*.

In order to let an expression from a monoidal signature refer or stand for a something in a particular monoidal category, we require a particular *interpretation* of the signature in that category. We may also speak about monoidal categories "in genaral", which requires a *free* construction.

Definition 1 ([JS91], p. 68; [Sel11], p. 12.)**.** A **monoidal signature**, Σ, is a pair (Σ_0, Σ_1) consisting of:

- A set Σ_0 of **object variables**. From this we may create a set $\hat{\Sigma}_0$ of **object terms**, which are exactly the binary $\otimes$-words in elements of Σ_0 and the special symbol $I \notin \Sigma_0$.
- A set Σ_1 of **morphism variables** which admit a pair of functions

  $$\mathrm{dom}, \mathrm{cod} : \Sigma_1 \to \hat{\Sigma}_0$$

  called **domain** and **codomain**, respectively.

We may write $f : X \to Y$ to indicate that $f \in \Sigma_1$, that $\mathrm{dom}(f) = X$ and that $\mathrm{cod}(f) = Y$ for $X, Y \in \hat{\Sigma}_0$.

A monoidal signature, then, describes all general well-formed expressions which may be written to describe objects and morphisms in a monoidal category. Such expressions are only useful if we can understand them to mean something in a monoidal category of choice.

Definition 2 ([Sel11], p. 12.)**.** An **interpretation** of a monoidal signature Σ in a monoidal category $\mathscr{V}$ is a pair of functions $\iota_0 : \Sigma_0 \to \mathrm{ob}\,\mathscr{V}$ and $\iota_1 : \Sigma_1 \to \mathrm{mor}\,\mathscr{V}$.

Observe that ι_0 extends uniquely to a function $\hat{\iota}_0 : \hat{\Sigma}_0 \to \mathrm{ob}\,\mathscr{V}$ satisfying

$$\hat{\iota}_0(X \otimes Y) = \hat{\iota}_0(X) \otimes \hat{\iota}_0(Y)$$

$$\hat{\iota}_0(I) = I$$

We require that

$$\iota_1(f) : \hat{\iota}_0(\mathrm{dom}\, f) \to \hat{\iota}_0(\mathrm{cod}\, f)$$

We write such an interpretation as $\iota : \Sigma \to \mathscr{V}$. We also frequently (ab)use "ι" to refer to each of $\iota, \iota_0, \hat{\iota}_0$, and ι_1, when to do so would not lead to ambiguity.

By interpreting an expression from a monoidal signature we can refer to objects and arrows in a monoidal category using symbols.

Remark 3. Some sources (e.g. [JS91]) don't specify brackets on tensor products because of the coherence theorem. Since $\mathscr{V}$ is equivalent to a strict monoidal category, a word from $\hat{\Sigma}_0$ gets sent by ι into a clique of identically ordered binary tensor words in $\mathscr{V}$. (Recall that a *clique* in a category is a collection of pairwise isomorphic objects in the category.)

However, in general we will wish to consider interpretations in specific non-strict monoidal categories, such as $\mathcal{S}et$. By omitting brackets, we would only be able to consider some strict monoidal category which is equivalent to $\mathcal{S}et$, which is less appealing.

We will shortly see that the possible interpretations of a monoidal signature inside a monoidal category have a categorical structure, which we will eventually use to consider free monoidal categories.

Definition 4. Let Σ be a monoidal signature and $\mathscr{V}$ be a monoidal category, and let $\iota, \kappa : \Sigma \to \mathscr{V}$ be two interpretations of Σ in $\mathscr{V}$. A **morphism of interpretations** is a Σ_0-indexed family of morphisms $\upsilon_X : \iota X \to \kappa X$ in $\mathscr{V}$. In such an instance, we write $\upsilon : \iota \to \kappa$.

Observe that this uniquely extends to a $\hat{\Sigma}_0$-indexed family (which we also call υ_X) such that for each $f : X \to Y$ in Σ_1, the square

$$\begin{array}{ccc} \hat{\iota}(X) & \xrightarrow{\iota f} & \hat{\iota}(Y) \\ {\scriptstyle \upsilon_X}\downarrow & & \downarrow{\scriptstyle \upsilon_Y} \\ \hat{\kappa}(X) & \xrightarrow[\kappa f]{} & \hat{\kappa}(Y) \end{array}$$

commutes. Morphisms of interpretations compose as morphisms in $\mathscr{V}$.

Routine calculation shows that, for a given monoidal signature Σ and monoidal category $\mathscr{V}$, composition of morphisms of interpretations is associative, and has an identity (given by the identity function on Σ_0).

From this we may define a category:

Definition 5. For a monoidal signature Σ and monoidal category $\mathscr{V}$, the **category of interpretations** of Σ in $\mathscr{V}$ is the category whose objects are interpretations $\iota : \Sigma \to \mathscr{V}$, and whose arrows $\iota \to \kappa$ are morphisms of interpretations $\upsilon : \iota \to \kappa$. We denote this category as $[\Sigma, \mathscr{V}]_{In}$.

Given an interpretation $\iota : \Sigma \to \mathscr{V}$ in a monoidal category, we may *compose* this with a strong monoidal functor $F : \mathscr{V} \to \mathscr{W}$ to get a new interpretation $\Sigma \to \mathscr{W}$, which we write $F \circ \iota$. $F \circ \iota$ interprets object variables X in Σ_0 as

$$(F \circ \iota)(X) = F(\iota X)$$

and given a morphism variable $f : X_1 \otimes X_1 \to Y$ in Σ_1 with $X_1, X_2, Y \in \Sigma_0$, $F \circ \iota$

interprets f as

$$\begin{array}{ccccc}
X_1 \otimes X_2 & & \xrightarrow{\qquad\qquad f \qquad\qquad} & & Y \\
\widehat{(F \circ \iota)}_0(X_1 \otimes X_2) & & \xrightarrow{\qquad (F \circ \iota)_1(f) \qquad} & & (F \circ \iota)_0 Y \\
\| & & & & \| \\
\widehat{(F \circ \iota)}_0(X_1) \otimes \widehat{(F \circ \iota)}_0(X_2) & & & & \| \\
\| & & & & \| \\
F(\hat{\iota}_0(X_1)) \otimes F(\hat{\iota}_0(X_2)) & \xrightarrow[\phi_2]{} & F(\hat{\iota}_0(X_1) \otimes \hat{\iota}_0(X_2)) & & \| \\
& & \| & & \| \\
& & F(\hat{\iota}_0(X_1 \otimes X_2)) & \xrightarrow[F(\iota_1(f))]{} & F(\hat{\iota}_0(Y))
\end{array}$$

and interprets other morphism variables by induction on their forms.

This construction provides an infix functor

$$\circ : [\mathscr{V}, \mathscr{W}]_{St} \times [\Sigma, \mathscr{V}]_{In} \to [\Sigma, \mathscr{W}]_{In}$$

The action of $\circ$ on morphisms is as follows: A morphism in $[\mathscr{V}, \mathscr{W}]_{St}$ is a monoidal natural transformation $\theta : F \Rightarrow G : \mathscr{V} \to \mathscr{W}$ and a morphism in $[\Sigma, \mathscr{V}]_{In}$ is a morphism of interpretations $\upsilon : \iota \to \kappa : \Sigma \to \mathscr{V}$. The morphism of interpretations $\theta \circ \upsilon$ is a family of arrows $(\theta \circ \upsilon)_X$ in $\mathscr{W}$ which are each the composition

$$\begin{array}{ccc}
F(\iota X) & \xrightarrow{(\theta \circ \upsilon)_X} & G(\kappa X) \\
& {\scriptstyle F\upsilon_X} \searrow \quad \nearrow {\scriptstyle \theta_{\kappa X}} & \\
& F(\kappa X) &
\end{array}$$

We may use this notion to describe a free monoidal category on a monoidal signature.

2.1.2 Free monoidal categories

Definition 6 ([JS91], Definition 1.5; [Sel11], p. 13.)**.** A monoidal category $\mathscr{F}$ is said to be **free on the monoidal signature** Σ if there's an interpretation $\iota : \Sigma \to \mathscr{F}$ such that

$$- \circ \iota : [\mathscr{F}, \mathscr{V}]_{St} \to [\Sigma, \mathscr{V}]_{In}$$

is an equivalence of categories for each monoidal category $\mathscr{V}$.

In other words, the monoidal category $\mathscr{F}$ is free on Σ (and may also be called "*the* free monoidal category") if we have an interpretation of Σ in $\mathscr{F}$, and any other interpretation κ of Σ in any monoidal category factors through ι via a strong monoidal functor which is unique up to unique monoidal isomorphism.

In this sense, a free monoidal category is the "most general" reflection of the structure of a monoidal signature — our symbolic notation scheme — within a monoidal category. It is *this* understanding which will be important for our purposes; an equation holds in all monoidal categories if and only if it holds as a consequence of the monoidal axioms, and if and only if it holds in the free monoidal category.

In what follows, we will construct a graphical notation which itself forms a free monoidal category, and hence be left with the resut that an equation holds due to monoidal axioms if and only if it holds in the graphical language.

While we have a *characterisation* of a free monoidal category in Definition 6, it will also be useful to have a *construction* of a free monoidal category given a monoidal signature.

Definition 7 ([JS93], p. 25.)**.** Given a monoidal signature Σ, we construct a category $\mathscr{F}_\Sigma$. The objects of $\mathscr{F}_\Sigma$ are exactly the elements of Σ_0. The morphisms $X \to Y$ are equivalence classes of arrows built inductively from Σ_1 and components of a, l, r and id using $\otimes$, substitution, inversion of isomorphisms and composition. The equivalence relation is given by the monoidal axioms (MC1) and (MC2).

Definition 8 ([JS93], p. 26.)**.** Given a monoidal signature Σ, we construct a category $\mathscr{F}^{\mathrm{s}}_\Sigma$. The objects of $\mathscr{F}^{\mathrm{s}}_\Sigma$ are exactly the elements of the free monoid on Σ_0; words in elements of Σ_0. The morphisms $X \to Y$ are words in the elements of Σ_1 such that the concatenation of the domains (ignoring brackets and Is) is X and the concatenation of the codomains is Y. $\otimes$ is concatenation of words.

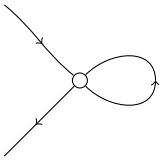
$\mathscr{F}^{\mathrm{s}}_\Sigma$ should be seen as the strict version of $\mathscr{F}_\Sigma$. Routine calculations from the definitions give us:

Proposition 9. *$\mathscr{F}_\Sigma$ is a free monoidal category on the monoidal signature Σ and $\mathscr{F}^{\mathrm{s}}_\Sigma$ is a free strict monoidal category on Σ.*

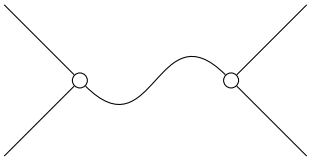
2.2 A graphical notation for monoidal categories: Joyal and Street's string diagrams

2.2.1 Formal characterisation of plane graphs

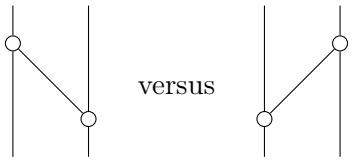
In Chapter 1 we saw some examples of string diagrams for monoidal categories and how they may fit together. In order to build a formal notation scheme, we must first characterise the class of graphs under consideration. What features do we want our graphs to have? We want to disallow things like



as these would have no interpretation in monoidal categories. We want directed edges, else things like



and



might become ambiguous, as it is unclear which node leads to which other node. It is important to be able to distinguish input and output, which will correspond to domain and codomain.

An approximate statement of the theorem we will eventually restate in more detail and prove is:

Theorem 10. *An equation between arrows in a monoidal category follows from the monoidal axioms if and only if the corresponding diagrams are equal up to a suitable planar isotopy.*

Now we come to the sequence of formal definitions which will lead us to a formal characterisation of labelled diagrams in a monoidal category.

Definition 11. A **progressive graph**, $\Gamma = (G, G_0)$, is given by:

- G, a Hausdorff space.
- $G_0 \subseteq G$, a finite subset such that $G \setminus G_0$ is the disjoint union of 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 (finite sequences of vertices beginning and ending with the same vertex with each consecutive pair connected in sequence by an edge).

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. From this we may form a set, called the set of **outer nodes**, which is defined to be

$$\partial\Gamma = \text{boundary}(\hat{\Gamma}) \setminus G_0$$

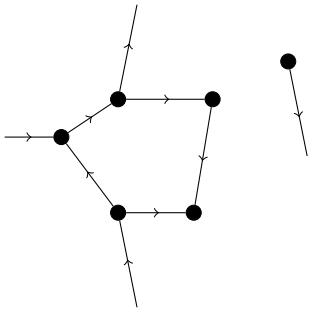


Figure 11: An example of a progressive graph. Note that while eventually we will have all edges pointing downwards, this graph is not embedded in the plane so there is no concept of "downwards".

For an inner node $x \in G_0$, we define the **domain** and **codomain** for x to be

$$\mathrm{dom}(x) = \mathrm{in}(x) \qquad \mathrm{cod}(x) = \mathrm{out}(x)$$

the set of edges directed into and out of x respectively. For the whole graph Γ we also define **domain** and **codomain** to be

$$\mathrm{dom}(\Gamma) = \{\text{edges with outer node as source}\}$$

$$\mathrm{cod}(\Gamma) = \{\text{edges with outer node as target}\}$$

respectively, which we will also sometimes identify with the corresponding sets of outer nodes.

Example 12. Figure 11 shows an example of a progressive graph with inner nodes highlighted. Figure 13(a) shows an example of a progressive graph with no outer nodes.

Definition 13. For a progressive graph $\Gamma = (G, G_0)$, a **progressive embedding of** Γ **in the plane** is given by a continuous injection $\phi : \hat{\Gamma} \hookrightarrow \mathbb{R} \times [a, b]$ for some $a < b \in \mathbb{R}$ such that:

(i) The image of the outer nodes falls within the boundary of the strip — $\phi(\partial\Gamma) \subset \mathbb{R} \times \{a, b\}$ — in such a way that

$$\phi(\mathrm{dom}(\Gamma)) \subset \mathbb{R} \times \{b\} \quad \text{and} \quad \phi(\mathrm{cod}(\Gamma)) \subset \mathbb{R} \times \{a\}$$

and no other part of the graph meets the boundary or exterior of the strip:

$$\phi(\partial\Gamma) = \phi(\Gamma) \cap \mathbb{R} \times \{a, b\}$$

(Here, $\mathrm{dom}(\Gamma)$ and $\mathrm{cod}(\Gamma)$ refer to sets of outer nodes.)

(ii) ϕ respects direction on edges: the "source" of an edge is "higher" than its "target" (with these words given a naïve interpretation).

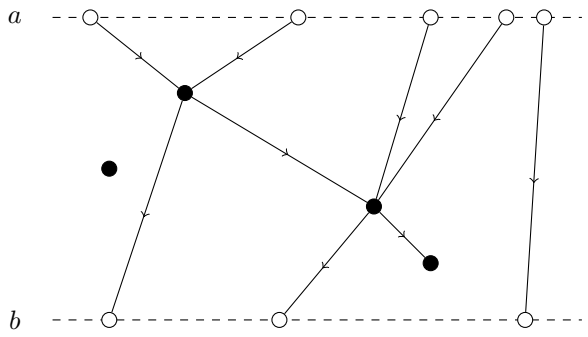


Figure 12: An example of a progressive plane graph.

(iii) The second projection $\pi_2 : \mathbb{R}^2 \to \mathbb{R}$ is injective on each edge.

We apply a linear order to edges in and out of each node, and likewise to $\mathrm{dom}(\Gamma)$ and $\mathrm{cod}(\Gamma)$. This is possible via Remark 15, below.

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

Since a plane graph Γ with its plane embedding ϕ is trivially deformable (via the identity deformation) into the graph-in-the-plane $\phi(\Gamma)$ 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. We will sometimes refer to progressive plane graphs as simply "graphs" for the sake of readability.

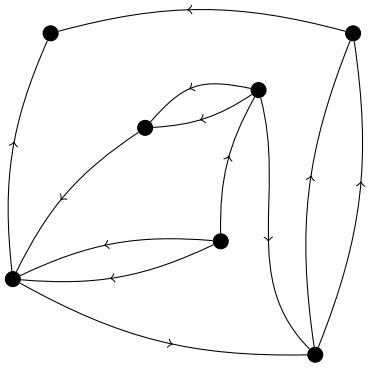
Example 14.

1. An example of a progressive plane graph can be seen in Figure 12, where solid dots ($\bullet$) denote (images of) inner nodes and hollow dots ($\circ$) denote (images of) outer nodes. Condition (i) means that the outer nodes are attached to the edges of the strip. Condition (ii) and (iii) mean that all edges point downwards and we don't end up with any weird edges which are horizontal or double-back.

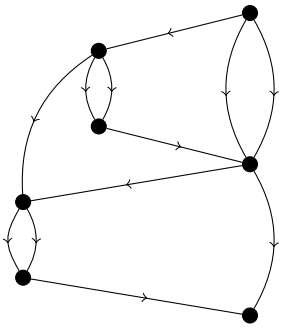
2. Figure 13(b) shows a progressive embedding of the progressive graph (with no outer nodes) shows in Figure 13(a).

3. Figure 14(a) shows a non-example of a progressive plane graph. It is not an example for several violations of axiom (i) of Definition 13. The point labelled with a ① is the image of an inner node which touches the edge of the strip and the point labelled with a ② is the image of an edge which is outside the strip on which the images of the outer nodes lie. Note that the area labelled ② also violates axiom (iii), since this edge "doubles-back" on itself.

4. Figure 14(b) shows a non-example of a progressive plane graph. It is not an example because at the point labelled ③ we see an arrow whose source is lower than its target, violating axiom (ii) of Definition 13.



(a) An example of a progressive graph.



(b) An example progressive embedding of the same graph.

Figure 13: A progressive plane graph with no outer nodes

5. Figure 14(c) shows a non-example of a progressive plane graph. It is not an example because at the point labelled ④ we see a crossing between two edges, in violation of Definition 13 which says that a progressive plane graph is an injective image of a collection of edges and nodes.

Remark 15. We can define a linear order on edges in and out of a node x by picking a suitable $u \in (a, b)$, such that the line $\mathbb{R} \times \{u\} \subset \mathbb{R} \times [a, b]$ doesn't intersect any node. The order on the edges is then the usual order in $\mathbb{R}$. This is illustrated in Figure 15.

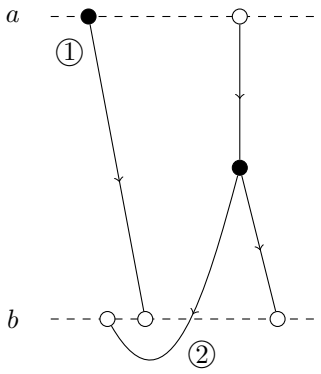
Similarly, we may form a linear order on $\mathrm{dom}(\Gamma)$ and $\mathrm{cod}(\Gamma)$ — since each edge in $\mathrm{dom}(\Gamma)$ and $\mathrm{cod}(\Gamma)$ has a unique associated outer node in $\mathbb{R}\times\{b\}$ and $\mathbb{R}\times\{a\}$ respectively, we may take the usual order on $\mathbb{R}$ to induce two linear orders on these two sets of edges.

It should be noted that there are other conventional ways of defining linear orders of edges around nodes, such as the "counterclockwise" order, but the method we have chosen fits nicely with the standard left-to-right reading of symbolic strings, and so makes translation between diagrams and symbols easier.

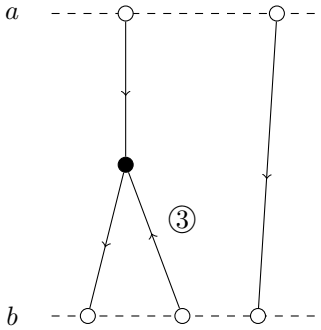
Definition 16. Graphs $\Gamma = (G, G_0)$ and $\Delta = (D, D_0)$ are said to be **isomorphic (as progressive graphs)** when there is a homeomorphism $G \cong D$ which induces a bijection $G_0 \cong D_0$ and preserves orientations of edges. In this case we write $\Gamma \cong \Delta$.

Definition 17. Let $\Gamma = (G, G_0)$ and $\Delta = (D, D_0)$ be isomorphic progressive graphs with embeddings $\phi : \hat{\Gamma} \hookrightarrow \mathbb{R} \times [a, b]$ and $\psi : \hat{\Delta} \hookrightarrow \mathbb{R} \times [c, d]$ as progressive plane graphs respectively. We say that Γ is **deformable into** Δ if there is a continuous function $h : G \times [0, 1] \to \mathbb{R}^2$ such that

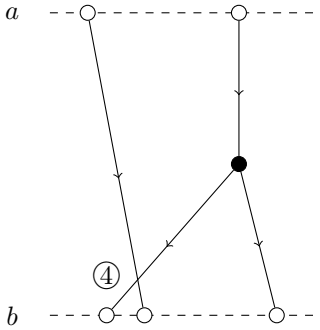
- $h(\hat{\Gamma}, 0) = \phi(\hat{\Gamma}) \subset \mathbb{R} \times [a, b]$ is an embedding of Γ as a progressive plane graph.
- $h(\hat{\Gamma}, 1) = \psi(\hat{\Delta}) \subset \mathbb{R} \times [c, d]$ is an embedding of Γ as a progressive plane graph.
- For each $t \in [0, 1]$, $h(-, t)$ is an embedding $\hat{\Gamma} \hookrightarrow \mathbb{R} \times [a_t, b_t]$ of Γ as a progressive plane graph.



(a) A non-example violating axioms (i) and (iii) of definition 13.

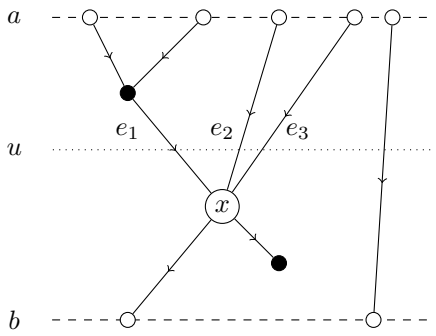


(b) A non-example violating axiom (ii).

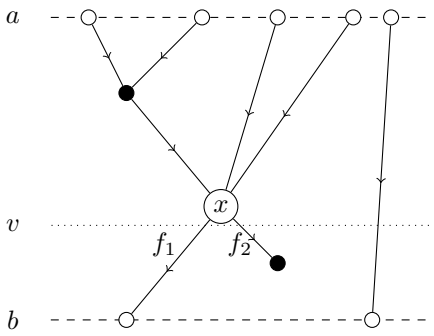


(c) A non-example violating injectivity of the embedding.

Figure 14: Non-examples of progressive plane graphs.



(a) Determining a linear order on $\mathrm{dom}(x)$ by intersection with the horizontal line $\mathbb{R} \times \{u\}$.

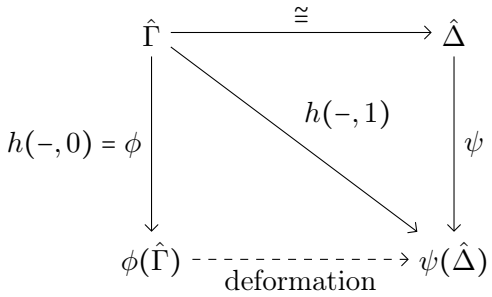


(b) Determining a linear order on $\mathrm{cod}(x)$ by intersection with the horizontal line $\mathbb{R} \times \{v\}$.

Figure 15: Determining linear orders on the domain and codomain of a specified point, as described in Remark 15.

In this case we say that h is the **deformation**, and may also say that the image $\phi(\hat{\Gamma})$ is a **deformation** of $\psi(\hat{\Delta})$, and vice versa.

As a diagram of arrows, the picture to have in mind is:



Note that we may make consistent the designations of inner and outer nodes for each of these graphs. These designations may be straightforwardly carried through deformations, and hence attached to the isomorphism class of progressive graphs and deformation classes of progressive plane graphs.

Example 18. A common example of a deformation is when $\Gamma = \Delta$ and the deformation is between two embeddings of Γ. For example, suppose that Γ is the progressive graph in Figure 16(a),and Γ has two embeddings ϕ and ψ in the plane as a progressive plane graph, as in Figures 16(b) and 16(c).

Then a deformation h of $\phi\Gamma$ into $\psi\Gamma$ can be shown using selected values of t in the embedding $h(-,t)$ of Γ, as in Figure 16(d).

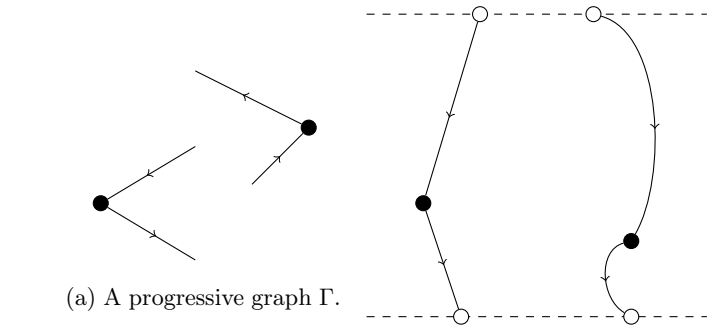
2.2.2 Evaluation of graphs

The progressive plane graphs as we have defined them will soon be labelled using a monoidal signature and used to denote expressions in a monoidal category. To capture the compositional structure of a monoidal category, we will described how progressive plane graphs can be composed and decomposed in ways corresponding to the $\otimes$ and $\circ$ of monoidal categories.

Definition 19. Suppose a progressive graph Γ is isomorphic to a disjoint union of progressive graphs $\Gamma_1, \ldots, \Gamma_n$. Suppose that there are progressive embeddings ϕ of Γ, and ϕ_i of each of the Γ_i such that the images of the Γ_i are disjoint, and the image of Γ coincides with union the images of the Γ_i.

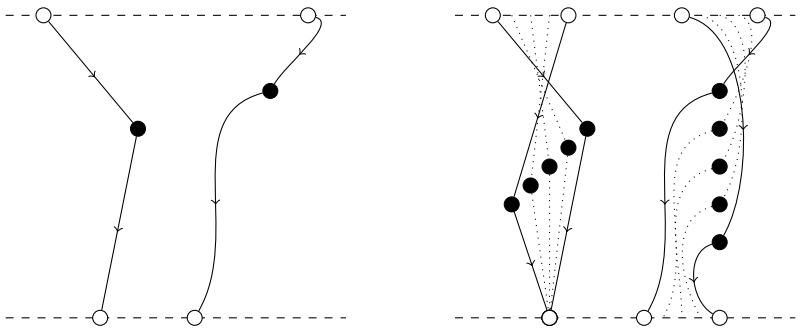
Suppose further that there is a choice of $u_1 < \cdots < u_{n+1} \in \mathbb{R}$ so that the vertical lines $\{u_1, \ldots, u_i\} \times \mathbb{R}$ separate $\mathbb{R}^2$ into n distinct vertical strips in such a way that $\phi_i(\Gamma_i) \subset [u_i, u_{i+1}] \times \mathbb{R}$. Then we say that the progressive plane graph Γ is **tensor-decomposible** into the progressive plane graphs $\Gamma_1, \ldots, \Gamma_n$.

In this case we may write $\Gamma = \Gamma_1 \otimes \cdots \otimes \Gamma_n$, and may use the same terminology to speak of $\Gamma, \Gamma_1, \ldots, \Gamma_n$ as progressive graphs, where to do so would not lead to ambiguity.



(a) A progressive graph Γ.

(b) Embedded image $\phi\Gamma$.



(c) Embedded image $\psi\Gamma$.

(d) A deformation of $\phi\Gamma$ into $\psi\Gamma$.

Figure 16: Two embeddings of Γ in the plane.

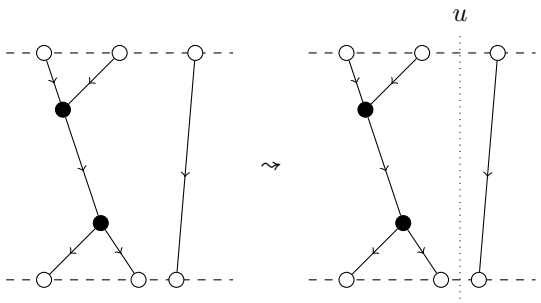


Figure 17: Tensor decomposition of a progressive plane graph

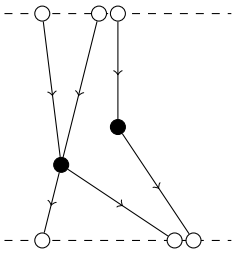


Figure 18: A progressive plane graph with two components but which is not obviously tensor decomposable

Example 20. See Figure 17 for a tensor decomposition of a simple progressive plane graph into two components.

Figure 18 shows a progressive plane graph whose disjoint components do not appear to be separable by a vertical line, but cases such as this are discussed later, in Remark 26.

Definition 21. A graph can also be **sliced** horizontally so that both parts are graphs. A choice of $u \in (b, a)$ will produce a horizontal line $\mathbb{R} \times \{u\}$; so long as this choice of u does not produce a line which intersects an inner node, it is a valid slice. The points of intersection between the horizontal line and the graph are removed and become outer nodes for each of the two new graphs.

Example 22. See Figure 19 for a horizontal slicing of a progressive plane graph.

As previously mentioned, the goal is to label the our progressive plane graphs with objects and morphisms from a monoidal category, so that we may use them to perform calculations.

Definition 23. Let Γ be a progressive plane graph and let $\mathscr{V}$ be a strict monoidal category. A **valuation** of Γ in $\mathscr{V}$ is a pair of functions

$$v_0 : \{\text{edges of } \Gamma\} \to \operatorname{ob} \mathscr{V}$$

$$v_1 : G_0 \to \operatorname{mor} \mathscr{V}$$

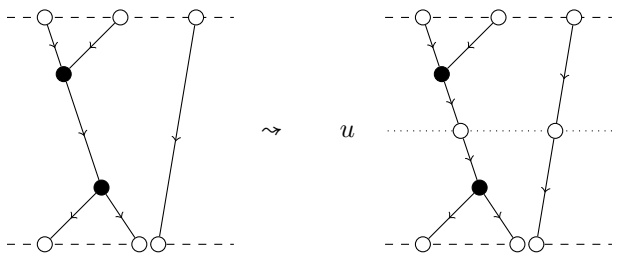


Figure 19: Horizontal slicing of a progressive plane graph

such that for each inner node x with $\mathrm{dom}(x) = \{e_1, \ldots, e_n\}$ and $\mathrm{cod}(x) = \{f_1, \ldots, f_m\}$ we have that

$$v_1(x) : v_0(e_1) \otimes \cdots \otimes v_0(e_n) \to v_0(f_1) \otimes \cdots \otimes v_0(f_m) \tag{2.1}$$

in $\mathscr{V}$.

We call a valuation of a progressive plane graph in $\mathscr{V}$ a **progressive plane diagram in $\mathscr{V}$** or just a **diagram in $\mathscr{V}$**, or even merely a **diagram**, where the context makes $\mathscr{V}$ clear or irrelevant.

A valuation of Γ may be inherited by subgraphs of Γ, and thus we may also refer to **sub-diagrams** of a diagram.

Remark 24. Notice that a valuation can be made consistent across deformations of a graph, since the components of each stage of the deformation are in bijection with each other. From now on we will assume that if a graph has a valuation applied to it, that this valuation can equally be seen to be applied to the deformation class of this graph. Equally, if a graph is equipped with a valuation, we can see a deformation of that graph as a deformation of a diagram.

Given a valuation on the components of a graph, we also want to be able to assign an overall value to the graph as a whole. To do this, we break the diagram up into components and assemble the value based on this decomposition.

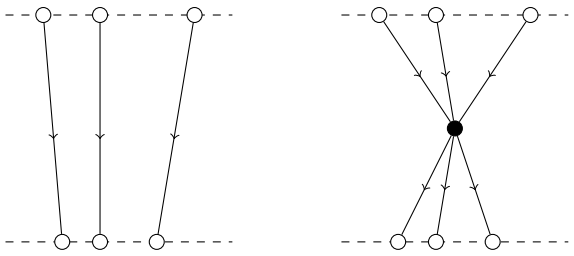
Definition 25. We will break the diagram up into rectangular pieces such that each piece contains some sub-diagram of one of the following types:

(A) If Γ has zero inner nodes then we will call it a diagram of **A-type**. For example, Figure 20(a).

(B) If Γ has a single inner node, x, and $\mathrm{dom}(\Gamma) = \mathrm{dom}(x)$ and $\mathrm{cod}(\Gamma) = \mathrm{cod}(x)$, we'll call it a diagram of **B-type**. For example, Figure 20(b).

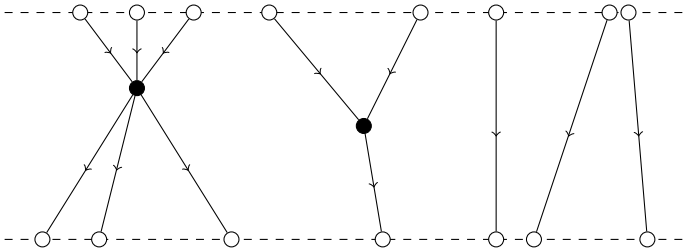
(C) If Γ has a $\otimes$-decomposition into subdiagrams of A- and B-type, we'll call it a diagram of **C-type**. For example, Figure 20(c).

(D) If Γ is horizontally slicable into C-type subdiagrams then we'll call it a diagram of **D-type**. For example, Figure 21.



(a) An A-type diagram.

(b) A B-type diagram.



(c) A C-type diagram.

Figure 20: Examples of A-, B- and C-type diagrams (labels omitted).

We call a particular choice of horizontal slices for the D-type diagrams, together with further choices of tensor-decompositions for C-type subdiagrams a **(full) decomposition** for the diagram.

Remark 26. We would like this to be a complete classification of diagrams, but there are a number of potential counter-examples we consider. First, consider Figure 22(a). This diagram is the disjoint union of subdiagrams, but is not tensor-decomposable as no necessary vertical separating lines exist. However, by further horizontal slicing we may present this as a D-type, rather than a C-type diagram, as can be seen in Figure 22(b).

Another, scarier candidate counterexample would be the diagram in Figure 23. In this diagram, the imprecisely rendered oscillating lines should be seen to represent rotated *topologist's sine curves* [SS78], or two intermeshing, slightly translated copies of the graphs of $x = \sin(1/y)$. One might think that no amount of horizontal slicing will present both of these as D-type diagrams. However, we may prove that this is not a problem in the following proposition.

Proposition 27. *Every diagram is of D-type.*

Proof. Let Γ be a diagram in $\mathbb{R} \times [a, b]$. For every $s \in [a, b]$ there's an $\varepsilon_s > 0$ such that the slice of Γ within the closure of the open ε_s-neighbourhood of $\mathbb{R} \times \{s\}$ is a C-type subdiagram. We know this since each valid horizontal slice intersects the graph at finitely-many points, and the embedding is continuous. The union of these ε_s-neighbourhoods form an open cover of the diagram. Since the diagram in the plane is a continuous image of a compact space (since the embedding of Γ in the plane is an embedding of the

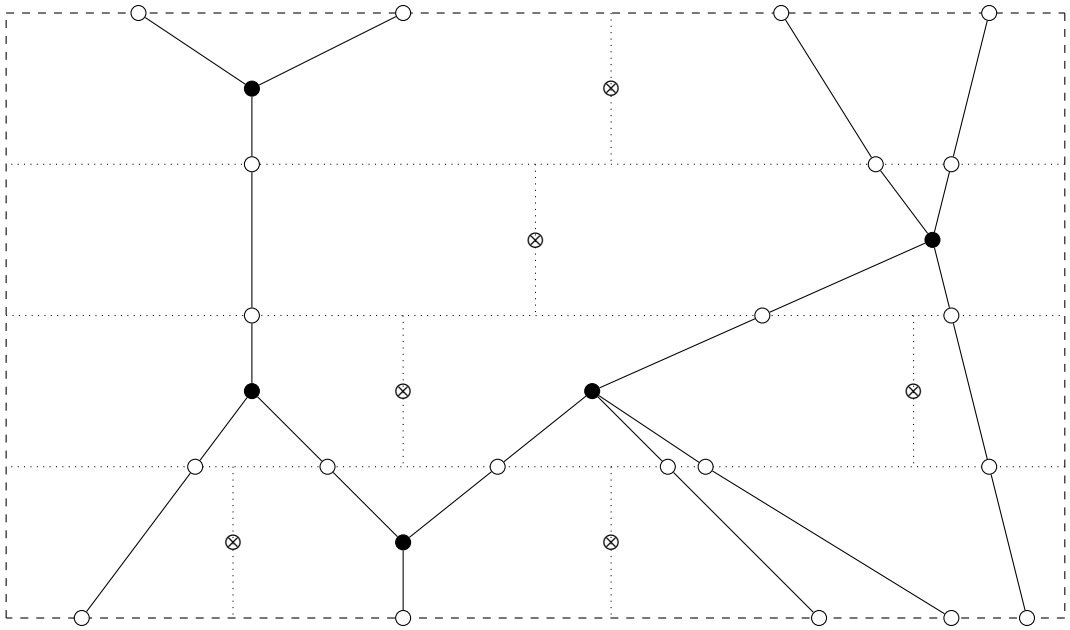
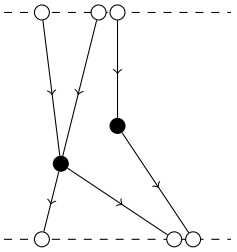
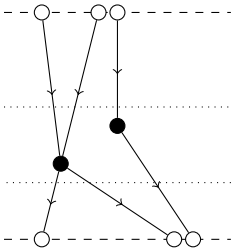


Figure 21: An example of a full decomposition of a D-type digram (direction arrowheads and labels omitted).



(a) A diagram for which a D-type presentation may appear impossible.



(b) A horizontal slicing that leaves each component a C-type diagram.

Figure 22: Not a real counterexample.

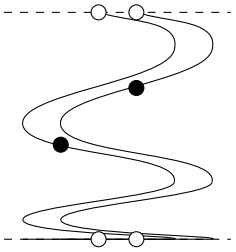


Figure 23: A candidate counterexample to the claim that all diagrams are of D-type

compactification $\hat{\Gamma}$ of G described in Definition 11), it is a compact subset of the plane. Hence, the open cover formed from the ε_s-neighbourhoods has a finite subcover, which gives rise to a finite decomposition of Γ into C-type diagrams. □

Now, returning to our second candidate counter-example, we see that a true topologist's sine curve is not a compact subset of the plane and so cannot be a valid progressive plane graph.

We now know that any progressive plane diagram is of D-type, and so from some choice of decomposition and a valuation on the components, we can construct the diagram's *value*.

Definition 28. Let Γ be a diagram in $\mathscr{V}$ by virtue of a valuation comprised of v_0 and v_1. Assume that we have a chosen full decomposition of Γ. A **value** for Γ in $\mathscr{V}$ is an assignation $v(\Gamma) \in \operatorname{mor} \mathscr{V}$, calculated via the structure of the decomposition in the following way:

(D) Since Γ is a diagram of D-type, its decomposition contains many subdiagrams $\Gamma_1, \ldots, \Gamma_n$, of C-type, vertically concatenated (top-to-bottom). We have that $v(\Gamma) = v(\Gamma_n) \circ \cdots \circ v(\Gamma_1)$, where the values of the C-type Γ_i are calculated via the following.

(C) A C-type subdiagram Δ is equipped with a tensor-decomposition into subdiagrams $\Delta_1, \ldots, \Delta_m$, each of A- or B-type, with order taken left-to-right. We have that $v(\Delta) = v(\Delta_1) \otimes \cdots \otimes v(\Delta_m)$, where the values of the A- and B-type Δ_i are calculated via the following.

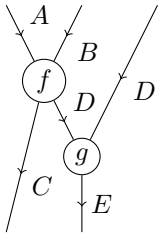
(B) If a subdiagram Δ_j is of B-type then it contains exactly one inner node, x. The value of Δ_j is then $v(\Delta_j) = v_1(x)$.

(A) If a subdiagram Δ_j is of A-type then it contains no inner nodes and so is a disjoint union of edges $e_1, \ldots, e_l$, each attached to two outer-nodes. Then the value of Δ_j is then $v(\Delta_j) = \mathrm{id}_{v_0(e_1)} \otimes \cdots \otimes \mathrm{id}_{v_0(e_l)}$.

We denote a valuation on a graph by labelling its edges with objects and nodes with morphisms, for example, for

$$f : A \otimes B \to C \otimes D \qquad g : D \otimes D \to E$$

we may write



to denote the composite $g \circ f : A \otimes B \otimes D \to C \otimes E$.

The value of a graph is usually not explicitly denoted, but may be implicitly inferred from a valuation in an inductive manner based on a chosen decomposition of the graph. In fact, the choice of decomposition does not affect the calculated value, as will be demonstrated in Proposition 31.

2.2.3 Robustness of values

We'll first show that it is invariant under different choices of decomposition and then that it is invariant under deformation of diagrams.

Lemma 29. *Any two choices of tensor decomposition of a C-type diagram yield the same calculated value.*

Proof. Suppose that Γ is a C-type diagram. Then Γ is tensor-decomposable into A-type and B-type subdiagrams $\Gamma_1 \otimes \cdots \otimes \Gamma_n$. A B-type subdiagram is connected and therefore not further tensor-decomposable. An A-type subdiagram is the disjoint union of connected, valued edges, and so if Γ_i, say, is an A-type subdiagram which is tensor-decomposable into $\Delta_1 \otimes \Delta_2$, say, we'd have (using informal notation)

$$\{\text{edges of } \Gamma_i\} = \{\text{edges of } \Delta_1\} + \{\text{edges of } \Delta_2\}$$

with a separating vertical line between Δ_1 and Δ_2. Since Γ_i is now presented as C-type as well as A-type, we have that

$$v(\Gamma_i) = v(\Delta_1) \otimes v(\Delta_2)$$

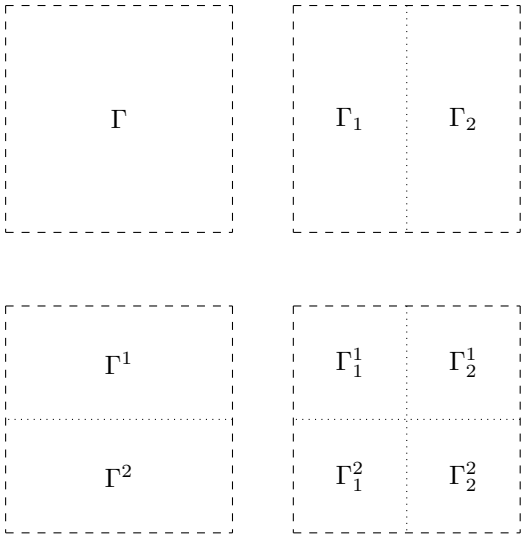
Hence, any two tensor-decompositions of a C-type diagram yield the same value. □

Lemma 30. *If a diagram's value is calculated from a decomposition as a D-type diagram then a finer decomposition yields the same value.*

Proof. To show this it is sufficient to show that slicing a C-type diagram does not alter the value, since a D-type diagram is by definition sliced into finitely many C-type diagrams.

Let us suppose that Γ is a diagram of C-type with value $v(\Gamma)$, and suppose that we horizontally slice it into two subdiagrams Γ^1 (upper) and Γ^2 (lower). Sice Γ is C-type, it has a tensor-decomposition into $\Gamma_1 \otimes \Gamma_2$ with Γ_1, Γ_2 of A- or B-type. We may assume without loss of generality that we have chosen a non-trivial such decomposition. Now let Γ_1^1 and Γ_1^2 be the same horizontal slice applied to Γ_1 and let Γ_2^1 and Γ_2^2 be the horizontal

slice applied to Γ_2, so that we we are left in the following state:



Since by slicing Γ horizontally into two C-type subgraphs we have presented it as a D-type diagram, we have a new value calculated based on the D-type compositional structure of Γ: $\hat{v}(\Gamma) = v(\Gamma^1) \circ v(\Gamma^2)$. However,

$$\begin{aligned}
\hat{v}(\Gamma) &= v(\Gamma^2) \circ v(\Gamma^1) \\
&= \left[v(\Gamma_1^2) \otimes v(\Gamma_2^2)\right] \circ \left[v(\Gamma_1^1) \otimes v(\Gamma_2^1)\right] \\
&= \left[v(\Gamma_1^2) \circ v(\Gamma_1^1)\right] \otimes \left[v(\Gamma_2^2) \circ v(\Gamma_2^1)\right] \\
&= v(\Gamma_1) \otimes v(\Gamma_2) \\
&= v(\Gamma)
\end{aligned}$$

□

Proposition 31. *Any two full decompositions of a diagram yield the same value.*

Proof. This follows from Lemma 29, together with the observation that the invarience of value under different choices of horizontal slicings follows from Lemma 30. □

For valuations and values to be useful, we need to show that they're deformation invariant. This was the whole idea!

Theorem 32. *Let $h : \hat{G} \times [0,1] \to \mathbb{R}^2$ be a deformation of a graph $\Gamma = (G, G_0)$ into a graph Δ. Then $v(h(G,0)) = v(h(G,1))$.*

For the theorem we will need the following lemma. In view of the lack of distinction between valuations of graphs and their deformations (etc.), this theorem may be seen as abusing notation, but its meaning should be unambiguous.

Lemma 33. *If a (sub)diagram of A- or B-type is deformed, its value doesn't change.*

Proof. This follows right from the definition of the value of an A- or B-type diagram, since we have already seen that a valuation on a diagram is unaffected by deformation. □

Now we are ready to prove the theorem.

Proof of Theorem 32. Given our deformation h, we pick a $t_0 \in [0,1]$. Now we'll consider the deformation at time t_0; that is to say, $h(\hat{G}, t_0)$. (We say "time" to aid intuition, but it has no further meaning or implication.) Since there are finitely many nodes in Γ there are finitely-many edges and, by Proposition 27, there's a finite open cover of these edges by horizontal strips which provide us a decomposition of Γ as a D-type diagram so that these strips are C-type subdiagrams. Let $\varepsilon > 0$ be the minimum distance of any inner node to any boundary, where "boundary" refers to the divisions between C-type slices and the tensor-divisions between A- and B-type subdiagrams of the C-type slices.

The deformation h is continuous, so there's a $\delta_0 > 0$ such that for every t within δ_0 of t_0, each node moves less than ε and so no node enters or leaves any subdiagram of our full decomposition and so, by Proposition 31, the value of our diagram doesn't change.

This collection of $(t_0 - \delta_0, t_0 + \delta_0)$ intervals provide an open cover of $[0,1]$. Since $[0,1]$ is compact so it must have a finite sub-cover of intervals, and a full decomposition of Γ for each interval so that $v(h(\Gamma, t))$ is constant for t within each interval.

Since h is continuous and v is locally constant, v is constant for the full extent of $[0,1]$. □

In light of this, it makes sense for us to consider graphs, together with their valuations and values, only up to deformation. From now on we will not tend to distinguish between the value of a graph and the value of its deformation class, and Theorem 32 justifies this.

Now that we can label a diagram in a monoidal category, the final step is to examine how the diagrams themselves behave. We will form the *free* category of labelled diagrams and show that this satisfies a reasonable definition of *free*. This will conclude our examination of diagrams for monoidal categories, as then we will know that by constructing and deforming diagrams, we may calculate in a monoidal category accurately up-to unique isomorphism.

2.2.4 A category of diagrams

As noted in [JS91], the definition of a valuation of a progressive plane graph in a monoidal category doesn't use composition of morphisms of the monoidal category — composition comes in when calculating the value of a diagram. Therefore, there is an easy modification of this definition to allow *valuations of progressive plane graphs in monoidal signatures*.

Definition 34. Let Γ be a progressive plane graph and let Σ be a monoidal signature. A **valuation** of Γ in Σ is a pair of functions

$$v_0 : \{\text{edges of } \Gamma\} \to \Sigma_0 \qquad v_1 : G_0 \to \Sigma_1$$

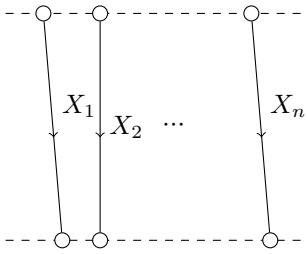


Figure 24: A diagram for $\mathrm{id}_{X_1 \otimes X_2 \otimes \cdots \otimes X_n}$.

such that for each inner node $x \in G_0$ with

$$\mathrm{dom}(x) = \{e_1, \ldots, e_n\} \qquad \mathrm{cod}(x) = \{f_1, \ldots, f_m\}$$

there is a $g \in \Sigma_1$ such that

$$\mathrm{dom}(g) = X \in \hat{\Sigma}_0 \qquad \mathrm{cod}(g) = Y \in \hat{\Sigma}_0$$

with X a tensor-word containing exactly $v_0(e_1), \ldots v_0(e_n)$ (in that order) and Y a tensor word containing exactly $v_0(f_1), \ldots, v_0(f_m)$ (in that order).

We call a PPG together with a valuation in a monoidal signature Σ a **progressive plane diagram in** Σ or just a **diagram in** Σ. This is of course closely related to Definition 28 of a progressive plane diagram in $\mathscr{V}$.

As before, we may also use this terminology to refer to the deformation-class of a diagram with valuation.

Now we will describe the *free monoidal category of diagrams in a monoidal signature*, $\mathscr{F}_\Sigma^{\mathrm{dia}}$, which we will then demonstrate to be the free strict monoidal category on a monoidal signature, $\mathscr{F}_\Sigma^{\mathrm{s}}$.

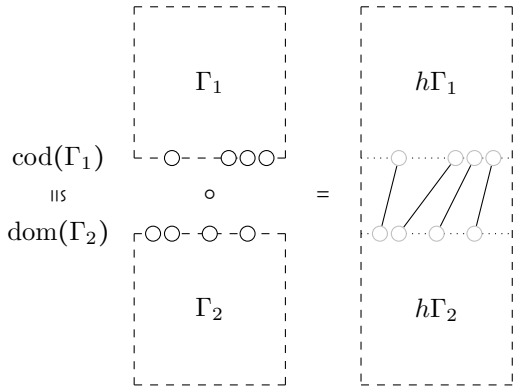
Definition 35. Given a monoidal signature Σ, we will describe a category which we call $\mathscr{F}_\Sigma^{\mathrm{dia}}$. $\mathscr{F}_\Sigma^{\mathrm{dia}}$ has as objects unbracketed $\otimes$-words of elements in Σ_0. An arrow $X_1 \otimes \ldots \otimes X_n \to Y_1 \otimes \ldots \otimes Y_m$ is a deformation-class of diagrams in Σ whose domain is $\{e_1, \ldots, e_n\}$ and whose codomain $\{f_1, \ldots, f_m\}$ (linear orders implied by indicies), with valuation such that

$$v_0(e_1) = X_1, \quad \ldots, \quad v_0(e_n) = X_n$$
$$v_0(f_1) = Y_1, \quad \ldots, \quad v_0(f_m) = Y_m$$

The identity morphism on an object is the A-type diagram which consists of a set of vertical line segments affixed to two parallel horizontal lines in the plane, for example, the diagram in Figure 24 shows $\mathrm{id}_{X_1 \otimes X_2 \otimes \cdots \otimes X_n}$.

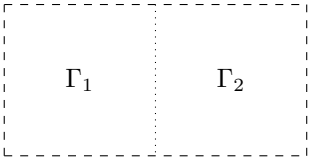
Composition of arrows is vertical composition of graphs whose domain and codomain agree (in the usual sense for composition of arrows). Given two such graphs, it may be that the outer nodes in their domain and codomain don't exactly match up in space.

They can, however, be joined up in the following manner. Suppose Γ_1 is a diagram in $\mathbb{R} \times [a, b]$ and Γ_2 is a diagram in $\mathbb{R} \times [c, d]$ such that $\mathrm{cod}(\Gamma_1) \cong \mathrm{dom}(\Gamma_2)$. Then we may deform Γ_1 in the manner of a rigid translation isometry of the plane so that it is a valued graph in $\mathbb{R} \times [a + d + 1, b + d + 1]$. A set of line segments, each homeomorphic to $[0, 1]$, may be glued to the outer nodes in $\mathrm{cod}(\Gamma_1)$ and $\mathrm{dom}(\Gamma_2)$ in a way which respects their linear orders. These outer nodes are now discarded and the union of Γ_1, Γ_2, and the new line segments forms a new progressive plane graph in $\mathbb{R} \times [c, b + d + 1]$ whose domain is $\mathrm{dom}(\Gamma_1)$, whose codomain is $\mathrm{cod}(\Gamma_2)$, and whose inner nodes are exactly the union of the inner nodes of Γ_1 and Γ_2.



We equip this progressive plane graph with the unique valuation which restricts to the original valuations on Γ_1 and Γ_2 after a suitable horizontal slice. We call this diagram $\Gamma_2 \circ \Gamma_1$.

$\mathscr{F}_\Sigma^{\mathrm{dia}}$ also has a monoidal structure. Tensor product of objects is $\otimes$-product of words. Tensor product of morphisms is horizontal juxtaposition of graphs (after suitable deformation so that they are in the same strip of $\mathbb{R}^2$).



We will denote this tensor product of diagrams also by "$\otimes$" when to do so will not be ambiguous.

Proposition 36 ([JS91], p. 71.)**.** *$\mathscr{F}_\Sigma^{\mathrm{dia}}$ is a strict monoidal category.*

Proof. The following points comprise a proof that this notion of $\mathscr{F}_\Sigma^{\mathrm{dia}}$ is well-defined and forms a monoidal category. In this proof we will use the notation "$\Gamma \sim \Delta$" to indicate that the diagram Γ is deformable into the diagram Δ.

- Domain and codomain are well defined; if $\Gamma \sim \Delta$ then $\mathrm{dom}(\Gamma) = \mathrm{dom}(\Delta)$ and $\mathrm{cod}(\Gamma) = \mathrm{cod}(\Delta)$.

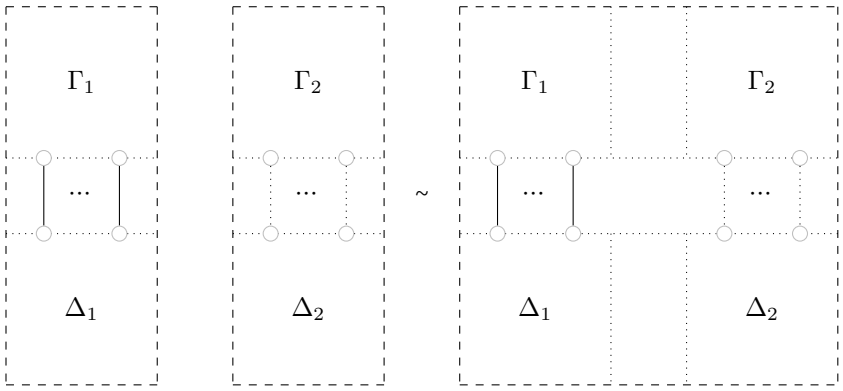


Figure 25: A deformation demonstrating the functoriality of $\otimes$.

- Tensor product is well defined; if $\Gamma_1 \sim \Delta_1$ and $\Gamma_2 \sim \Delta_2$ then $\Gamma_1 \otimes \Gamma_2 \sim \Delta_1 \otimes \Delta_2$.
- Composition is well defined; if $\Gamma_1 \sim \Delta_1$ and $\Gamma_2 \sim \Delta_2$ then $\Gamma_2 \circ \Gamma_1 \sim \Delta_2 \circ \Delta_1$, so long as each of these composites is defined.
- Tensor and composition are associative by construction and associativity is strict (up to deformation) in both cases. Note that the left and right units are also strict since I diagrammatically invisible.
- Tensor is functorial. If we have diagrams $\Gamma_1, \Gamma_2, \Delta_1, \Delta_2$ such that the composites $\Delta_1 \circ \Gamma_1$ and $\Delta_2 \circ \Gamma_2$ then the fact that

  $$(\Delta_1 \circ \Gamma_1) \otimes (\Delta_2 \circ \Gamma_2) \sim (\Delta_1 \otimes \Delta_2) \circ (\Gamma_1 \otimes \Gamma_2)$$

  is manifest in the deformation shown in Figure 25.

□

Now we come to the main theorem.

Theorem 37 ([JS91], Theorem 1.2.)**.** *$\mathscr{F}_\Sigma^{\mathrm{dia}}$ is the free (strict) monoidal category on the monoidal signature Σ.*

Proof. Recall from Definition 6 that in order to show that $\mathscr{F}_\Sigma^{\mathrm{dia}}$ is free on the monoidal signature Σ we must find an interpretation $\iota \in [\Sigma, \mathscr{F}_\Sigma^{\mathrm{dia}}]_{In}$ such that

$$- \circ \iota : [\mathscr{F}_\Sigma^{\mathrm{dia}}, \mathscr{V}]_{St} \to [\Sigma, \mathscr{V}]_{In}$$

is an equivalence of categories for each monoidal category $\mathscr{V}$.

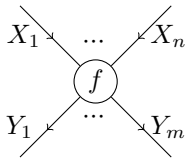
We construct ι in the following way. $\iota \in [\Sigma, \mathscr{F}_\Sigma^{\mathrm{dia}}]_{In}$ is an interpretation $\Sigma \to \mathscr{F}_\Sigma^{\mathrm{dia}}$.

Objects of $\mathscr{F}_\Sigma^{\mathrm{dia}}$ are words in elements of Σ_0, so to each element X of Σ_0 is sent by ι to the singleton word X; the inclusion of the generators in the free monoid $\big(\mathrm{ob}(\mathscr{F}_\Sigma^{\mathrm{dia}}), \otimes\big)$. Observe that this uniquely extends to

$$\hat{\iota_0} : \hat{\Sigma}_0 \to \mathrm{ob}(\mathscr{F}_\Sigma^{\mathrm{dia}})$$

which forgets the brackets of binary words in $\hat{\iota_0}$.

Morphisms of $\mathscr{F}_\Sigma^{\mathrm{dia}}$ are diagrams with valuations in Σ, so each element f of Σ_1 — whose domain is a tensor product containing exactly the variables $X_1, \ldots, X_n \in \Sigma_0$ (in order) and whose codomain is a tensor product containing exactly the variables $Y_1, \ldots, Y_m \in \Sigma_0$ — ι_1 sends f to the (deformation class of the) diagram



with valuation in Σ as indicated by labels.

Let $\mathscr{V}$ be any monoidal category, which we may assume, without loss of generality, to be strict (by Proposition 240). The functor $- \circ \iota$ assigns to a strong monoidal functor $F : \mathscr{F}_\Sigma^{\mathrm{dia}} \to \mathscr{V}$ an interpretation $\Sigma \to \mathscr{V}$ which is F applied to the "natural" interpretation ι of Σ in $\mathscr{F}_\Sigma^{\mathrm{dia}}$. Similarly, given a monoidal natural transformation

$$\theta : F \implies G : \mathscr{F}_\Sigma^{\mathrm{dia}} \to \mathscr{V}$$

then $- \circ \iota$ will give us $\theta \circ \iota : (F \circ \iota) \to (F \circ \iota)$, a morphism of interpretations defined as

$$\begin{array}{ccc}
(F \circ \iota)(X) & \xrightarrow{\quad (\theta \circ \iota)_X \quad} & (G \circ \iota)(X) \\
\| & & \| \\
F(\iota X) & & G(\iota X) \\
\| & & \| \\
F(X_1 \otimes \cdots \otimes X_n) \in \mathscr{V} & \xrightarrow[\theta_{X_1 \otimes \cdots \otimes X_n}]{} & G(X_1 \otimes \cdots \otimes X_n) \in \mathscr{V}
\end{array}$$

To show $- \circ \iota$ is an equivalence of categories for arbitrary $\mathscr{V}$, we'll show that it's bijective on hom sets and surjective on objects.

Pick an object $\kappa \in [\Sigma, \mathscr{V}]_{In}$, an interpretation $\kappa : \Sigma \to \mathscr{V}$; we'll find a strong monoidal functor $T : \mathscr{F}_\Sigma^{\mathrm{dia}} \to \mathscr{V}$ such that $T \circ \iota = \kappa$.

Since $\mathrm{ob}(\mathscr{F}_\Sigma^{\mathrm{dia}})$ is the free monoid on Σ_0, the action of T on objects must be uniquely determined if it's to be monoidal. An arrow of $\mathscr{F}_\Sigma^{\mathrm{dia}}$ is a (deformation class with representative) diagram Γ, together with valuation (v_0, v_1) in Σ. By applying κ to this valuation we achieve a valuation $(\kappa v_0, \kappa v_1)$ of Γ in $\mathscr{V}$. Since this valuation is in a monoidal category rather than a monoidal signature, it yields a value in $\mathscr{V}$, which we'll

call $(\kappa v)(\Gamma)$. Set

$$T(\Gamma, v_0, v_1) = (\kappa v)(\Gamma)$$

(where we take (Γ, v_0, v_1) to a diagram together with valuation in Σ — a morphism of $\mathscr{F}_\Sigma^{\mathrm{dia}}$ whose source and target are implict in v_0). This is unique and well-defined by Theorem 32, and preserves composition and tensor by the definition of value.

Now suppose that $F, G : \mathscr{F}_\Sigma^{\mathrm{dia}} \to \mathscr{V}$ are strong monoidal functors and

$$F \circ \iota, G \circ \iota : \Sigma \to \mathscr{V}$$

are interepretations, and suppose we have morphism of interpretations

$$\upsilon : F \circ \iota \to G \circ \iota : \Sigma \to \mathscr{V}$$

Then we want a monoidal natural transformation

$$\theta : F \implies G : \mathscr{F}_\Sigma^{\mathrm{dia}} \to \mathscr{V}$$

such that $\theta \circ \iota = \upsilon$.

We must have $\theta_{X_1 \cdots X_n}$ defined as

$$\begin{array}{ccc} F(X_1 \cdots X_n) & \xrightarrow{\theta_{X_1 \cdots X_n}} & G(X_1 \cdots X_n) \\ \phi_2^{-1} \downarrow & & \uparrow \phi_2 \\ \vdots & & \vdots \\ \phi_2^{-1} \downarrow & & \uparrow \phi_2 \\ FX_1 \otimes \cdots \otimes FX_n & \xrightarrow[\upsilon_{X_1} \otimes \cdots \otimes \upsilon_{X_n}]{} & GX_1 \otimes \cdots \otimes GX_n \end{array}$$

so that for $X \in \Sigma_0$ we have

$$\upsilon_X = (\theta \circ \iota)_X : (F \circ \iota)(X) = FX \to GX = (G \circ \iota)(X)$$

Monoidal conditions (MN1) and (MN2) are satisfied by construction, and naturality square

$$\begin{array}{ccc} FX & \xrightarrow{\theta_X} & GX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FY & \xrightarrow[\theta_Y]{} & GY \end{array}$$

commutes if and only if it commutes with f a diagram with at most one inner node, which is true by the definition of a morphism of interpretations. □

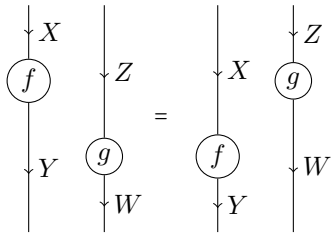
Corollary 38. *An equation between morphisms in a monoidal category follows from the monoidal axioms if and only if the graphs for the morphisms are equal up to deformation.*

Now we should be confident with proving in monoidal categories using this graphical notation.

Example 39. For a simple example, the "interchange law" for monoidal categories is very easy to prove. For $f : X \to Y$ and $g : Z \to W$,

$$(f \otimes Z) \circ (Y \otimes g) = (X \otimes g) \circ (f \otimes W)$$

becomes



which follows from a simple deformation.

Corollary 38 is what provides the real utility of string diagrams for monoidal categories. Where before a particular calculation or proof may have involved repeated applications of substitutional instances of monoidal axioms, now a researcher may quickly see a deformation providing an equality.

Chapter 3

A graphical foundation for interleaving structures in games

Having looked at progressive plane graphs in some detail in the previous chapter, we now turn our attention to game semantics. In what follows, we will develop a graphical representation for various interleaving structures. We again use the framework of progressive plane graphs for our diagrams, but the interpretation of the graphs will be entirely different to that in Chapter 2.

3.1 Graphical foundations for schedules

Harmer, Hyland and Melliès introduced the concept of *schedules* in [HHM07] give an explicit interleaving structure on games à la Lamarche.

3.1.1 Combinatorial $\multimap$-schedules

We recall the combinatorial definition of schedules and of composition of schedules from [HHM07].

Definition 40 (as in Harmer et al. [HHM07])**.** 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 $|e| = n = |e|_0 + |e|_1$.

$\multimap$-scheduling functions are also called *schedules* in [HHM07], but we will take care not to confuse this with Definition 46 of *schedule*, to follow. When necessary for disambiguation, we will call the latter a "graphical schedule". In Theorem 62 we will show that the definitions are equivalent.

Example 41. The following are scheduling functions:

(i) `1001111001` and `1001100001` are examples illustrated in Figure 7(a).

(ii) Any nonempty prefix (or *restriction* [HHM07]) of a schedule is a schedule.

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

Definition 42 (as in Harmer et al. [HHM07], p. 4.)**.** We will use the notation $e : p \to q$ when e is a schedule $e : \{1, \ldots, p+q\} \to \{0, 1\}$ with $|e|_0 = p$ and $|e|_1 = q$.

Let $e : p \to q$ be a such a schedule. Writing $[n]$ to denote the set $\{1, \ldots, n\}$, 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 (i) of Example 41 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 7(b) and 7(c).

Definition 43 ([HHM07])**.** 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" possible schedule of its length. Any nonempty prefix of a copycat function is also a copycat function.

Theorem 44 ([HHM07])**.** *Positive natural numbers and schedules $e : p \to q$ form a category, Υ, with composition as in Definition 42, and with copycat scheduling functions as identities.*

A proof of Theorem 44 does not appear explicitly in [HHM07], though for associativity of composition, reference is made to the merges of sketches from [HS02]. The theorem is certainly true, but a proof of associativity seems combinatorially cumbersome.

3.1.2 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 string diagrams, which is laid out in Chapter 2. We have chosen this framework as it resembles the pictures of schedules which exist in the literature. Further discussion of this can be found in Section 1.4.

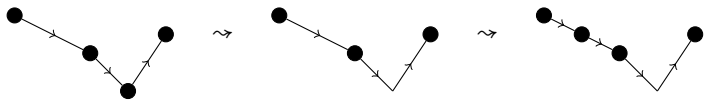


Figure 26: A node is removed; a node is added; the path remains a path.

Recall Definition 13 of *progressive plane graph*. When a graph Γ has no outer nodes, condition (i) (that outer nodes fall within the boundary of a specified strip) is vacuously satisfied, so a progressive embedding is simply one which respects the direction of edges (condition (ii)) and sends edges to uniformly downward-pointing curves in the plane (condition (iii)). See Figure 13(b) for such an example.

Example 45. Consider the left-hand graph of Figure 7(a). 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 7(b), 7(c) and 50(b) are progressive plane graphs.

In this chapter, our primary interest is in progressive plane graphs that are given by directed paths, since schedule diagrams and interleaving diagrams are paths (see for example Figures 7(a) and 7(c)). 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 make use of in its definition is that paths remain paths when we remove nodes (and glue adjoining edges) or add nodes (and split adjoining edges). See Figure 26 for an illustration of this.

Definition 46. 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:

$$p_1 = v_1 \tag{Sc1}$$

$$\text{for each } k, \text{ either } \{p_{2k}, p_{2k+1}\} \subset U \text{ or } \{p_{2k}, p_{2k+1}\} \subset V \tag{Sc2}$$

- 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 will 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, and we will tend to do this for the sake of the clarity.

We will also frequently refer to a schedule's "left-hand" or "right-hand" nodes.

For examples, Figure 7(a) shows a schedule $4 \to 6$ on the left and a schedule $6 \to 4$ on the right, and Figure 7(c) shows a schedule $4 \to 4$.

It is our intention that graphical schedules, as defined, be a characterisation of of the plane graphs drawn by researchers. We therefore need a notion of "sameness" which is less strict that the identity on subsets of $\mathbb{R}^2$. This will permit different drawings of (intuitively) "the same" schedule to refer to the same mathematical object, as well as making certain operations on schedules easier to define and understand. There is also a notion of deformation of progressive plane graph (Definition 17), but we make a refinement here to *deformation of schedule*. In what follows, we will make similar refinements to the notion of deformation for each of the classes of progressive plane graphs we consider.

Definition 47. 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.

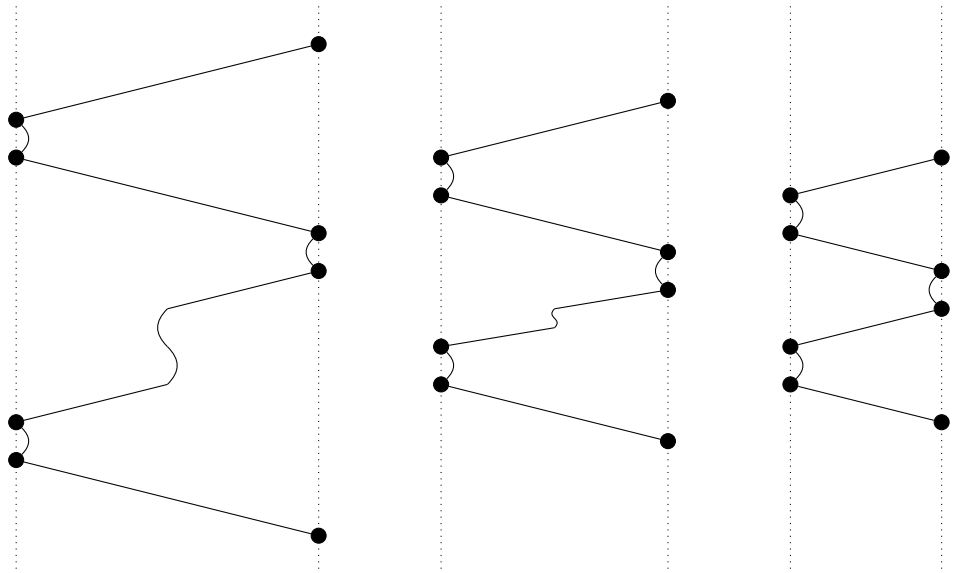
We say that S **is deformable into S' (as a $\multimap$-schedule)** if there is a deformation $h : \hat{\Sigma} \times [0, 1] \hookrightarrow \mathbb{R}^2$ of Σ and ι into Σ' and ι' such that for each $t \in [0, 1]$, $h(-, t)$ is an embedding $\hat{\Sigma} \hookrightarrow [u_t, v_t] \times \mathbb{R}$ of Σ as a schedule in the plane such that $h(U, t) \subset L_{u_t}$ and $h(V, t) \subset L_{v_t}$.

Observe that in this case we are guaranteed that $U \cong U'$, $V \cong V'$ and $P \cong P'$ are order-preserving bijections (where P and P' are the path orders of $U + V$ and $U' + V'$ respectively).

For example, looking again at the schedule in Figure 7(c), we may deform this by planar isotopy, ensuring that the vertical order of nodes is not disturbed, and such that at each point in time it remains a schedule. Figure 27 shows an example of this. One might use a deformation such as this in the "cleaning up" of composite schedules before reuse.

Example 48. For any schedule, the following are examples of deformations which we will use a number of times in this paper:

1. A translation of that schedule in the plane.



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

Figure 27: A "time-lapse" view of a deformation of the schedule in Figure 7(c). Arrowheads used to indicate directions have been omitted for clarity.

2. A horizontal or vertical scaling in the plane.

3. 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 28. This will allow us to place the nodes of a schedule wherever required without altering their order or left–right arrangement.

3.1.3 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 49. 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

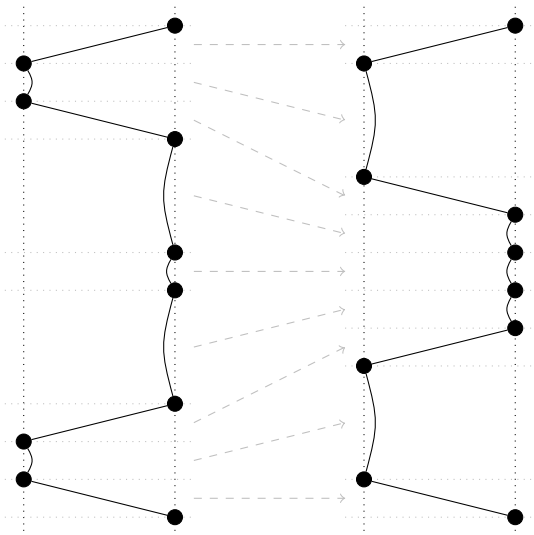


Figure 28: An illustration of a piecewise vertical scale. The strip on the left is horizontally sliced into rectangles, each of which is then independently vertically scaled.

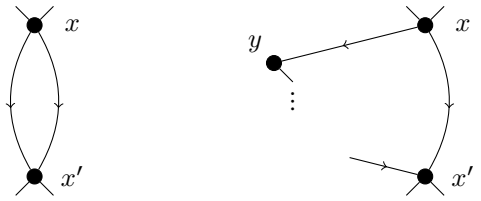
progressive plane graph formed in this way a (**2-fold**) **composition diagram** and denote it $S \cdot S'$; it has nodes $U + V + W$ and an edge for each edge in S and in S'. (We will now not differentiate between vertex sets U, V, W and their chosen embeddings $\iota(U), \iota(V) = \iota'(V), \iota'(W)$ where the context makes it clear what "ϵ" 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.

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 26).

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.

Observe 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.



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

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

Figure 29: Local pictures around internal nodes in composition diagrams

Lemma 50. *The path chosen in Definition 49 is the unique Hamiltonian path (up to deformation) through all nodes of the composition diagram.*

Proof. Let schedules S and S' be as in Definition 49. 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 29(a) and 29(b).

1. *As in Figure 29(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.

2. *As in Figure 29(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 $U + V + W$. □

Based on Lemma 50, we could have *defined* the composite simply as the unique (up to deformation) path through every node in the composition diagram. In case 1., 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 [Mel12b].

Proposition 51. *For schedules $S : U \to V$ and $S' : V \to W$, the composite $S \| S'$ is a schedule $U \to W$.*

Proof. The data of a schedule and conditions on the embedding follow easily from Definition 49, as does condition (Sc1).

Condition (Sc2) 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 52. In the proof of Lemma 50, 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 31(a) and 31(b), though in fact they can never occur.

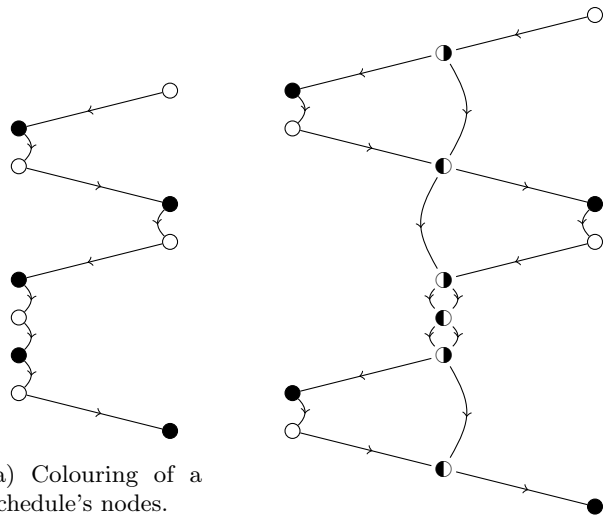
While the reason for this may be derived from (Sc1) and (Sc2) by induction, there is also a "local" proof inspired by the colouring (or O/P-labelling) of nodes found in the literature [AM99, HO00]. Observe that an arbitrary schedule $S_{m,n} : U \to V$ with path order $U + V = \{p_1, \ldots, p_{m+n}\}$ 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 (Sc1) and (Sc2), 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 30(a) shows our original schedule from Figure 7(a) decorated in this way. Observe that (Sc2) is satisfied if and only if this colour scheme is followed.

Edges are always directed $\circ \to \bullet$ if they move from one side to the other (this is the *switching condition* for $\multimap$ [Abr96]). 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 30(b). 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



(a) Colouring of a schedule's nodes.

(b) Colouring of nodes in a composition diagram.

Figure 30: Colouring of nodes.

two cross-schedule edges to the same internal node. Figures 31(c) and 31(d) show the hypothetical fragments with a choice of colours, and the illegal edges marked with a ×. Analogous arguments using state diagrams exist elsewhere in the game semantics literature; for example, [Abr96, Har00].

Definition 53. Let $S : U_m \to V_n$ be a $\multimap$-schedule. The **truncation to** j 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 54. Figure 32 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 $S \restriction_7 : 2 \to 5$.

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

Proposition 55. *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 $(S \| T) \restriction_k$. □

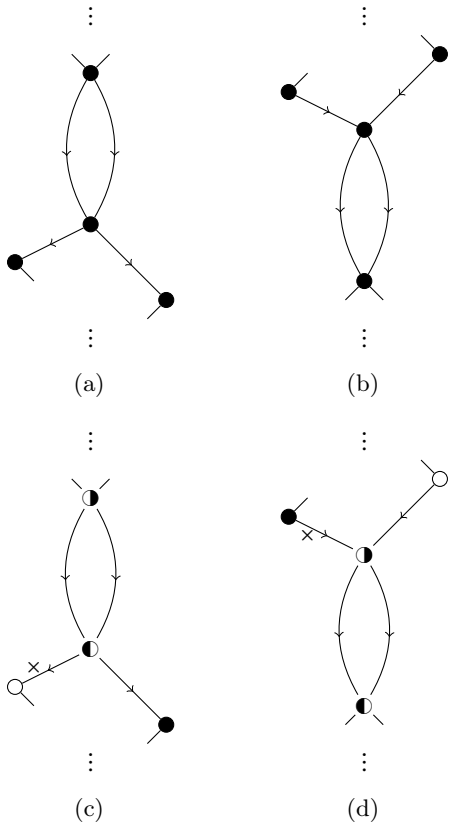


Figure 31: Colouring of internal nodes in a composition diagram.

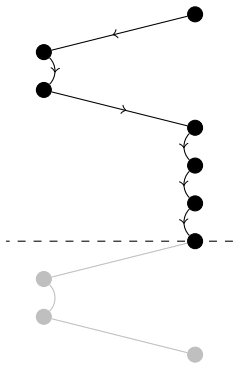


Figure 32: A truncation $S\restriction_7$ of a $\multimap$-schedule S. The dashed line shows the horizontal line through the node p_7 below which all nodes and edges are removed.

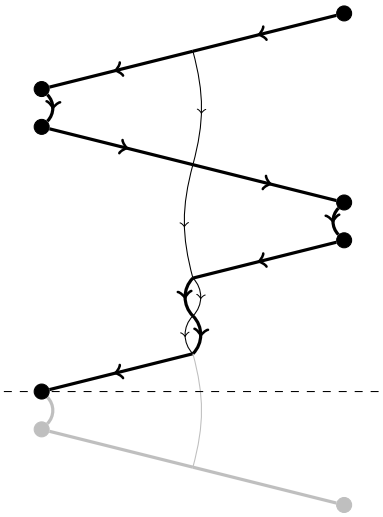


Figure 33: Truncation of a composite $\multimap$-schedule. The thick lines are edges of $S \| T$, the thin lines are edges of S and T which are not taken in the composite, the grey lines and nodes are those removed in the truncation.

An example of this scenario is shown in Figure 33.

3.1.4 The category *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 56. *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 34.

$(S \| S') \| S''$ 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 50, 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.

Remark 57. This is not dissimilar to the proof of associativity of the composition and tensor product in $\mathscr{F}_\Sigma^{\mathrm{dia}}$ described in detail in Chapter 2. In both cases, the associativity of a graphical construction follows essentially from the natural associativity inherent in the geometry of plane graphs. By using the plane, we get facts like "left of left is left" for free. In a more combinatorial setting, some kind of reindexing lemma would be required, though would often be elided.

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 58. 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 35. Alternatively, these copycat schedules may be characterised by saying that also $\{p_{2k+1}, p_{2k+2}\} \notin U_n$ and $\notin U'_n$.

Lemma 59. *Copycat schedules $I_{n,n}$ are the identities of schedule composition.*

Proof. We will argue for composition on the left with a copycat schedule, and an entirely analogous argument can be made for composition on the right.

Let $S_{m,n} : U \to V$ be a schedule and let $I_{m,m} : U' \to U$ be a copycat schedule. (We give an example of such in Figure 36(a).) We want to show that $I_{m,m}\|S_{m,n} \sim S_{m,n}$. Since $S_{m,n}$ is a $\multimap$-schedule, nodes in U come (taken top-to-bottom) in pairs connected by an edge in S. Consider such a pair u_{2k+1}, u_{2k+2}. Since $I_{m,m}$ is a copycat schedule $U' \to U$, the outward path from u_{2k+1} in I crosses to a pair of nodes (u'_{2k+1} and u'_{2k+2}) in U' before immediately returning to u_{2k+2}.

In the composite $I\|S$, the nodes u_{2k+1} and u_{2k+2}, and the edge in S connecting them are removed. The path in S passing through u_{2k+1} instead continues to u'_{2k+1} and then u'_{2k+2} before returning to points in S. All other parts of S are unchanged in the composite $S\|I$.

There is a simple isotopy taking the edge $u'_{2k+1} \to u'_{2k+2}$ in $I\|S$ to the edge $u_{2k+1} \to u_{2k+2}$ in S, an illustration of which can be seen in Figure 36(b). □

Lemma 59 and Theorem 81 together prove:

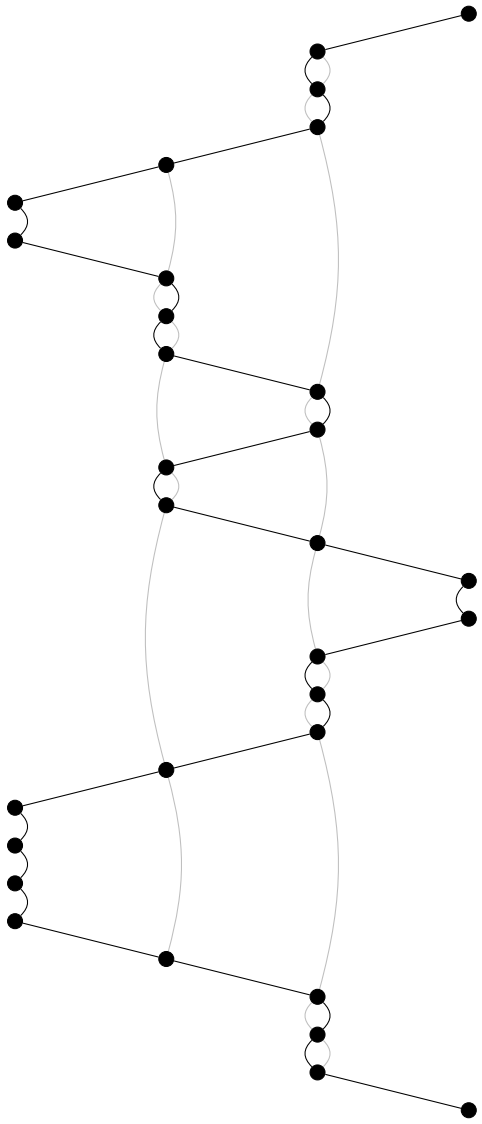


Figure 34: A three-way composition diagram with composite path highlighted. Note that, since we must always cross between schedules on reaching an internal node, there are no choices to be made in determining the composite.

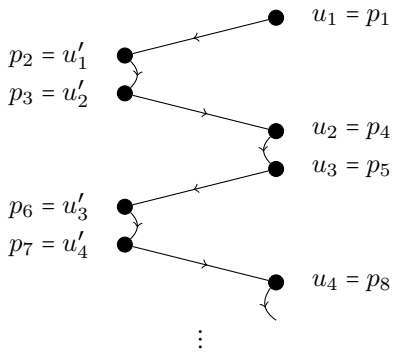


Figure 35: A prefix fragment of a copycat schedule.

Theorem 60. *Positive natural numbers, together with the graphical schedules up to deformation form a category, called* $\mathcal{Sched}$*, where composition is defined by Definition 49 and identities are copycat schedules.*

$\mathcal{Sched}$ is isomorphic to Υ, which we will demonstrate 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

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

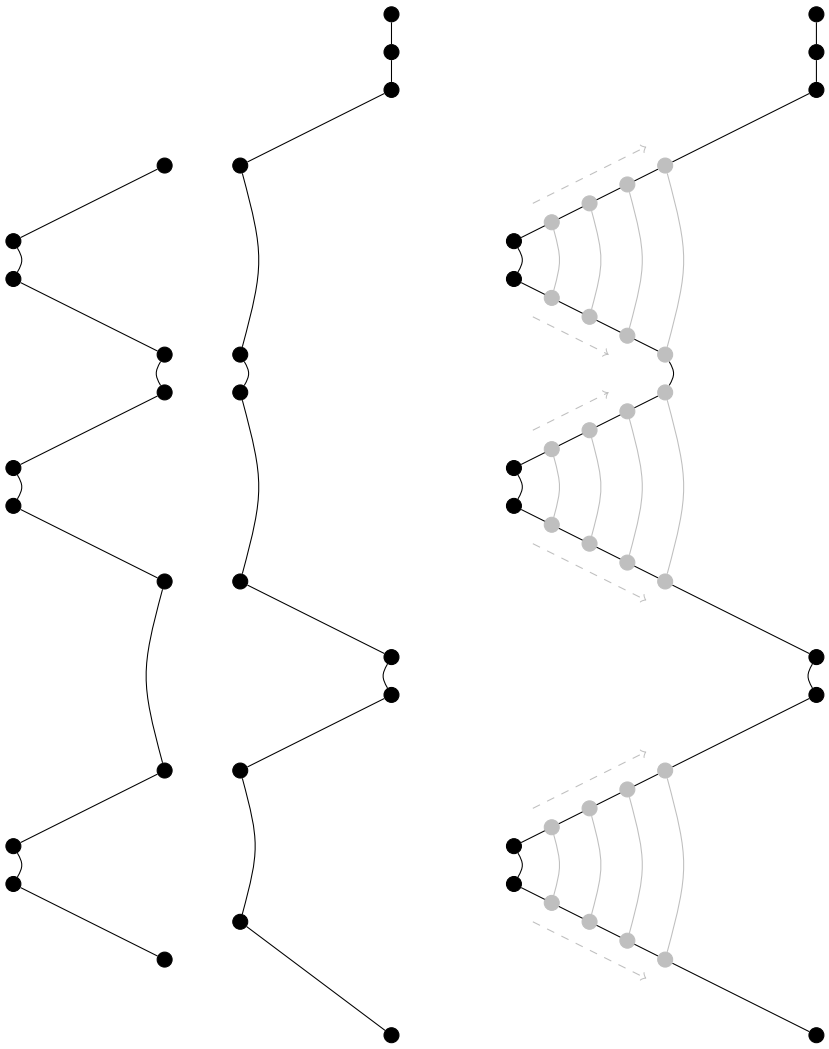
In the combinatorial terms of [HHM07], a schedule $e : m \to n$ corresponds to injections $e_L : [m] \hookrightarrow [m + n]$ and $e_R : [n] \hookrightarrow [m + n]$, which in turn correspond to order relations $e_L(x) < e_R(y)$ from $[m]^+$ to $[n]^+$ and $e_R(y) < e_L(x)$ from $[n]^-$ to $[m]^-$. Thinking 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 61. *C is a functor $\mathcal{Sched} \to \Upsilon$.*

Theorem 62. *$C : \mathcal{Sched} \to \Upsilon$ 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 graphical 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



(a) Example of a copycat schedule which may be composed on the left with another schedule.

(b) A "time-lapse" view of the deformation of the composite to the original schedule.

Figure 36: Composition with a copycat schedule on the left.

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 $\pi_1 p_i = \pi_1 p_{i+1}$).

$CG = \mathrm{id}$ 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. □

3.2 Games and strategies

A core notion in game semantics is (unsurprisingly) that of a *game*. A game, roughly speaking, is a description of an alternating play between a player "P" (representing a program) and an opponent "O" (representing the program's environment). The *moves* each participant can play are recorded in a forest, starting with specified legal initial moves for O, and with children given all legal responses to a particular *position*. A *strategy* for P is a deterministic set of instructions detailing a response to each possible move that O can make.

3.2.1 Games

We adapt our specific definition of game from Definition 3 of [HHM07] in order that it more naturally fit with later use of schedules. This definition bears similarity to that of Lamarche [Lam95].

Definition 63. 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

$$\underline{a} = (\pi^{n-1}(a), \pi^{n-2}(a), \ldots, \pi(a), a) \tag{3.1}$$

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 program [Abr96].

Remark 64. 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. [Abr96] 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. Further discussion of this can be found in [Lam95].

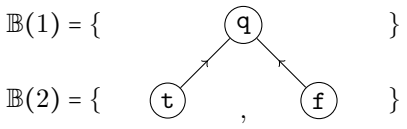
As can be seen in Example 65, we may completely recover a game A from its labelled forest. Initial positions are roots, elements of $A(k)$ are those nodes at depth k and π assigns to each node its parent.

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 [Abr96, Hyl97].

Example 65. 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



3.2.2 Strategies

We similarly take our definition of *strategy* to be similar to Definition 4 of [HHM07].

Definition 66. A **strategy** σ for a game A is given by a graded subset $\sigma(2k) \subseteq A(2k)$ satisfying

$$\pi^2(\sigma(2k+2)) \subseteq \sigma(2k) \tag{St1}$$

$$x, y \in \sigma(2k) \text{ and } \pi(x) = \pi(y) \text{ then } x = y \tag{St2}$$

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 exactly one response. We can think of (St1) as saying that σ is closed under double predecessor, guaranteeing that every position of σ is reachable. We can think of (St2) as saying that σ is deterministic.

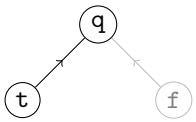


Figure 37: The forest for the strategy $\mathtt{t} : \mathbb{B}$. This forest is a subforest of the game tree for $\mathbb{B}$, which is shown in grey. Technically, only the move labelled with $\mathtt{t}$ is in the strategy, but this subforest depiction is convenient.

Remark 67. As was noted in Remark 64, 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, (St1) 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 (St2).

Just as games are thought of as modelling types, strategies for a game can be thought of as modelling terms of that game's type [Abr96].

Example 68. The term $\mathtt{t}$ of type $\mathbb{B}$ is modelled by the strategy σ given by

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

As described in Remark 67, Figure 37 exhibits $\mathtt{t} : \mathbb{B}$ as a subtree of the tree for $\mathbb{B}$ given in Example 65. We show not only the nodes in σ, but also all nodes in $\pi(\sigma)$, so that the rooted paths in the forest are still plays of the game according to σ.

3.2.3 $\multimap$-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.

Definition 69. 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 $v_j \in V_n$. (Subscripts on u_i and v_j indicate their order in U and V respectively.) 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.

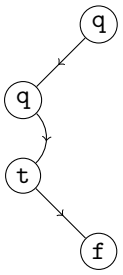


Figure 38: A $\multimap$-schedule $S : 2 \to 2$ play-labelled in the games $\mathbb{B}$ and $\mathbb{B}$.

Definition 70. 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**.

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 71. 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 49, 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 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 38. In this way, we can largely avoid working with labelling functions explicitly, and instead work directly with decorated graphs.

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

Definition 72 (Cf. *lifting of arenas* in [Lau04].)**.** 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 73. 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

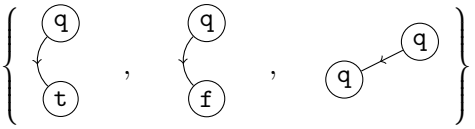
$$\pi_{A\multimap B} : (S, a, b) \mapsto \begin{cases} (S \restriction_{k-1}, \pi_A(a), b) & \text{if } p_k \in U_m \\ (S \restriction_{k-1}, a, \pi_B(b)) & \text{if } p_k \in V_n \end{cases}$$

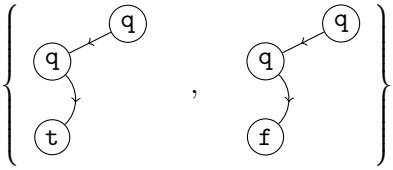
We will often use the notation (S, a, b) to refer to a schedule $S : m \to n$ play-labelled by $a \in A(m)$ and $b \in B(n)$. We will also use the notation $(S \cdot T, a, b, c)$ to refer to the labelled composition diagram for the composition $(S, a, b) \| (T, b, c)$.

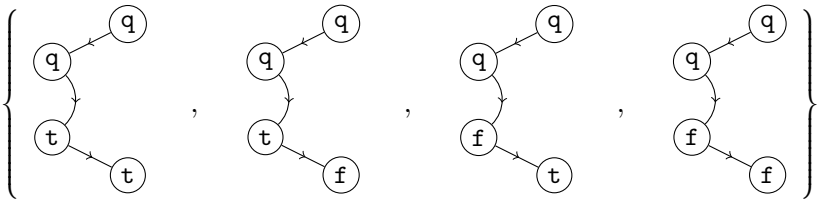
In $A \multimap B$, positions are given by a $\multimap$-schedule and the most recent moves in the component games A and B. The triple (S, a, b) lets us draw a $\multimap$-schedule with a play labelling of U_m with $\underline{a}$ and V_n with $\underline{b}$. Then, the predecessor function $\pi_{A\multimap B}$ maps (S, a, b) to $(S \restriction_{k-1}, a', b')$, 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 lifted game $\hat{A}$ 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 74. 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.

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

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

$(\mathbb{B} \multimap \mathbb{B})(3) =$ 

$(\mathbb{B} \multimap \mathbb{B})(4) =$ 

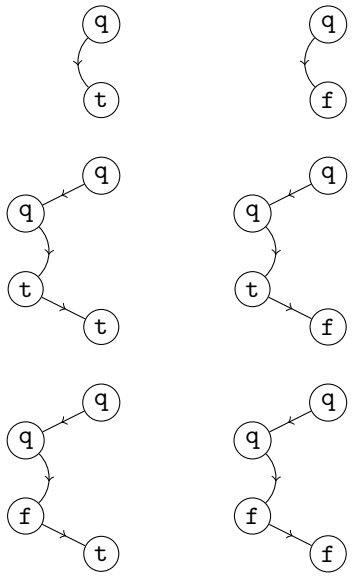


Figure 39: All complete plays for $\mathbb{B} \multimap \mathbb{B}$.

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 39, 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.

Remark 75. 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 truncating. This also gives us the natural forest structure of the game.

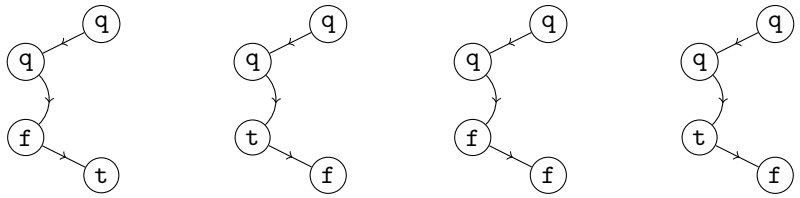
We can therefore give a strategy σ on an arrow game by specifying a set of "maximal" even-length labelled $\multimap$-schedules whose even-length truncations give the remaining positions in σ. As in Remark 67, this set of maximal $\multimap$-schedules will subject to the condition that the longest common truncation is of any two members is of even length.

Example 76. Consider the strategy given by the $\multimap$-schedules in Figure 40(a). 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 $x \mapsto \neg x$. Similarly, the strategy $\mathbb{B} \multimap \mathbb{B}$ which is constantly `f` is shown in Figure 40(b).

3.3 The category of games

3.3.1 Composition of strategies

Strategies on arrow games comprise the morphisms of the category of games. Here, we adapt the description of strategy composition from Definition 7 of [HHM07].



(a) The $\multimap$-schedules giving all complete plays of the strategy "$\neg$".

(b) A strategy denoting the function which evaluates to `f`.

Figure 40: Two strategies for the game $\mathbb{B} \multimap \mathbb{B}$.

Definition 77. A strategy $\sigma : A \multimap B$ may be **composed** with a strategy $\tau : B \multimap C$ to give a set $\sigma \| \tau : A \multimap C$ where

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

We use the following lemma to show that $\sigma \| \tau$ is a strategy $A \multimap C$.

Lemma 78. *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) $(S \| T, a, c) = (S' \| T', a', c')$

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

(iii) 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 then (i) holds. If they differ because one is a prefix of the other then (ii) 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 52.

2. *x is an A-node and x' is a B-node, or vice versa.* From the switching condition in Remark 52, we see that x, x' are both at an even-length position (they are P-moves) meaning that σ would violate (St2), 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 (St2).

(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 (St2). In this case, (iii) holds.

□

Theorem 79. *Strategies are closed under composition. In other words, if $\sigma : A \multimap B$ and $\tau : B \multimap C$ then $\sigma \| \tau$ is a strategy for $A \multimap C$.*

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 (St1)) by Proposition 55, since the truncation of $S \| T$ can be seen to act on S and T, and σ and τ satisfy (St1).

Lemma 78 guarantees that the longest common prefix of any two schedules in $\sigma \| \tau$ is of even length, and thus that $\sigma \| \tau$ satisfies (St2). □

Example 80. The strategy denoting the function $x \mapsto \neg x$ shown in Example 76 may be composed with either of the strategies for functions which are constantly `f` also given in Example 76. 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 40(a) and the strategy for $x \mapsto$ `(if` x `then f else f)` is given by the set of labelled $\multimap$-schedules of Figure 40(b).

Figure 41(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 41(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.

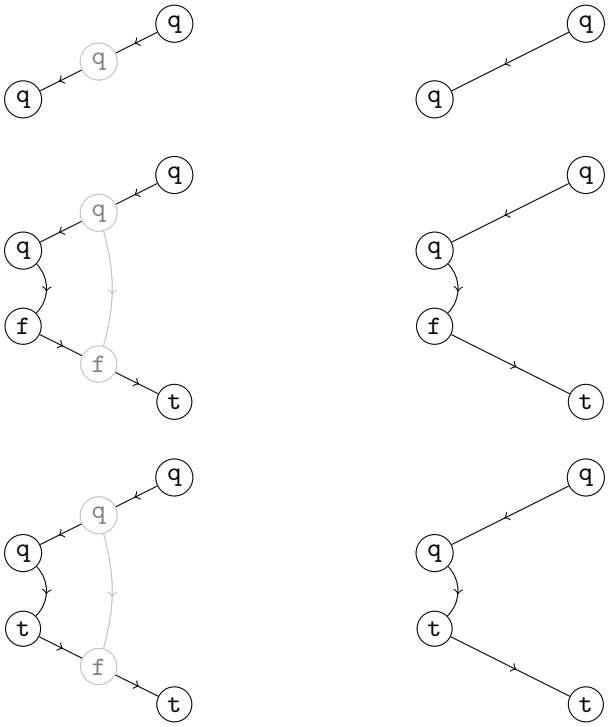
3.3.2 The category *Game*

Theorem 81. *Composition of strategies is associative.*

Proof. This follows from the fact that composition of $\multimap$-schedules is associative; Proposition 56. □

Definition 82. 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.



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

Proposition 83. *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 59, copycat $\multimap$-schedules are identities of $\multimap$-schedule composition. □

Theorem 81 and Proposition 83 together provide us with the following result.

Theorem 84. *Games and strategies form a category.*

And thus we make the following definition (cf. [HHM07], Definition 7.).

Definition 85. The **category of games** $\mathcal{G}ame$ from Theorem 84 is given as follows:

- The objects of $\mathcal{G}ame$ are games .
- $\mathcal{G}ame(A, B) = \{\text{strategies } \sigma : A \multimap B\}$, with composition given by composition of strategies.
- The identity map $A \to A$ is given by the copycat strategy $\kappa_A : A \multimap A$.

Theorem 86. *The category $\mathcal{G}ame$ is isomorphic to the category of games and strategies $\mathscr{G}$ from [HHM07], 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 62. □

3.3.3 ⊗-schedules and tensor games

Along the same lines as the definition of $\multimap$-schedule, we make the following definition for a ⊗-*schedule*, which will describe the interleaving of plays in ⊗-composition of two games [Abr96, HHM07].

Definition 87. A **⊗-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:

$$\begin{aligned} \text{for each } k \geq 0, \text{ either } \{p_{2k+1}, p_{2k+2}\} \subset U \\ \text{or } \{p_{2k+1}, p_{2k+2}\} \subset V \end{aligned} \tag{3.2}$$

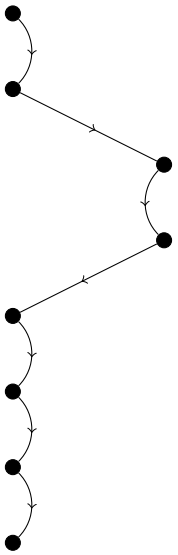


Figure 42: A ⊗-schedule $S^{\otimes}_{6,2}$.

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

Example 88. Figure 42 shows an example of a ⊗-schedule.

We may easily extend Definitions 47, 53 and 69 to describe the *deformation*, *labelling* and *truncation* of ⊗-schedules.

Definition 89. 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 ⊗-schedule with $m + n = k$ and path order $\{p_1, \ldots, p_k\}$. The predecessor function $\pi_{A \otimes B}$ is given by

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

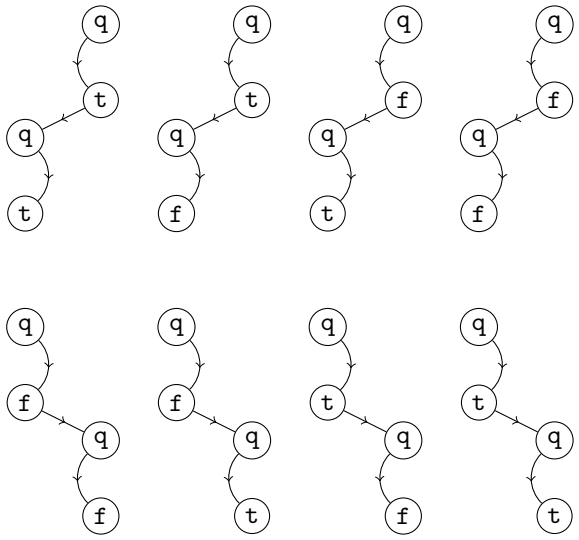


Figure 43: All complete plays for $\mathbb{B} \otimes \mathbb{B}$.

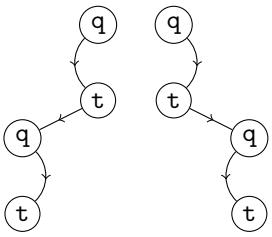


Figure 44: The schedules giving all complete plays of the strategy $\mathtt{t} \otimes \mathtt{t} : \mathbb{B} \otimes \mathbb{B}$.

Example 90. The game $\mathbb{B} \otimes \mathbb{B}$ is given by its graded sets which, in a similar manner to Example 74, we may specify entirely by the leaves of the game tree, which are labelled schedules. Figure 43 shows all complete plays for $\mathbb{B} \otimes \mathbb{B}$, with $\pi_{\mathbb{B} \otimes \mathbb{B}}$ given by truncation.

From Remark 67 we may give a strategy on a game A as a set of truncations of complete plays of A. From Remark 75 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 91. The strategy $\mathtt{t} \otimes \mathtt{t} : B \otimes B$ is given by the labelled schedules in Figure 44.

3.4 Graphical representations of games with more than two components

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 $B \otimes C$. The position x is therefore itself a triple (T, b, c), with T a $\otimes$-schedule, b a position of $\hat{B}$ and c a position of $\hat{C}$.

From our definitions so far, a position of $A \multimap (B \otimes C)$ would be a $\multimap$-schedule whose left-hand nodes are labelled $(\ldots, \pi^2_A(a), \pi_A(a), a)$ and whose right-hand nodes are labelled with $(\ldots, \pi^2_{B\otimes C}(T,b,c), \pi_{B\otimes C}(T,b,c), (T,b,c))$.

An example of such a position is shown in Figure 45(a). The enlarged nodes on the right are labelled with $\otimes$-schedules which show the play in $B \otimes C$.

This representation is useful for the purpose of performing composition calculations (as we have a concrete notion of $\multimap$-schedule composition), but does not reflect the intuitive arguments used by researchers who frequently argue about multi-component games in similar ways to two-component games [Hyl97, AM99, HO00]. 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 45(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.

3.4.1 Interleaving graphs

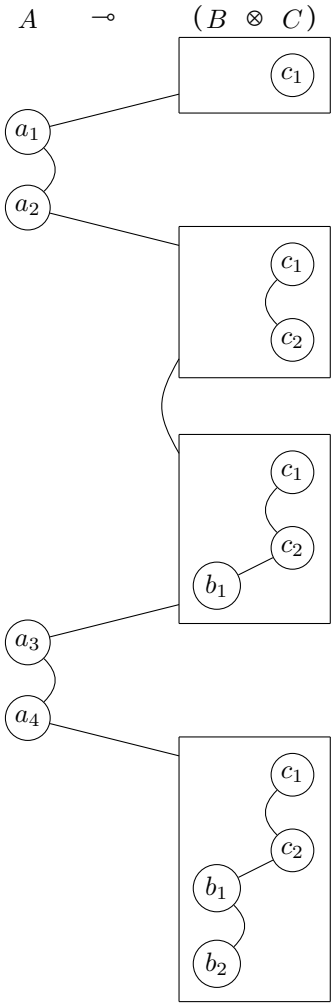
We begin, as we did with schedules, with unlabelled graphs.

Definition 92. An **n-interleaving graph** is a list

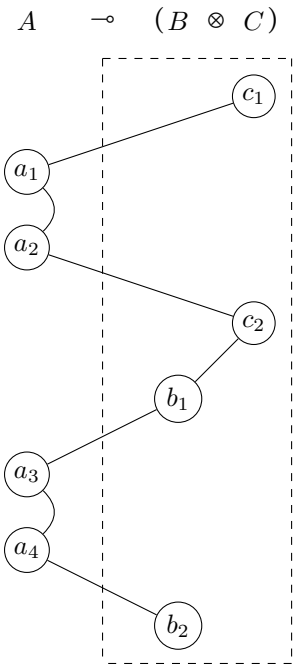
$$S_{m_1,\ldots,m_n} = (U^{(1)}, \ldots, U^{(n)}, \Sigma, \iota)$$

consisting of the following data:

- 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 $U^{(i)}$.



(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 45

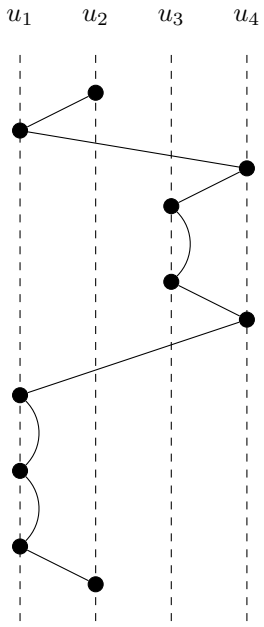


Figure 46: An example of a 4-interleaving graph.

- 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 93. The following are examples of n-interleaving graphs.

- All $\multimap$-schedules and $\otimes$-schedules are examples of 2-interleaving graphs.
- Figure 45(b) shows an example of a 3-interleaving graph whose nodes are labelled with moves from games A, B and C.
- Figure 46 shows an example of an 4-interleaving graph.
- Figure 47 shows an example of a 5-interleaving graph.

We will frequently describe interleaving graphs "up to deformation" in a way similar to Definition 47. 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 94. In the more general case,

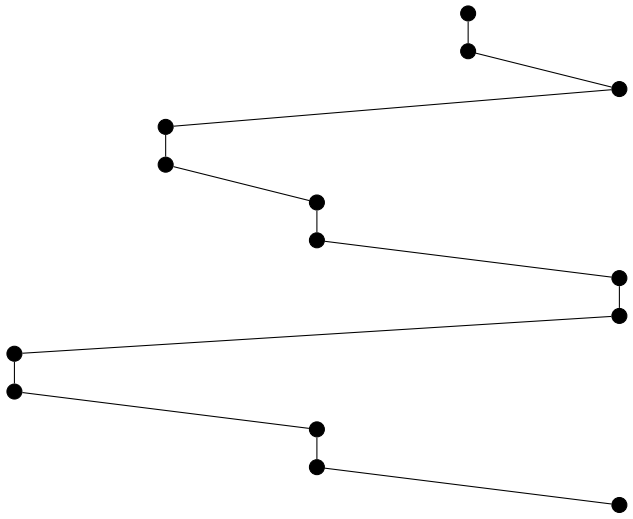


Figure 47: An example of a 5-interleaving graph

when we are manipulating the structure of interleaving graphs in Section 3.4.3, 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 the more generalised notion of deformation, which we recall from Definition 17.

In general when we speak of deformations of interleaving graph, and in particular interleaving graphs "up-to-deformation", we mean deformation *as interleaving graphs*, though this may go unsaid. Only in Section 3.4.3 and Proposition 117 will we use the more general 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.

Definition 94. Let S and S' be n-interleaving graphs

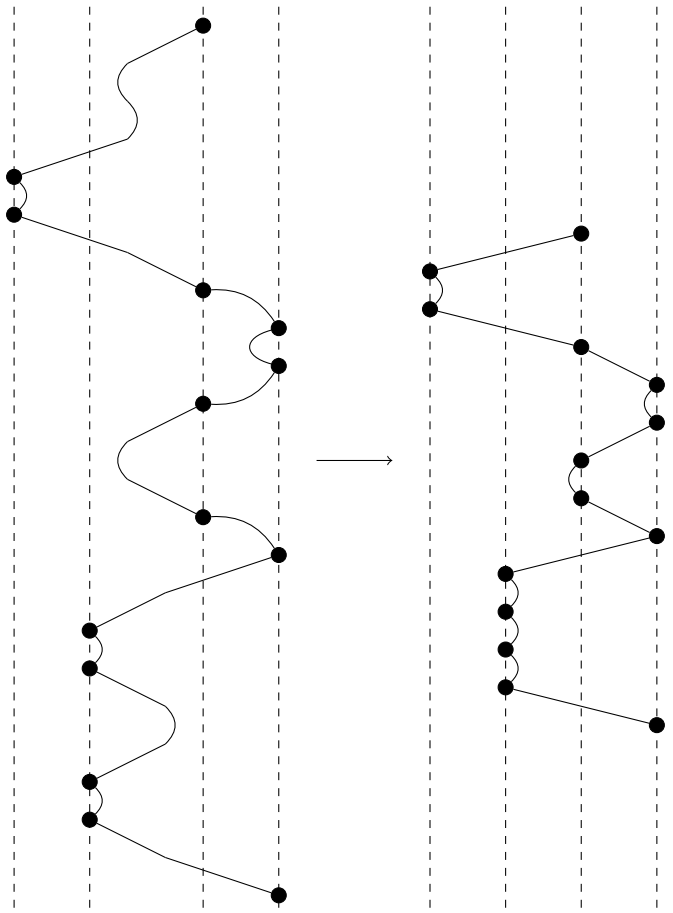
$$S = S_{m_1,\dots,m_n} = (U_{m_1}, \dots, U_{m_n}, \Sigma, \iota)$$

$$S' = S'_{m_1,\dots,m_n} = (U'_{m_1}, \dots, U'_{m_n}, \Sigma', \iota')$$

in the plane. We say that S **is deformable into** S' (**as an n-interleaving graph**) if there is a deformation $h : \Sigma \times [0,1] \to \mathbb{R}^2$ of Σ and ι into Σ' and ι', such that for each $t \in [0,1]$, $h(-,t)$ is an embedding $\hat{\Sigma} \hookrightarrow \mathbb{R}^2$ as an n-interleaving graph.

Example 95.

- Translations and piecewise scalings are examples of deformations as interleaving graphs.



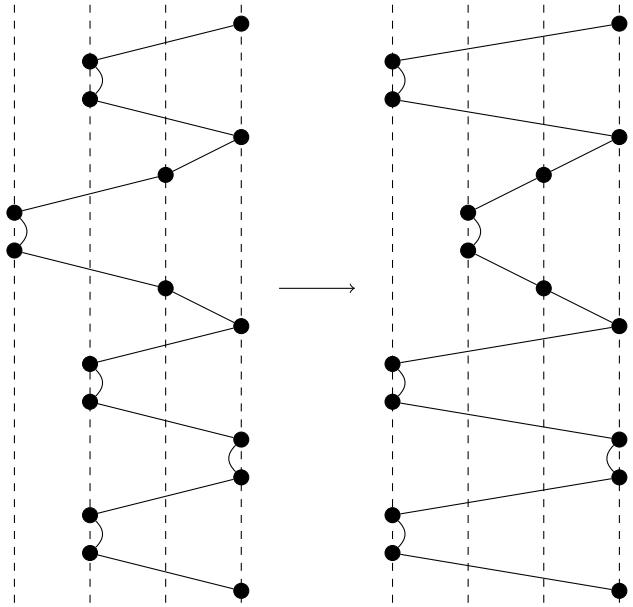
(a) Deformation as an interleaving graph. Example from Figure 71(f).

Figure 48: Two "deformations" of interleaving graphs.

- Figure 48(a) shows an example of a deformation of an interleaving graph *as an interleaving graph*.
- Figure 48(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.

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 in game built with $\otimes$ and $\multimap$. 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 96. Let $S = (U^{(1)}, \ldots, U^{(n)}, \Sigma, \iota)$ be an n-interleaving graph in $[u_1, u_n] \times \mathbb{R}$. We consider two fundamental operations on it.



(b) Deformation as a progressive plane graph. Example from Figure 51(a).

Figure 48: (Continued.)

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

$$[S]^{(u_i,u_j)} = (U^{(1)}, \dots, U^{(i)} + \dots + U^{(j)}, \dots, U^{(n)}, \Sigma', \iota')$$

where Σ' is a deformation of Σ with embedding ι so that the nodes in $U^{(i)} + \dots + U^{(j)}$ 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.

Restrict: For a subset

$$\bar{U} = \{u_{i_1}, \dots, u_{i_j}\} \subseteq \{u_1, \dots, u_n\} = U$$

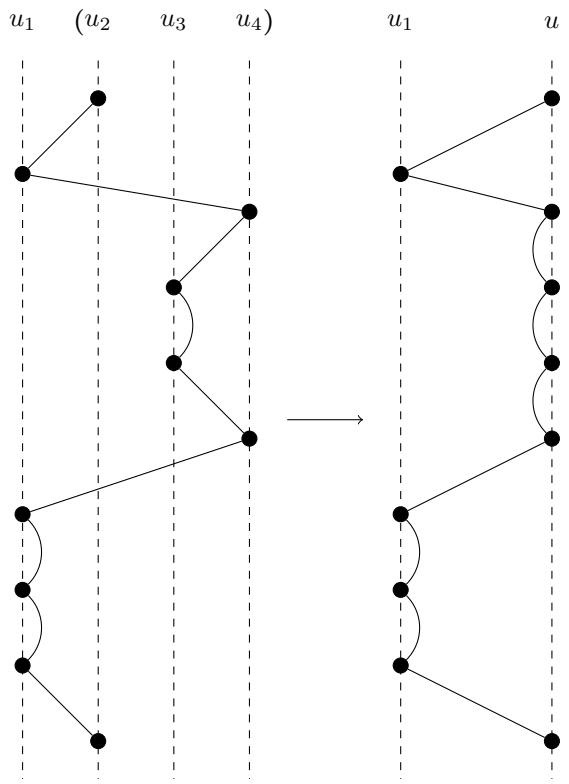
we form a j-interleaving graph

$$[S]_{i_1,\dots,i_j} = (U^{(i_1)}, \dots, U^{(i_j)}, \Sigma', \iota')$$

where Σ' is achieved from Σ by gluing (declassifying) all nodes in $U \setminus \bar{U}$ and discarding any half-edges.

From these we derive an additional useful operation:

Segregate: For some $u \in (u_1, u_n)$, pick u_i with $u_i < u < u_{i+1}$. Collapse nodes on $\{u_1, \dots, u_i\}$ to u_1 and collapse nodes on $\{u_{i+1} \dots, u_n\}$ to u_n.



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

Figure 49: Examples of operations on n-interleaving graphs.

Example 97. Figures 49(a), 49(b) and 49(c) show examples of the three operations on n-interleaving grpahs.

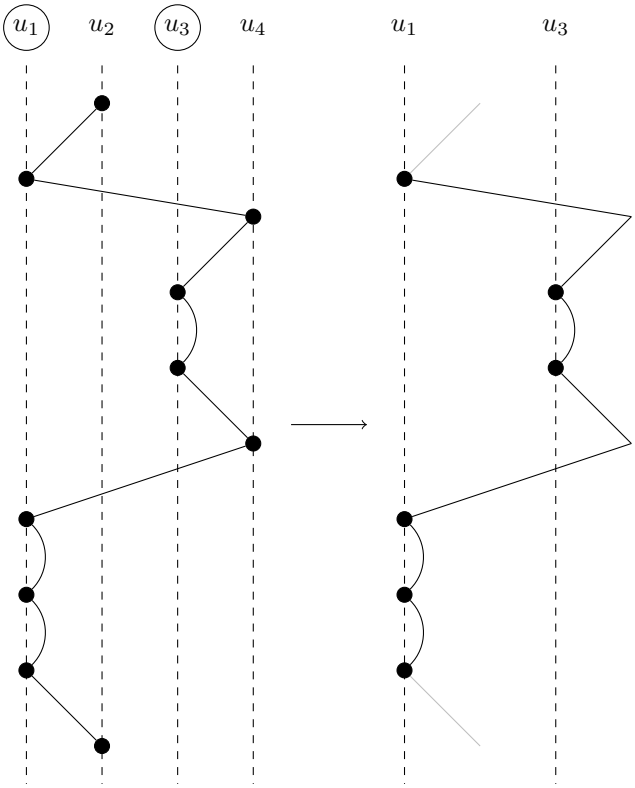
We are now able to classify those interleaving graphs which may represent plays in games whose compositional structure is known. If a game has a structure defined by a binary $(\multimap, \otimes)$-word w, we wish our interleaving graph to exhibit the appropriate qualities when the interleaving is considered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

Definition 98. 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:

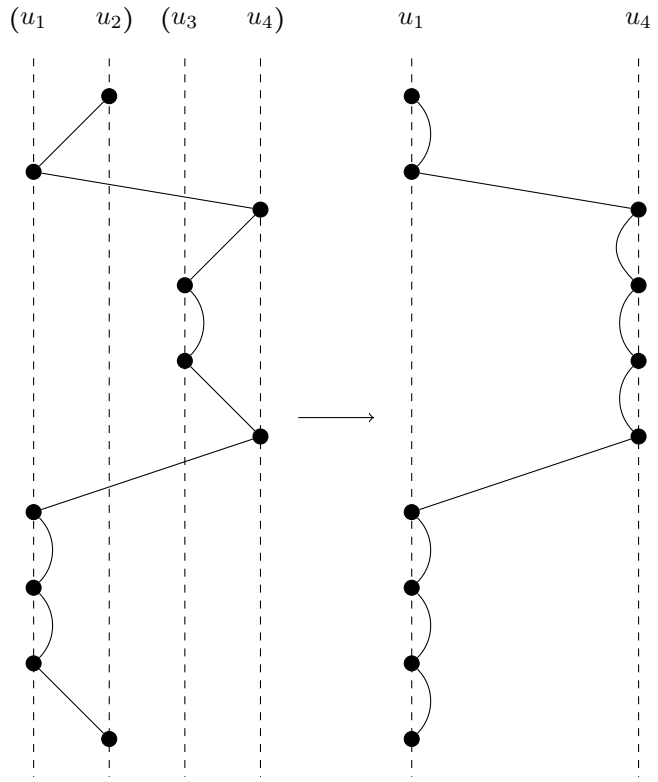
(i) For each bracketed sub-word of w, the corresponding restriction of S yields a graph suitable for that sub-word.

(ii) For each connective χ_i of w, restricting S to the bracketed sub-word whose major connective is χ_i and then segregating on some u strictly between the u_i, u_j



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

Figure 49: (Continued.)



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

Figure 49: (Continued.)

corresponding to the closest letter symbols to that connective yields a 2-interleaving graph which is a schedule for that connective.

Example 99. See Appendix A.3.1 for an example demonstration of the suitability of the 5-interleaving graph from Figure 47 for the word $(A \otimes B) \multimap (C \multimap (D \otimes E))$.

Definition 100. 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 $U^{(i)} + \cdots U^{(j)}$.

3.4.2 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 3.4 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 our game A, when we consider the interleaving of play across a connective of w, we find a schedule for that connective. Therefore, the n-interleaving graph which represents the play must be such that, when segregated over this connective, is the same as the schedule in A. Also, we require that when restricting the n-interleaving graph to some bracketed subword of w, we get an interleaving graph which describes play of A when attention is restricted to just that subword. Both of these requirements are 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 101. 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

- $S = S_{k_1,\ldots,k_n} = (U_{k_1}, \ldots, U_{k_n}, \Sigma, \iota)$ 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 $p_k \in U_{k_i}$.

Example 102.

1. 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, with schedules viewed as 2-interleaving graphs.
2. The unfolded form of the game $(\mathbb{B} \otimes \mathbb{B}) \otimes \mathbb{B}$ consists of all 3-interleaving graphs of length 6 whose nodes come in pairs in each of the three columns, with each pair either labelled with $\mathtt{q}, \mathtt{t}$ or $\mathtt{q}, \mathtt{f}$.

   Notice that this is the same as the unfolded form of the game $\mathbb{B} \otimes (\mathbb{B} \otimes \mathbb{B})$.

3.4.3 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 103. 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, 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 follows 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 bracketed sub-words in a recursive fashion.)

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

Example 104.

- Figure 45(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 45(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.

- Appendix A.3.2 shows a further example, working through the unfolding of a 3-interleaving graph with nodes labelled with a schedule into a 4-interleaving graph.

Proposition 105. *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 106. 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 107. *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 108. *A game A of the form $A_1 \,\square\, A_2$ is isomorphic to its unfolded form $\tilde{A}$.*

Corollary 109. *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.

3.5 Symmetric monoidal closed structure on *Game*

In this section we will extend the notion of $\otimes$ from an operation on the objects of *Game* to the morphisms, providing a monoidal structure on *Game* [JS91, Mac97].

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 110. 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.

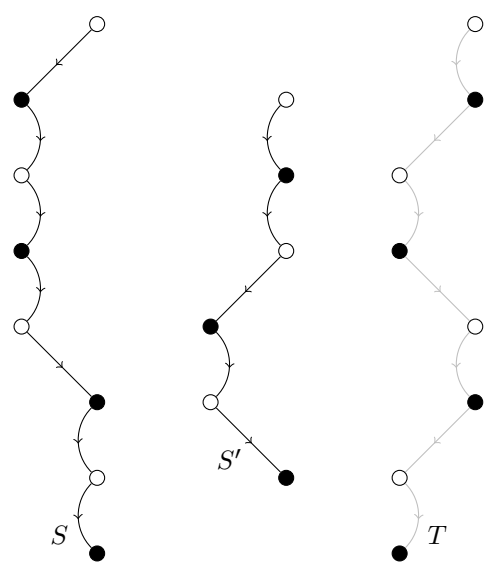
This gives us a kind of composition graph from which the appropriate 4-interleaving graph, which we call $S \otimes_T S'$ may be extracted.

- 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 $[v, v'] \times \mathbb{R}$.
  - All unselected edges are discarded.

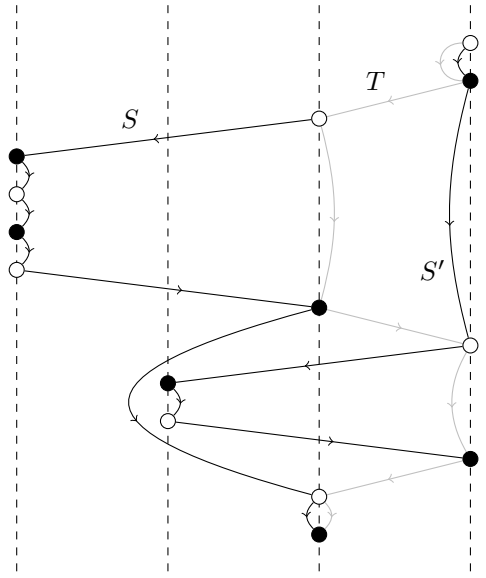
Example 111. Figure 50 shows an example of this construction.

The construction of Definition 110 may be straightforwardly extended to the construction of a *labelled* 4-interleaving graph from two labelled $\multimap$-schedules and a $\otimes$ schedule.

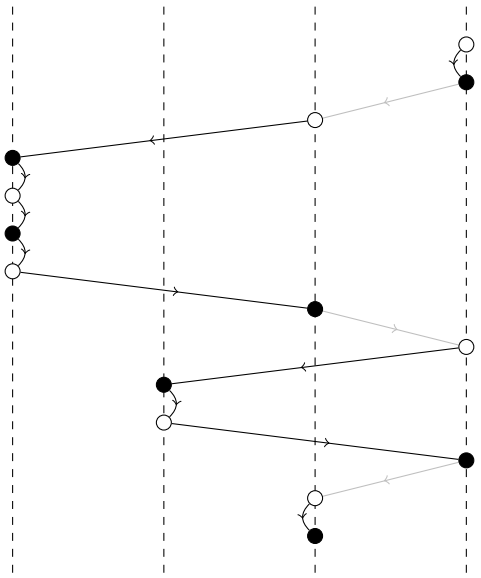
The following follow from the recursive definition of suitability.



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



(b) Putting the components together



(c) $S \otimes_T S'$.

Figure 50: Example of construction $S \otimes_T S'$.

Proposition 112. *With $\multimap$-schedules S suitable for $A \multimap B$, S' suitable for $A' \multimap B'$ and $\otimes$-schedule T suitable for $B \otimes B'$, the 4-interleaving graph $S \otimes_T S'$ is suitable for $(A \otimes A') \multimap (B \otimes B')$.*

Corollary 113. *$S \otimes_T S'$ thus uniquely determines a $\otimes$-schedule, found by restricting to its $(A \otimes B)$-nodes. We call this schedule $\check{T}$.*

Remark 114. 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 T 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 $\mathcal{G}ame$.

Definition 115. 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 116. *$\otimes$ is a tensor product on $\mathcal{G}ame$.*

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 precis of the proof of bifunctoriality. For full details see Appendix A.3.3, with illustrations in Figure 71.

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 109. See also: item 2 of Example 102. □

Proposition 117. *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 51(a).

Now we have that $\bar{S} \| G = S$, as desired. This is illustrated by example in Figure 51(b).

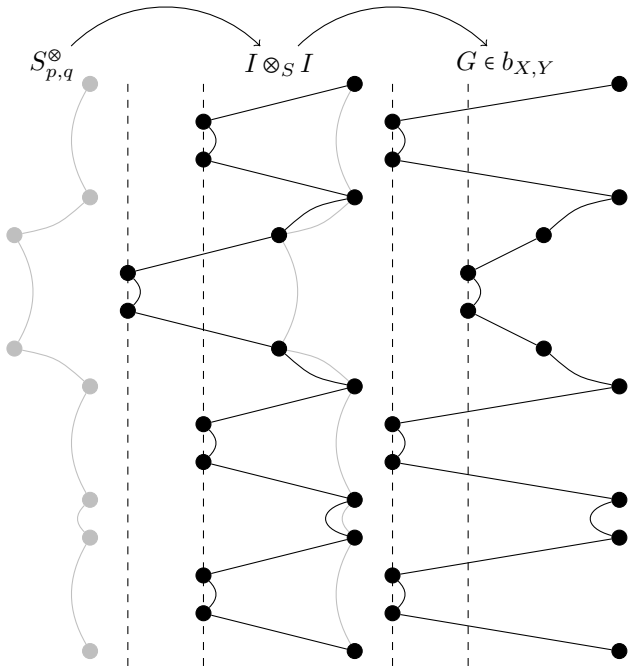
The strategy $b_{X,Y} : (X \otimes Y) \multimap (Y \otimes X)$ is then the collection of all such labelled 4-interleaving graphs, for each labelled $\otimes$-schedule in $Y \otimes X$. The strategy $b_{X,Y}$ is self-inverse by construction. □

Proposition 118. *For games* A, B, C*, we have*

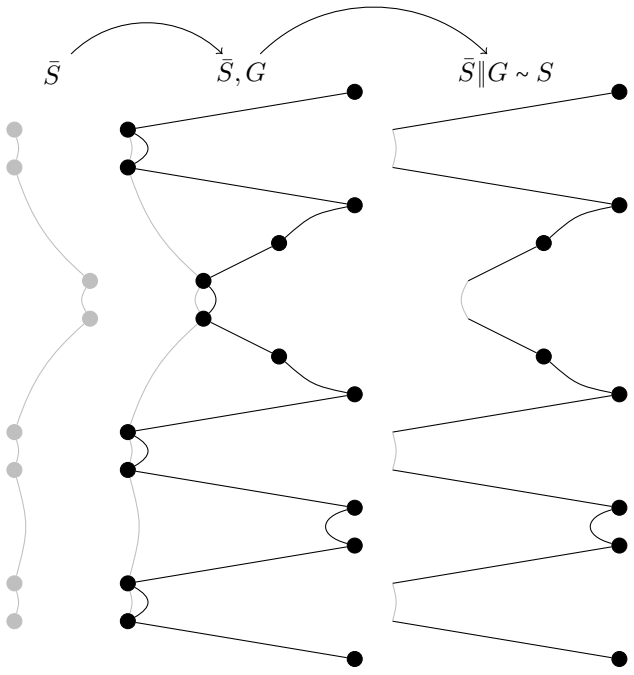
$$(A \otimes B) \multimap C \cong A \multimap (B \multimap C)$$

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 109 allows us to make such an argument. □

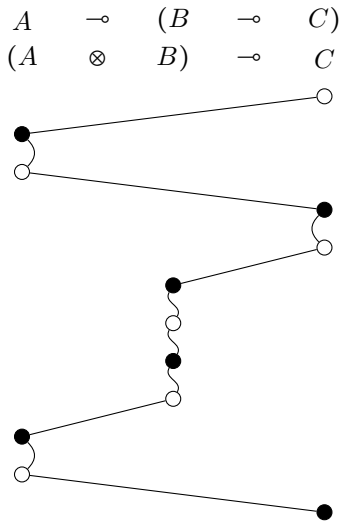


(a) Construction of G from $I \otimes_S I$.



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

Figure 51: Symmetric structure on $\mathcal{G}ame$.



(a) An example 3-interleaving graph.

Figure 52: A demonstration that a particular 3-interleaving graph is suitable for both $A \multimap (B \multimap C)$ and $(A \otimes B) \multimap C)$.

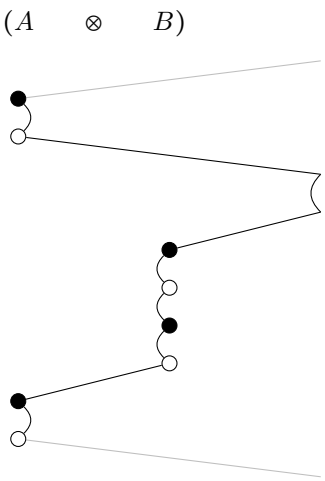
Proposition 118 is illustrated by example in Figure 52.

With Propositions 116, 117 and 118 providing a structure on $\mathcal{G}ame$, we get the following important results [See87, HHM07].

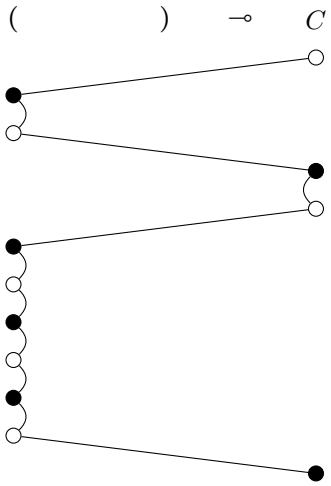
Theorem 119. *$\mathcal{G}ame$ has a symmetric monoidal closed structure.*

Corollary 120. *$\mathcal{G}ame$ models multiplicative intuitionistic linear logic.*

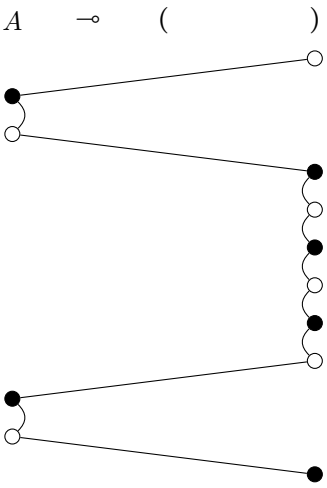
Remark 121. Proposition 118 is a mild rephrasing of the intuitive argument "the game $(A \otimes B) \multimap C$ *looks the same* as the game $A \multimap (B \multimap C)$ when unfolded: the first move is in C, with subsequent moves in pairs in A, B or C". Our graphical foundation allows such an intuitive argument to be made.



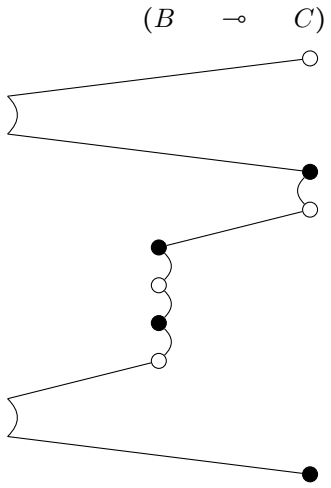
(b) Restricting to the A- and B-nodes yields a $\otimes$-schedule.



(c) Segregating between the B- and C-nodes yields a $\multimap$-schedule.



(d) Segregating between the A- and B- nodes yields a $\multimap$-schedule.



(e) Restricting to the B- and C-nodes yields a $\multimap$-schedule.

Figure 52: (Continued.)

Chapter 4

Pointer diagrams and backtracking

We extend our graphical treatment of interleaving structures in games to account for backtracking pointers. The pointers we describe here allow "nesting" backtracks for O and P. The exponential ! giving O permission to backtrack is the functorial component of a comonad (the *linear exponential comonad* [HS99, HHM07] cf. [Hyl97]). As we will discuss in detail below, the comonad we are concerned with is such that repeated applications of $! : \mathcal{Game} \to \mathcal{Game}$ correspond to O backtracking on multiple nested "timelines". We also give a monad ? providing permission for P to backtrack.

4.1 Pointers and graphs

Following the conventions of standard texts in topology [Arm83, Hat02], we say *map* to mean *continuous function* unless explicitly stated otherwise.

4.1.1 Heaps

A heap is a way of structuring ordered data and allows us to equip $\mathcal{Game}$ with a linear exponential comonad [HHM07, HO00]. The following few definitions are standard.

Definition 122. A **heap** on the set $[n] = \{1, \ldots, n\}$ is a partial function $\phi : [n] \rightharpoonup [n]$ such that $\phi(i)\downarrow \implies \phi(i) < i$. Here we write $\phi(i)\downarrow$ to indicate that $\phi(i)$ is defined, and otherwise may write $\phi(i)\uparrow$. We may write ϕ_n to indicate that ϕ is a heap on $[n]$.

A **parity heap** is one where if $\phi(i)\downarrow$, $\phi(i)$ is the opposite parity of i.

A heap ϕ on $[n]$ may be **restricted** to a heap $\phi\restriction_m$ on $[m]$ for $m \leq n$ where $\phi\restriction_m$ is defined to be ϕ on its domain.

A heap ϕ gives a forest structure with ϕ assigning parents.

Definition 123. Given a heap ϕ on $[n]$, the **ϕ-thread** of $i \in [n]$ is given by the ordered list

$$\left(\phi^j(i)\right)_{j \in \{0,\ldots,n\} \text{ and } \phi^j(i)\downarrow}$$

We may write $\underline{\phi(i)}$ for the ϕ-thread of i.

Example 124. Let ϕ be the heap on $\{1, \ldots, 13\}$ given as in Figure 54(a). Then the ϕ-thread of 9 is $(9, 6, 5, 2, 1)$.

Definition 125. An **O-heap** is a heap ϕ on $[n]$ with the additional properties

(O1) If i is even, then $\phi(i) = i - 1$.

(O2) If i is odd and $\phi(i)\downarrow$, then $\phi(i)$ is even.

Dually, a **P-heap** is a heap ϕ on $[n]$ with the additional properties

(P1) If i is even and $\phi(i)\downarrow$, then $\phi(i)$ is odd.

(P2) If $i > 1$ is odd, then $\phi(i) = i - 1$.

O-heaps and P-heaps are both parity heaps, and are so called as they correspond to O's and P's permissions to backtrack respectively.

Remark 126. The *predecessor* function $\pi : i \mapsto i - 1$ is both an O-heap and a P-heap. In fact it is the only such heap [HHM07].

Definition 127. While a heap ϕ is defined on a set $[n]$, we may attach a **heap structure** to any finite ordered set $V_n = \{v_1, \ldots, v_n\}$ with the bijection $i \leftrightarrow v_i$.

We may abuse notation and also refer to the heap structure as ϕ, writing $\phi(v_i)$ to mean $v_{\phi(i)}$ where to do so is not ambiguous.

If ϕ is an O-heap or a P-heap, we may speak of an **O-heap structure** or a **P-heap structure** respectively.

4.1.2 Heap graphs

Heaps and heap structures are often denoted diagrammatically [CH10, DH01, HO00] (see, for example, Figures 9 and 10). A heap is a labelled directed forest and as such has a natural planar structure. However, it is common practice to encode the ordering on the nodes with their position in a diagram, rather than with their labels. This frequently creates diagrams which do not exhibit the planarity of the heap structure but do use the geometry of the plane to effect. It is these more general diagrams which we wish to capture here, as they encode ordering the the same way that schedule diagrams do.

We first consider a characterisation of the underlying progressive graph [JS91] and then a characterisation of those of its images in the plane corresponding to the diagrams most used in the literature.

Definition 128. A heap ϕ on $[n]$ corresponds to a progressive graph $\Phi = (G, G_0)$ — called a **heap graph** — such that the following hold:

(i) $|G_0| = n$ and G_0 has an explicit ordering $G_0 = \{g_1, \ldots, g_n\}$. This gives us a natural heap structure ϕ on G_0.

(ii) $G_0 \supseteq \partial G$ (no *outer nodes*).

(iii) There is an edge with g_i as source and g_j as target exactly when $\phi(g_i) = g_j$.

Remark 129. Observe that these conditions, in particular (iii), guarantee that each node has at most one outward edge and that G a disjoint union of contractible components. G has no loops or cycles and is compact.

Figures 54(a) and 54(b) show examples of heap graphs.

From the definition of heap graph, we easily deduce:

Proposition 130. *For a heap ϕ on $[n]$, a point i is in the ϕ-thread of j if and only if g_i is reachable from g_j by a directed path in a heap graph $\Phi = (G, G_0)$ of ϕ.*

Definition 131. If ϕ is a heap on $[n]$ with a heap graph $\Phi = (G, G_0)$, then for $i \leq n$, the heap graph $\Phi \restriction_i$ of the restriction $\phi \restriction_i$ can be found by removing the points $\{g_{i+1}, \ldots, g_n\}$ from G_0 and all edges adjacent to a point removed.

To characterise diagrams in the plane representing heap graphs but allowing crossing edges, we may not directly use a notion of *planar embedding* [JS91]. Instead we consider, following the examples set by knot diagrams [Cro04, Rol76], a map which is the composite of a progressive embedding in $\mathbb{R}^3$ followed by a projection to the plane allowing only finitely many transversal crossing double points.

Definition 132. Given a progressive graph $\Gamma = (G, G_0)$, an **upwardly progressive embedding of Γ in $\mathbb{R}^3$** is given by an injective map $\iota : \hat{\Gamma} \hookrightarrow \mathbb{R}^3$ such that:

(i) ι respects the direction on edges: the source of each edge is lower than its target.

(ii) The second projection

$$\pi_2 : \mathbb{R}^3 \to \mathbb{R} \qquad \pi_2 : (x_1, x_2, x_3) \mapsto x_2$$

is injective on each edge.

Here, π_2 should not be confused with the predecessor functions π mentioned in Definition 63.

Similarly, a **downwardly progressive embedding of Γ in $\mathbb{R}^3$** is as an upwardly progressive embedding, but with "lower" replaced by "higher".

Example 133. Figure 53 shows an example of a downwardly progressive embedding of a progressive graph in $\mathbb{R}^3$ with the images of projections onto two vertical planes.

Definition 134. A map $f : \Gamma \to \mathbb{R}^2$ is a **progressive** map of progressive graph $\Gamma = (G, G_0)$ to the plane if each of the following conditions hold:

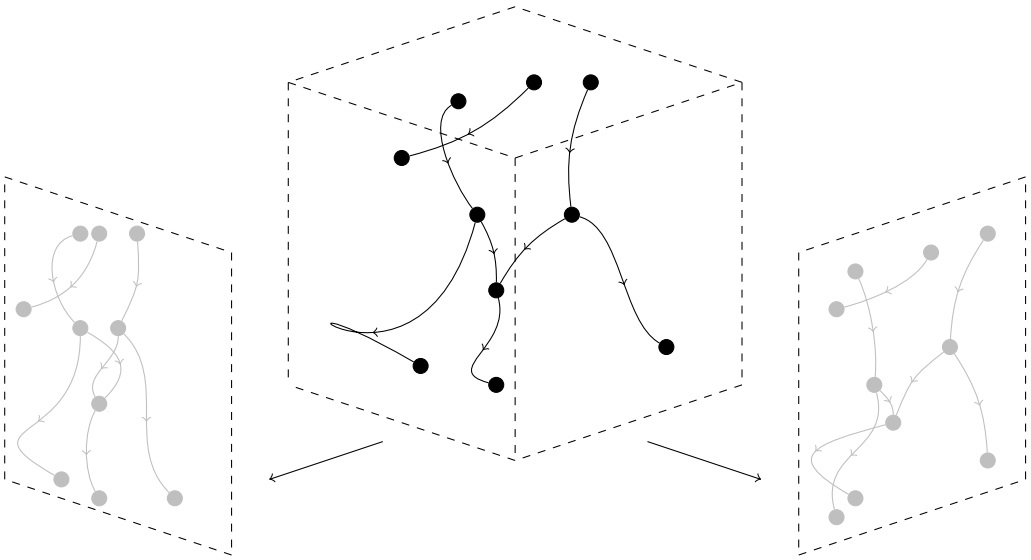


Figure 53: A downwardly progressive embedding of a progressive graph in $\mathbb{R}^3$.

(i) f factors as $f = \varpi \circ \iota$, with ι a progressive embedding of Γ in $\mathbb{R}^3$ and ϖ one of the projections $\varpi : (x_1, x_2, x_3) \mapsto (x_1, x_2)$ or $\varpi : (x_1, x_2, x_3) \mapsto (x_2, x_3)$ a shadow of $\iota(\Gamma)$ in one of the vertical axis planes

(ii) $|f(G_0)| = |G_0|$ and $|f(\partial\Gamma)| = |\partial\Gamma|$ (the images of all nodes and endpoints are distinct)

(iii) There are finitely many **singular points**, points $x \in G$ where $|f^{-1}(x)| > 1$ (using the terminology of [Cro04], p. 52). Each singular point is a **double point**, $|f^{-1}(x)| = 2$. At each of these double points is a **transverse** intersection; no tangencies.

We may call a progressive map either **upwardly** or **downwardly** progressive based on the nature of ι.

Example 135. Figure 53 shows two progressive images of a progressive graph in the plane, each the shadow in a vertical plane of a progressive embedding of the progressive map in $\mathbb{R}^3$.

Remark 136. The notion of progressive map generalises the notion of *progressive embedding* in the sense that a progressive embedding is a progressive map where no points are identified.

We will tend to speak of "a heap graph Φ in the plane" to indicate that Φ comes with an upwardly progressive map of Φ to the plane.

Example 137. Figure 54(a) shows an example of a heap graph for an O-heap ϕ.

Note that all even-numbered nodes are connected directly to their antecedent nodes, and all odd-numbered nodes, if they are connected to anything, are connected to even-numbered nodes.

The ϕ-thread of 12 in this example is $(12, 11, 6, 5, 2, 1)$.

Figure 54(a) shows the image of a progressive embedding of a heap graph for ϕ. Figure 54(b) shows the image of a progressive map of a heap graph for ϕ which is not an embedding.

Remark 138. We will tend to consider the image of heap graphs $\Phi = (G, G_0)$ in the plane under upwardly progressive maps, so that the images of the nodes lie in a vertical line $L_u = \{u\} \times \mathbb{R}$ for some $u \in \mathbb{R}$ and with g_i above g_j exactly when $i < j$. We will refer to this as **standard configuration**. For example, see Figure 54(b); some edges cross, but all nodes are clearly marked and the source and target of each edge is unambiguous.

Graphs in standard configuration may have their heap structure recovered from the vertical orders of the nodes, and as such do not require decorations on the nodes to record this. We will not always number the nodes of graphs in standard configuration.

We will frequently draw heap graphs in standard configuration, allowing edges to cross (so that they are images of progressive maps which are not embeddings).

See Appendix A.4.1 for more discussion of standard configuration.

Remark 139. We will tend to take a heap graph to come equipped with an upwardly progressive map into the plane. We will also tend to take progressive heap graphs' images in the plane to be in standard configuration.

Definition 140. Let $\Phi = (G, G_0)$ be a heap graph together with a progressive map $f = \varpi \circ \iota$ to the plane. Let $\Phi' = (G', G_0')$ be another heap graph with progressive map $f' = \varpi \circ \iota'$ to the plane. We say that $f(\Phi)$ is **deformable into $f'(\Phi')$ as a heap graph** if $\Phi \cong \Phi'$ and there is a continuous function $h : \hat{\Phi} \times [0, 1] \to \mathbb{R}^2$ such that:

- For each $t \in [0, 1]$, $h(-, t)$ is a progressive embedding of Φ in $\mathbb{R}^3$
- $(\varpi \circ h)(\hat{\Phi}, 0) = (\varpi \circ \iota)(\hat{\Phi})$ is a progressive image of Φ in the plane
- $(\varpi \circ h)(\Phi, 1) = (\varpi \circ \iota')(\hat{\Phi'})$ is a progressive image of Φ (and also of Φ') in the plane

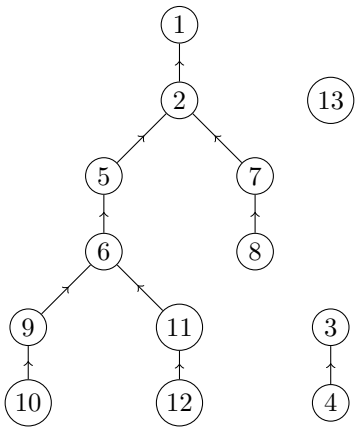
Then we also say that h is a **deformation** and may write $\Phi \sim \Phi'$.

Example 141. Figures 54(a) and 54(b) show two progressive images of a heap graph in the plane which are deformable into each other.

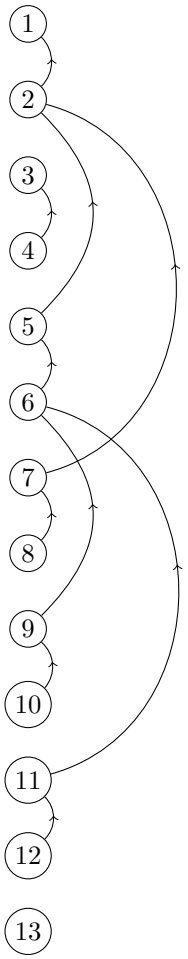
4.1.3 A partial order on heaps

Definition 142. [[HHM07], p. 5.] For heaps ϕ and ψ on $[n]$, we write $\phi \succcurlyeq \psi$ when, for each $k \in [n]$,

$$\psi(k) \in \underline{\phi(k)} \tag{4.1}$$



(a) An O-heap.



(b) The same O-heap. In this arrangement, the ordering of the nodes is recorded by their vertical position so the labels are superfluous.

Figure 54: Two O-heap graphs for the same O-heap.

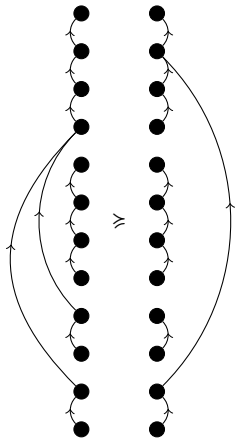


Figure 55: Two O-heaps on $\{1, \ldots, 12\}$. The heap with edges on the left of the nodes is $\succcurlyeq$ the heap with edges on the right of the nodes.

Note that (4.1) is equivalent to $\underline{\psi(k)}$ being a subsequence of $\underline{\phi(k)}$.

We intend to deal with heaps in terms of their graphs, and hence we exhibit this relation on heaps.

Definition 143. Let Φ and Ψ be heap graphs on $[n]$. We will refer to the outward edge from the $k \in [n]$ of the graph Φ as the k-edge of Φ, and likewise for Ψ.

We write $\Phi \succcurlyeq \Psi$ whenever the k-edge of Ψ is either missing, or points somewhere in the path above k in Φ.

Furthermore, we will write $\Phi \dot{\succcurlyeq} \Psi$ when Φ and Ψ differ by no more than one edge.

The preceding an obvious rephrasing of Definition 142 in terms of graphs, so that $\Phi \succcurlyeq \Psi$ if and only if $\phi \succcurlyeq \psi$.

Proposition 144. *Let Φ and Ψ be heap graphs on $[n]$. $\Phi \succcurlyeq \Psi$ if an only if there is a sequence*

$$\Phi = \Phi_0 \dot{\succcurlyeq} \Phi_1 \dot{\succcurlyeq} \cdots \dot{\succcurlyeq} \Phi_n = \Psi$$

Proof. If such a sequence exists, then $\Phi \succcurlyeq \Psi$, since $\succcurlyeq$ is transitive. For the converse, a sequence may be constructed by iterating this step:

- On Φ_i, replace the $(n-i)$-edge with the $(n-i)$-edge of Ψ.

(This terminates as there are finitely many nodes.) □

Example 145. Figure 55 shows two heaps on $\{1, \ldots, 12\}$ with one heap $\succcurlyeq$ the other.

Lemma 146. *$\succcurlyeq$ is a partial order, and with respect to $\succcurlyeq$, π is maximal and 0 (the heap with no edges) is minimal. ("0" should not be confused with the "O" of "O-heap"!)*

Proof. This follows from the characterisation of $\succcurlyeq$ in terms of graphs. □

4.1.4 Heap constructions

Since the underlying heap of a heap graph may be easily recovered, we will extend many of the heap constructions of [HHM07] to constructions on heap graphs. Unless otherwise stated, we will assume a correspondence between uppercase and lower-case Greek letters: we will tend to refer to a heap graph Φ_n with the tacit understanding that it is a graph of a heap ϕ_n.

Definition 147 ([HHM07], p. 6.)**.** Suppose that $S : p \to q$ is a schedule, ϕ an O-heap on q and ψ a P-heap on p. S gives injections $l : [p] \to [p+q]$ and $r : [q] \to [p+q]$. We define the O-heap (ϕ, S, ψ) on $p+q$ by setting

$$(\phi, S, \psi)(k) = \begin{cases} r(\phi(j)) & \text{if } k = r(j) \text{ is odd} \\ l(\psi(i)) & \text{if } k = l(i) \text{ is odd} \\ k-1 & \text{otherwise} \end{cases}$$

We are working from a graphical foundation for schedules and a graphical representation of heaps, so we extend this definition to the graphical forms.

Definition 148. Given a $\multimap\circ$-schedule $S : m \to n$, an O-heap graph Φ_n and a P-heap graph Ψ_m, we may construct an O-heap graph $[\Psi, S, \Phi]$ on $[m+n]$ in the following way:

(T1) $S : U_m \to V_n$ comes with a progressive embedding in the plane with nodes coloured as in Remark 52. Reverse the direction on all the edges.

(T2) Take progressive maps of Φ_n and Ψ_m into the plane so that the images are in standard configuration and so that the image of the nodes of Ψ_m coincide with the image of U_m and the images of the nodes of Φ_n coincide with the image of V_n, and the images of the edges of Φ_n and Ψ_m don't intersect with S.

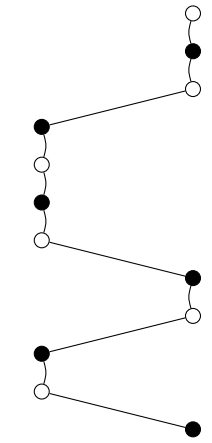
In this way the underlying heap ϕ_n of Φ_n gives a heap structure on the ordered set V_n, and likewise with ψ_m on U_m.

(T3) The O-heap graph $[\Psi, S, \Phi]$ has nodes $U_m + V_n$ with ordering given by the path of S. The following edges are taken for $[\Psi, S, \Phi]$:

- For a $\bullet$ node, take the outward edge in S.
- For a $\circ$ node in U_m, take the outward edge in Ψ.
- For a $\circ$ node in V_n, take the outward edge in Φ.

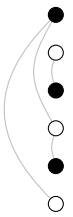
It is immediate from this definition that the underlying heap of $[\Psi, S, \Phi]$ is (ϕ, S, ψ).

Remark 149. As the reader will likely have noticed, we have chosen to write the components of this construction on graphs in the opposite order to that of [HHM07] on heaps. We have chosen to do this so that the symbolic denotation of the construction more closely resemble its graphical counterpart, with, for example the P-heap Ψ being *on the left* of the schedule S. We hope this will not cause confusion.

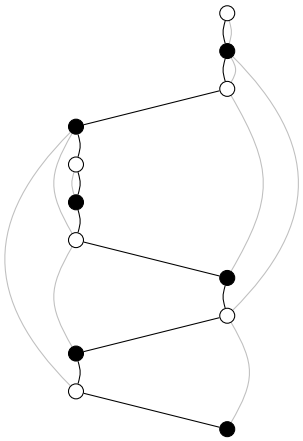


(a) The schedule $S : 6 \to 6$.

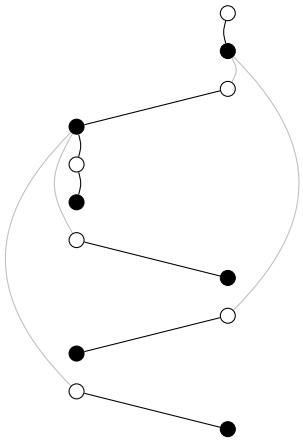
(b) The O-heap graph Φ on $\{1, \ldots, 6\}$.



(c) The P-heap graph Ψ on $\{1, \ldots, 6\}$.



(d) S, Φ and Ψ drawn together.



(e) The O-heap $[\Psi, S, \Phi]$.

Figure 56: The construction $[\Psi, S, \Phi]$.

Remark 150. The alternating natures of $-\circ$-schedules, O-heaps and P-heaps means that the edges we take in (T3) are exactly those which are not required to exist by the definitions of the components.

Definition 151. Let Φ be a heap graph with set of nodes G_0. Then the underlying heap ϕ acts as a heap structure on G_0. For any $X \subseteq G_0$, the heap structure on X achieved by removing from Φ all nodes corresponding to $G_0 \setminus X$ and all attached edges is the **restriction** of Φ to X, written $\Phi \restriction_X$. The underlying heap of $\Phi \restriction_X$ is written $\phi \restriction_X$.

Example 152. Let $S : 6 \to 6$ be the $-\circ$-schedule in Figure 56(a), Φ be the O-heap graph shown in Figure 56(b) and Φ be the P-heap graph shown in Figure 56(c).

We construct the heap graph $[\Psi, S, \Phi]$ by step (T2) (shown in Figure 56(d)), step (T3) (shown in Figure 56(e)).

Consideration of the colours of nodes in the construction of $[\Psi, S, \Phi]$ and the characterisation of an O-heap in terms of colours of nodes yields:

Proposition 153. *The construction of the above definition yields an O-heap graph.*

Definition 154 ([HHM07], p. 6.)**.** Given a schedule $S : p \to q$ and an O-heap ϕ on q, we define an O-heap $S^*\phi$ on p. Set $S^*\phi(k) = j$ just when j is maximal with $j < k$ and $l(j)$ is in the (ϕ, S, π)-thread of $l(k)$ if such exists, and undefined otherwise.

We translate this * into an operation on graphs.

Definition 155. Given a $\multimap$-schedule $S : U_m \to V_n$ and an O-heap graph Φ with nodes V_n, we may construct an O-heap graph $S^*\Phi$ with U_m as follows:

Consider $[\Pi, S, \Phi]$ (where Π is a heap graph of the maximal heap π), which has nodes $U_m + V_n$. $S^*\Phi$ has nodes U_m, and has an edge $u_i \to u_j$ if u_j is the first node reached from u_i by a path in $[\Pi, S, \Phi]$.

It is immediate from the definition that the underlying heap of $S^*\Phi$ is $S^*\phi$.

Remark 156. In many cases we can simplify this definition to the following operation on graphs:

Take the heap graph $[\Pi, S, \Phi]$, declassify all nodes of V_n and then remove all edges which have fewer than two endpoints (also remove any endpoints they alone have).

However, in some cases the node we remove will not be 2-valent and so the result will not be treelike, for example consider the fragment in Figure 57(a). In cases such as this we must make choices of new edges, such as in Figure 57(b).

Unless to do so would be ambiguous, however, we may draw figures which resemble Figure 57(a), which should be taken to indicate a distinct pair of edges. For example, Figure 60(c).

Proposition 157. *The construction in Definition 155 gives a graph of an O-heap.*

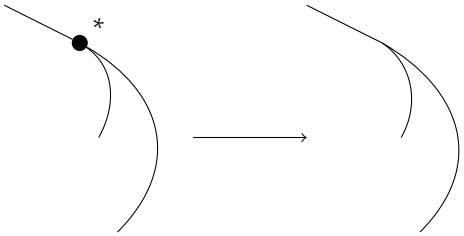
Proof. Consider $S^*\Phi$ as a graph with nodes U_m, with U_m coloured as in the standard colouring of S; i.e., if i is odd u_i is $\bullet$ and if i is even u_i is $\circ$.

If i is even, $[\Pi, S, \Phi]$ has an edge $u_i \to u_{i-1}$, so $S^*\Phi$ has a similar edge, so $(S^*\phi)(i) = i - 1$.

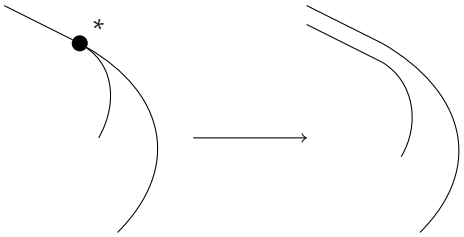
If i is odd, $[\Pi, S, \Phi]$ the outward edge from u_i is an edge of S. If this edge is the first edge in a path which eventually leads back to U_m, this path will necessarily end with an edge from S. All such edges from S have a $\circ$ node as a target, so $(S^*\phi)(i)$ is even. $\square$

Example 158. Let Π, S, Φ be as in Figure 58(a) so that $[\Pi, S, \Phi]$ is as in Figure 58(b). Then $S^*\Phi$ is as in Figure 58(c).

Definition 159 ([HHM07], p. 6.)**.** Given a $\multimap$-scheduling function $S : p \to q$ and a P-heap ϕ on p, we define a P-heap $S_*\phi$ on q. Set $S_*\phi(k) = j$ just when j is maximal with $j < k$ and $r(j)$ in the (π, S, ϕ)-thread of $r(k)$, if such exists, and undefined otherwise.

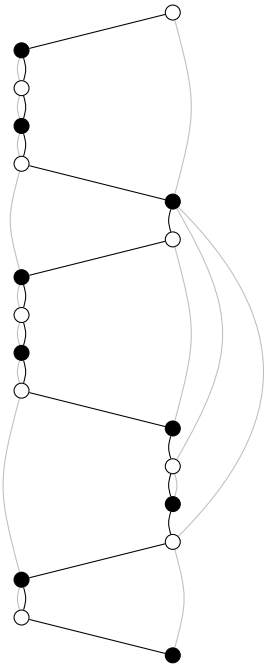


(a) Removing (declassifying) this node does not produce a graph which is a heap graph.

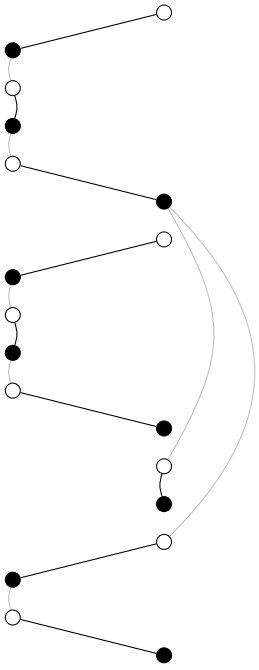


(b) Instead we must draw two new edges.

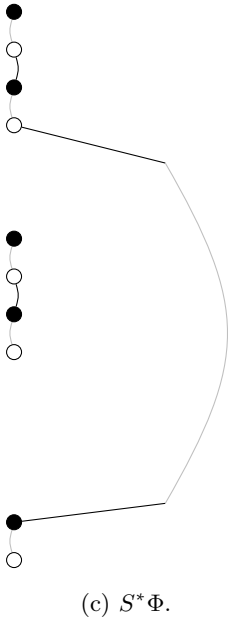
Figure 57: A case where the simplified version of Definition 155 does not work.



(a) Π, S, Φ as in the construction of $[\Pi, S, \Phi]$.



(b) $[\Pi, S, \Phi]$.



(c) $S^*\Phi$.

Figure 58: The construction of $S^*\Phi$.

We translate this $_*$ into an operation on graphs.

Definition 160. Let $S_{m,n} : U_m \to V_n$ be a $\multimap$-schedule and Ψ_m be a P-heap graph with nodes U_m. We construct P-heap graph $S_*\Psi$ with nodes V_n as follows:

$S_*\Psi$ has nodes V_n, and has an edge $v_i \to v_j$ (for $j < i$) when v_j is the first node reached from v_i by a path in $[\Psi, S, \Pi]$.

As before with S^* the underlying heap of $S_*\Psi$ is $S_*\psi$. A similar argument to Proposition 157 proves:

Proposition 161. *The construction of Definition 160 yields a P-heap graph.*

Example 162. Let Ψ, S, Π be as in Figure 59(a). Then $S_*\Psi$ is as in Figure 59(c).

4.2 Categories of heaps

We are now in a position to examine some of the categorical structures in heaps.

4.2.1 Heap functors Ohp and Php

Lemma 163. *Let $S : p \to q$ be a $\multimap$-schedule and let Π be the maximal heap graph on $[q]$. Then $S^*\Pi = \Pi$.*

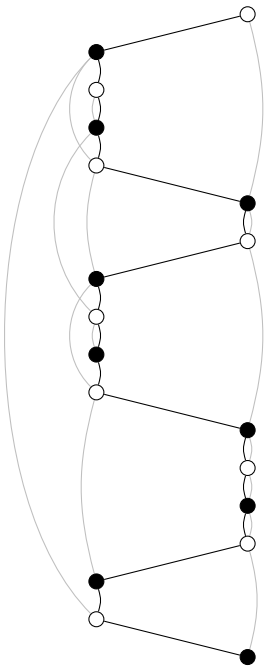
Proof. To avoid confusion, we will write $\Pi^{[n]}$ to mean the maximal heap Π on $[n]$. Draw $S : U_p \to V_q$ in the plane with nodes coloured as in Remark 52. Draw the graphs of $\Pi^{[p]}$ and $\Pi^{[q]}$ with nodes on U_p and V_q respectively. In the graph of the heap $\left[\Pi^{[p]}, S, \Pi^{[q]}\right]$, there is an edge from each $\bullet$ node to the previous (in the path ordering of S) node, which is $\circ$, as the edge is taken from S. From a $\circ$ node in U_p, the outward edge is taken from $\Pi^{[p]}$ and from a $\circ$ node in V_q, the outward edge is taken from $\Pi^{[q]}$.

Thus, the graph of $\left[\Pi^{[p]}, S, \Pi^{[q]}\right]$ is equal to the graph of $\Pi^{[p+q]}$ (up to deformation) and so the graph of $S^*\Pi^{[q]}$ is equal to the graph of $\Pi^{[p]}$ (up to deformation). □

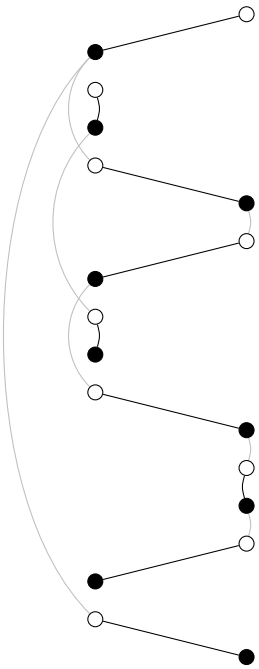
Proposition 164. *If $\Phi_n \succcurlyeq \Psi_n$ are O-heaps on $[n]$ and $S : U_m \to V_n$ is a $\multimap$-schedule, then $[\Pi, S, \Phi] \succcurlyeq [\Pi, S, \Psi]$ as heaps on $[m+n]$.*

Proof. The edges of $[\Pi, S, \Phi]$ are taken from those of Φ, S and Π as described in Definition 148. We consider how the graph of Ψ is achieved from the graph of Φ (the sequence given by Proposition 144), and how this affects the edges chosen for the graph of $[\Pi, S, \Psi]$.

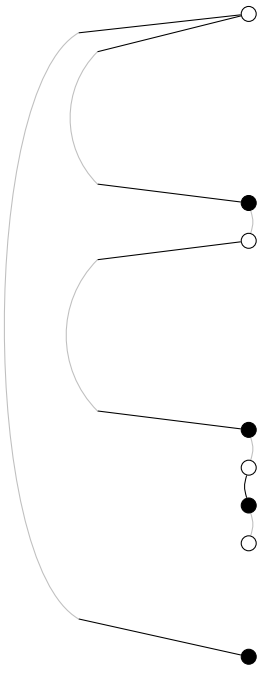
The edges from $\bullet$ nodes of $[\Pi, S, \Phi]$ are taken from S and thus changes to Φ will have no effect. Similarly edges from $\circ$ nodes of $[\Pi, S, \Phi]$ which are in U_m are taken from Π and so changes to Φ will have no effect. So the only edges which may change are those from $\circ$ nodes in V_n which are taken from Φ.



(a) Ψ, S, Π as in the construction of $[\Psi, S, \Pi]$.



(b) $[\Psi, S, \Pi]$.



(c) $S_* \Psi$.

Figure 59: The construction of $S_* \Psi$.

Suppose that $\Psi \dot{\preccurlyeq} \Phi$. So that an edge $v_i \to v_j$ is either missing in Ψ, or has been replaced by an edge $v_i \to v_k$ where there is a path $v_j \to v_k$ in Φ. Since Φ and Ψ are both O-heaps by assumption, the original edge must be from a $\circ$ node, and hence was an edge in $[\Pi, S, \Phi]$, and the new edge will be taken in $[\Pi, S, \Psi]$. If there is a path $v_j \to v_k$ in Φ then there is a path $v_j \to v_k$ in $[\Pi, S, \Phi]$ since each $\bullet$ node in V_n is connected to the previous $\circ$ node in V_n by edges in S and Π.

The result therefore follows from Proposition 144. □

Corollary 165. *For heaps $\Phi_n \succcurlyeq \Psi_n$ and a $\multimap$-schedule $S : m \to n$, $S^*\Phi \succcurlyeq S^*\Psi$ as heaps on $[m]$.*

Example 166. Figure 60 shows an example of heaps $\Phi \succcurlyeq \Psi$ and the explicit construction of $S^*\Phi \succcurlyeq S^*\Psi$ for a schedule S.

Lemma 163 and Corollary 165 together give:

Corollary 167. *S^* is a map of posets which preserves the top element.*

Corollary 168. *S_* is a map of posets which preserves the top element.*

Lemma 169. *For copycat $\multimap$-schedule $I : n \to n$ and O-heap Φ_n, we have $I^*\Phi = \Phi$.*

Proof. We require, for I the copycat schedule $U_n \to V_n$, that $I^*\Phi = \Phi$ up to deformation.

In Φ (with nodes V_n) there is an edge from each $\bullet$ node to the previous $\circ$ node; this is condition (O1) in Definition 125. Similarly, in $[\Pi, I, \Phi]$ each $\circ$ node in U_n has an edge to the previous $\bullet$ node (the edge taken from Π).

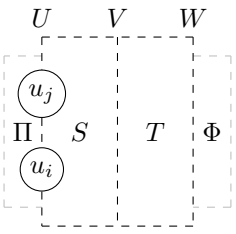
From each $\bullet$ node u_i in U_n, the outward edge in $[\Pi, I, \Phi]$ is the edge from I, and so has v_i ($\circ$) as its target. The outward edge from v_i in Φ leads to some $\bullet$ node v_j, and the outward edge from this is an edge $v_j \to u_j$ from I. Hence there is a path $u_i \to u_j$ in $[\Pi, I, \Phi]$.

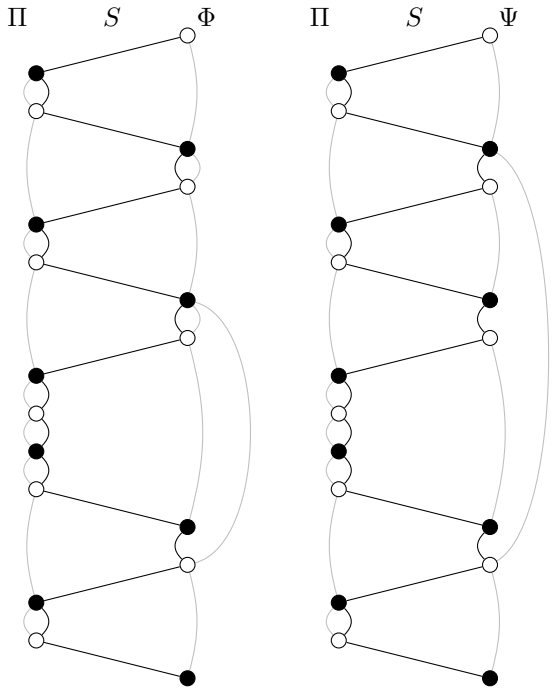
Thus there is an edge $u_i \to u_j$ in $I^*\Phi$ if and only if there is an edge $v_i \to v_j$ in Φ. □

Lemma 170. *For $\multimap$-schedules $S : U_m \to V_n$ and $T : V_n \to W_r$ and O-heap Φ on $[r]$,*

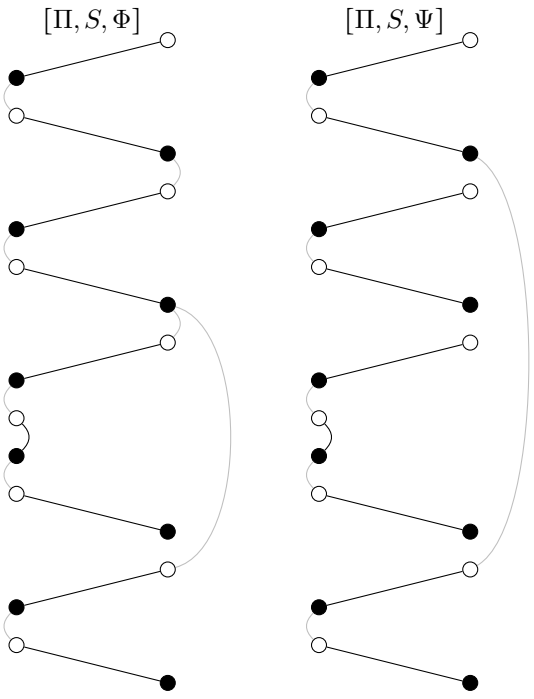
$$(S \| T)^*\Phi = S^*(T^*\Phi) \tag{4.2}$$

Proof. In the calculation of $(S\|T)^*\Phi$, we use a composition diagram



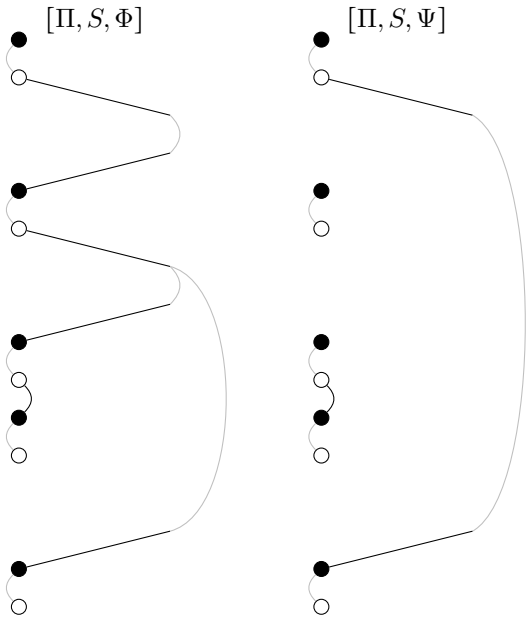


(a) Π, S and $\Phi \gtrsim \Psi$.



(b) $[\Pi, S, \Phi] \gtrsim [\Pi, S, \Psi]$.

Figure 60: An explicit construction of $S^*\Phi \gtrsim S^*\Psi$.

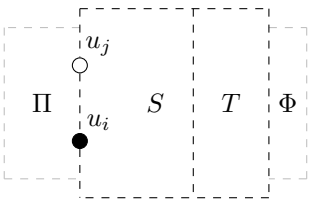


(c) $S^*\Phi \gtrsim S^*\Psi$.

Figure 60: (Continued.)

with nodes coloured as in Remark 52. We pick $u_i, u_j \in U$ with $j < i$ (as shown) and with i and j having opposite parity.

Suppose there's an edge $u_i \to u_j$ in $(S\|T)^*\Phi$. This must be because there is a path $u_i \to u_j$ in $[\Pi, S\|T, \Phi]$. If u_i is $\circ$ then this path is by an edge in Π, which would also lead to edge $u_i \to u_j$ in $S^*(T^*\Phi)$. So let's suppose that u_i is $\bullet$ (and u_j is $\circ$).



Now there is an edge $u_i \to u_j$ in $(S\|T)^*\Phi$ if and only if there is a path $u_i \to u_j$ in $[\Pi, S\|T, \Phi]$ passing through no other nodes in U. This can happen if and only if exactly one of the following three cases occurs:

1. The path $u_i \to u_j$ in $[\Pi, S\|T, \Phi]$ only passes through nodes in U, so is just an edge $u_i \to u_j$.
2. The path $u_i \to u_j$ in $[\Pi, S\|T, \Phi]$ passes through nodes in U and in V but not W.
3. The path $u_i \to u_j$ in $[\Pi, S\|T, \Phi]$ passes through nodes in U and V and W.

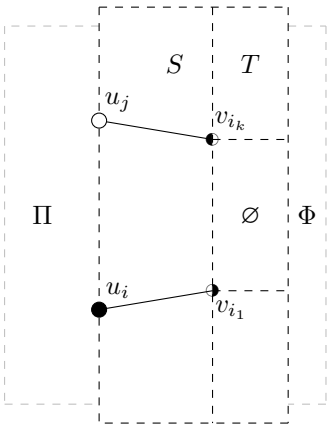
We consider each case separately.

Only U. In this case, the path consists only of an edge $u_j \to u_i$ in S. Therefore, regardless of T and Φ, this edge will be taken in $[\Pi, S, T^*\Phi]$ and so represents a path $u_i \to u_j$ in $S^*(T^*\Phi)$.

U and V. In this case, the path consists of edges of S and edges of T between V-nodes. It is of the form

$$u_i \to v_{i_1} \to \cdots \to v_{i_k} \to u_j$$

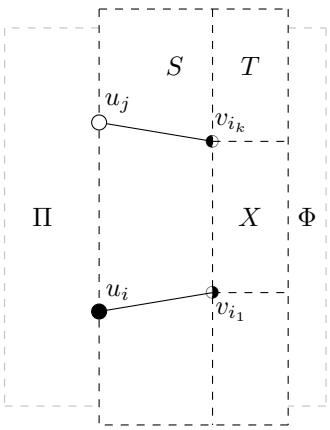
and there are edges between each V-node in S and in T. Equivalently, there will be a path $v_{i_1} \to v_{i_k}$ in $T^*\Phi$.



U and V and W. In this case the path can be broken into components

$$u_i \to X \to u_j$$

where X is a path through only V-nodes and W-nodes. Again, $u_i \to u_j$ is a path in $[\Pi, S \| T, \Phi]$ if and only if X is a path in $T^*\Phi$.



In all cases the path $u_i \to u_j$ is unbroken if and only if there is a path $u_i \to u_j$ in $S^*(T^*\Phi)$. □

Definition 171. Let $\mathrm{Ohp}(n)$ be the set of (deformation classes of) O-heap graphs on $[n]$. Also, given a $\multimap$-schedule $S : m \to n$, let $\mathrm{Ohp}(S)$ be the map of sets S^*.

With Ohp defined as above, Lemmata 169 and 170 together give:

Theorem 172. Ohp *is a functor* $\mathrm{Ohp} : \mathcal{Sched}^{\mathrm{op}} \to \mathcal{Set}$.

Here we are viewing S^* as a map of sets of O-heaps, but by Corollary 165, S^* is also a map of posets preserving the top element, so we may likewise extend the definition of Ohp:

Definition 173. Let $\mathrm{Ohp}_{\succcurlyeq}(n)$ be the set of O-heap graphs on $[n]$, partially ordered by $\succcurlyeq$. Also, given a $\multimap$-schedule $S : m \to n$, let $\mathrm{Ohp}_{\succcurlyeq}(S)$ be the map of posets S^*.

Then Theorem 172 gives us:

Corollary 174. *Writing* $\mathcal{Poset}$ *for the category of partially ordered sets and order-preserving functions,* $\mathrm{Ohp}_{\succcurlyeq}$ *is a functor* $\mathcal{Sched}^{\mathrm{op}} \to \mathcal{Poset}$.

The following proceed along similar lines:

Definition 175. Let $\mathrm{Php}(m)$ be the set of (deformation classes of) P-heap graphs on $[m]$. Also, given a $\multimap$-schedule $S : m \to n$, let $\mathrm{Php}(S)$ be the map of sets S_*.

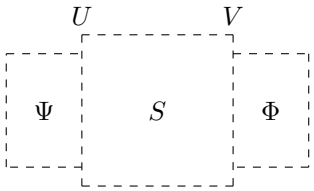
Theorem 176. Php *is a functor* $\mathcal{Sched}^{\mathrm{op}} \to \mathcal{Set}$.

Definition 177. Let $\mathrm{Php}_{\succcurlyeq}(m)$ be the set of P-heap graphs on $[m]$, partially ordered by $\succcurlyeq$. Also, given a $\multimap$-schedule $S : m \to n$, let $\mathrm{Php}_{\succcurlyeq}(S)$ be the map of posets S_*.

Corollary 178. $\mathrm{Php}_{\succcurlyeq}$ *is a functor* $\mathcal{Sched}^{\mathrm{op}} \to \mathcal{Poset}$.

4.2.2 Composing and decomposing threads

We will examine for a moment the nature of individual threads of an O-heap $[\Psi, S, \Phi]$ (for some appropriate graphs Ψ, S, Φ). $[\Psi, S, \Phi]$ is a graph with a natural progressive map to the plane given by its construction diagram:



(Recall that in such a diagram, the edges of S point upwards, reversed from the standard graph of a schedule.) A thread of $[\Psi, S, \Phi]$ is therefore a subgraph of this; a path in the plane with nodes in U and V. It can be extracted from the construction diagram after first colouring the nodes as in Remark 52.

We begin by selecting the node whose $[\Psi, S, \Phi]$-thread is to be found. If it is $\bullet$, we take the upward edge from S; if it is $\circ$, we take the upward edge from Φ or Ψ, continuing in this way until no appropriate outward edge exists.

The path yielded in this way satisfies the following conditions:

- Nodes alternate $\circ$–$\bullet$ along the path, as each edge of S, Φ and Ψ has endpoints of different colours.
- The subset of the nodes of the path which are in U alternate $\circ$–$\bullet$, as do those in V.
- Edges taken from S are $\bullet \to \circ$ by the switching condition of S, and edges taken from Φ or Ψ are $\circ \to \bullet$ by the definitions of O- and P-heaps.

This observation is almost enough to make these threads into $-\circ$-schedules (with reversed edge directions). In fact, the only missing condition is that the uppermost node is in V. Even without this condition, however, we are still able to extend the notion of composition of $-\circ$-schedules to *composition* of threads of this kind. Informally speaking: if we have a thread formed in this way with nodes in $U + V$ and another such thread with nodes in $V + W$, we may deform each thread without moving the nodes so that all edges are in the interior of the vertical strip of the plane whose left and right boundaries pass through the left- and right-hand subsets of the thread's nodes. Then, as with composition of $-\circ$-schedules, we remove an "extended $\}$" from the nodes of V before declassifying them.

This notion is alluded to in [HHM07], and the following proposition is stated, but not proved.

Proposition 179 ([HHM07], Proposition 4.2.)**.** *Let $S : p \to q$ and $T : q \to r$ be $-\circ$-scheduling functions. Let ψ_p be a P-heap on p and ϕ_r be an O-heap on r. Then threads for $(\phi, T.S, \psi)$ on $p + r$ are composites of unique threads for $(\phi, T, S_*\psi)$ on $q + r$ and $(T^*\phi, S, \psi)$ on $p + q$.*

We restate this theorem in graphical terms in order to prove it here.

Proposition 180. *Let $S_{m,n} : U_m \to V_n$ and $T_{n,r} : V_n \to W_r$ be $-\circ$-schedules. Let Ψ_m be a P-heap graph on U_m and Φ_r be an O-heap graph on W_r. Then threads of $[\Psi, S \| T, \Phi]$ are composites of threads of $[\Psi, S, T^*\Phi]$ and $[S_*\Psi, T, \Phi]$.*

Proof. We will illustrate this proof with a running example, with S, T, Φ, Ψ as in Figure 61. We begin by selecting a node of $U + W$ whose thread in $[\Psi, S \| T, \Phi]$ we wish to consider. Assume, without loss of generality, that the node we pick is in W. Figure 62 shows $[\Psi, S \| T, \Phi]$ with one such thread highlighted.

Consider the graphs of $[\Psi, S, T^*\Phi]$ and $[S_*\Psi, T, \Phi]$. (Running example in Figure 63.) Observe that $T^*\Phi$ provides path-adjacency (or "first-reachability") information about V-nodes in $[\Pi, T, \Phi]$, since there is an edge $v_i \to v_j$ in T^*S just when there is a minimal path $v_i \to v_j$ in $[S_*\Psi, T, \Phi]$, since such a path would also exist and be minimal in

$[\Pi, T, \Phi]$, from whence $T^*\Phi$ is constructed. Likewise $S_*\Psi$ provides path-adjacency information about the corresponding V-nodes in $[\Psi, S, \Pi]$. In this sense, $T^*\Phi$ and $S_*\Psi$ emulate what is happening in the other diagram.

With this in mind, we can find a thread of $[\Psi, S, T^*\Phi]$ and a thread of $[S_*\Psi, T, \Phi]$ which compose to form our originally chosen thread of $[\Psi, S\|T, \Phi]$. As the node of $[\Psi, S\|T, \Phi]$ we selected was in W, the corresponding node is in $[S_*\Psi, T, \Phi]$ and has a $[S_*\Psi, T, \Phi]$-thread there. Following this thread until it reaches a V-node, we may select a thread in $[\Psi, S, T^*\Phi]$. (In the running example, the two threads are shown in Figure 64.)

These two threads are bound to be composable, since $T^*\Phi$ and $S_*\Psi$ provide the path-adjacency information which guarantees that an "extended $\}$ shape" can be removed (as described previously). To form the composite, we start at our chosen W-node, and follow the thread, passing between the $[\Psi, S, T^*\Phi]$- and $[S_*\Psi, T, \Phi]$-threads each time we reach a V-node. This is also exactly descriptive of the edges taken in the $[\Psi, S\|T, \Phi]$ of our chosen node. (The composition of the two threads in the running example is shown in Figure 65.) □

4.2.3 *Oheap*, the category of O-heaps

We wish to exhibit a category whose objects are O-heap graphs. The morphisms of this category will require some notion of *maps of O-heap graphs*. This is not given in [HHM07], but an equivalent category could be given in their terms.

Definition 181. A **map of O-heaps** from Ψ_m to Φ_n is given by a $\multimap$-schedule $S : m \to n$ for which $\Psi = S^*\Phi$.

Definition 182. We define *Oheap* to have as objects (deformation-classes of) O-heap graphs Φ. A morphism $\Psi_m \to \Phi_n$ of *Oheap*, where Ψ_m and Φ_n are O-heaps, is a map of O-heaps $S : m \to n$.

The identity morphism on Φ_n is the copycat $\multimap$-schedule $I_{n,n}$. Composition of morphisms is given by composition of $\multimap$-schedules

Proposition 183. *Oheap is a category.*

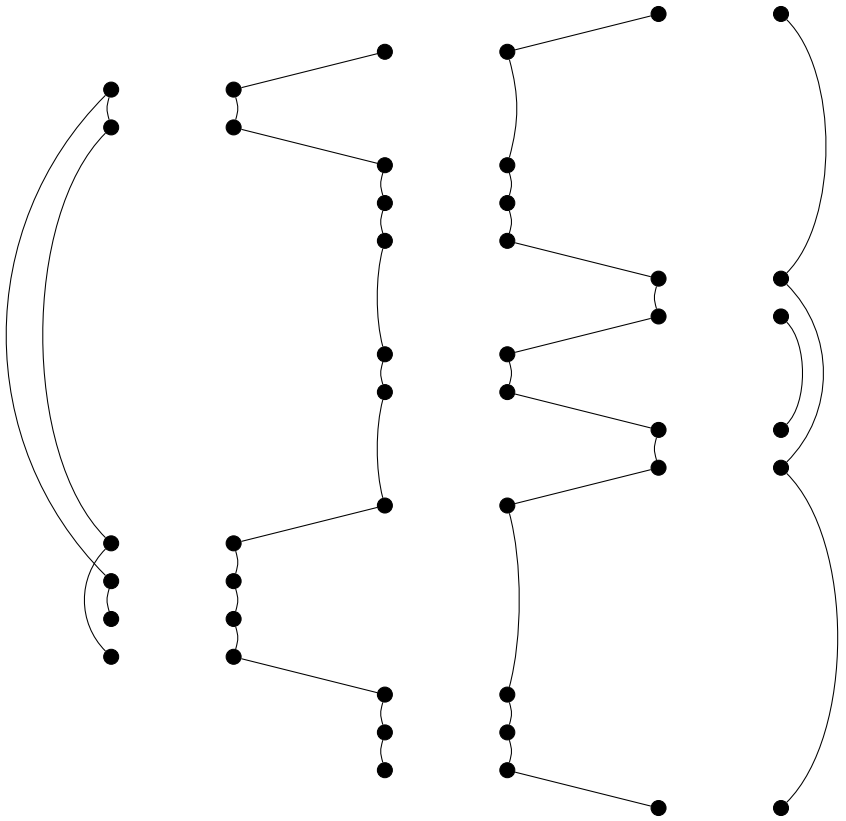
Proof. If we have O-heaps Ψ_m, Φ_n and Θ_r with maps of O-heaps $S : m \to n$ and $T : n \to r$ as in the diagram

$$\Psi_m \xrightarrow{S} \Phi_n \xrightarrow{T} \Theta_r$$

then $\Psi = S^*\Phi$ and $\Phi = T^*\Theta$, so

$$\Psi = S^*(T^*\Theta) = (S\|T)^*\Theta$$

by Lemmata 169 and 170. □



(a) Ψ. (b) S. (c) T. (d) Φ.

Figure 61: The components for the running example in the proof of Proposition 180.

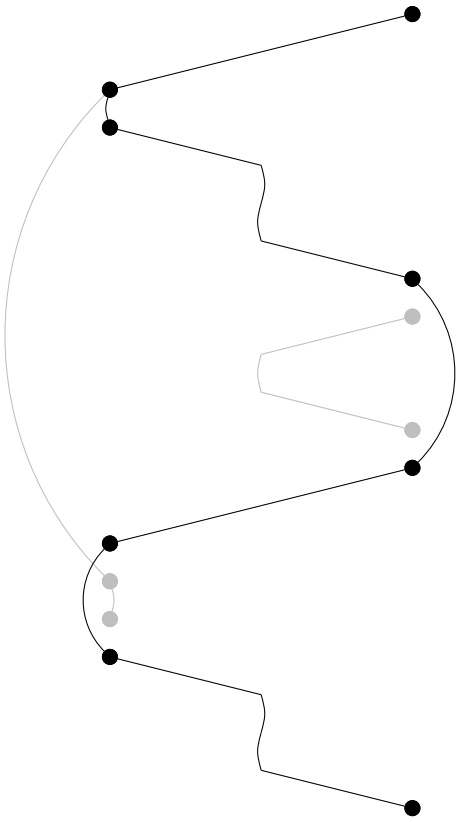
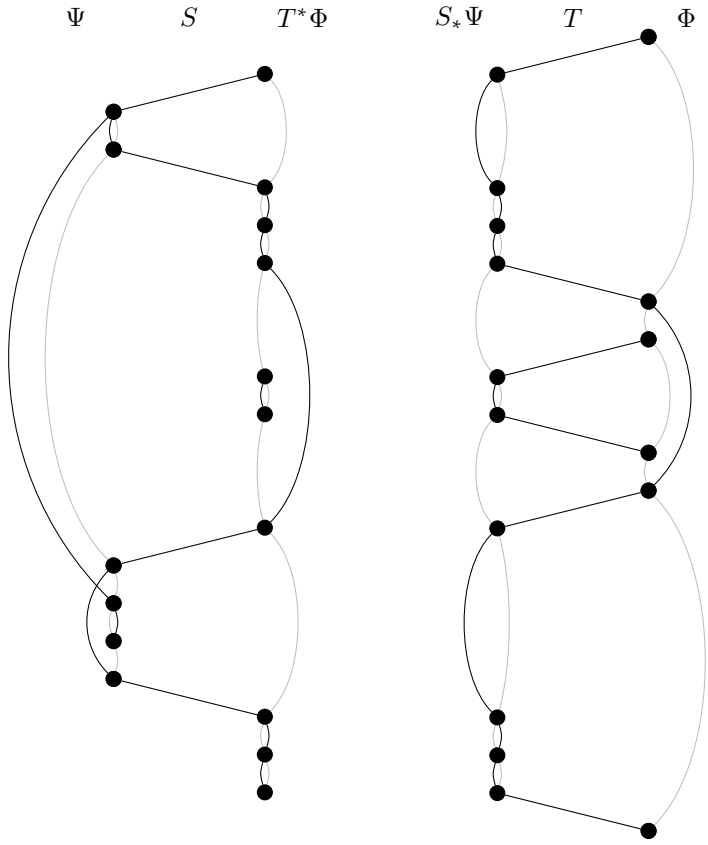


Figure 62: $[\Psi, S \| T, \Phi]$ is shown in grey with one particular thread highlighted in black.



(a) $[\Psi, S, T^*\Phi]$.

(b) $[S_*\Psi, T, \Phi]$.

Figure 63: Two heaps highlighted in black on their construction diagrams.

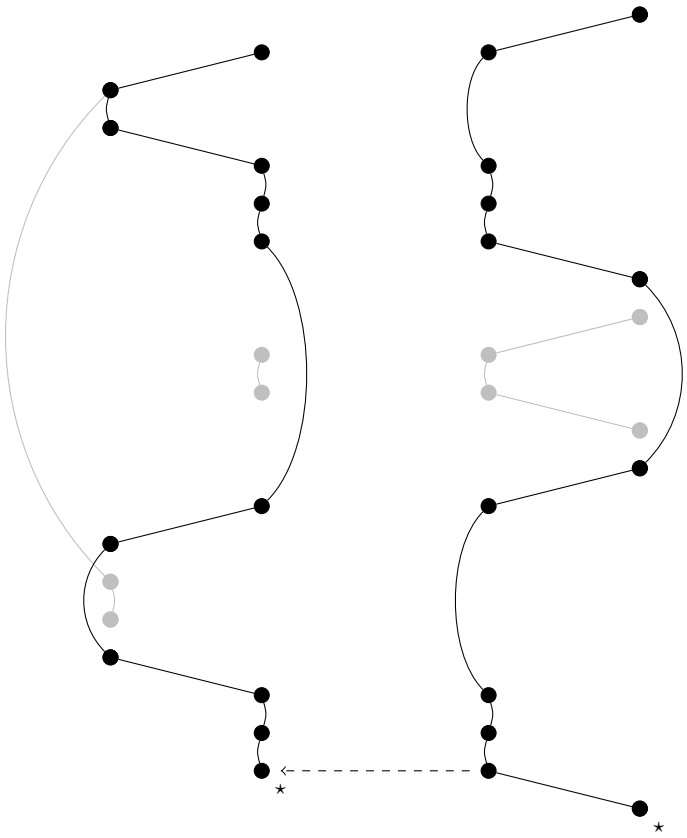


Figure 64: A $[\Psi, S, T^*\Phi]$-thread and a $[S_*\Psi, T, \Phi]$-thread.

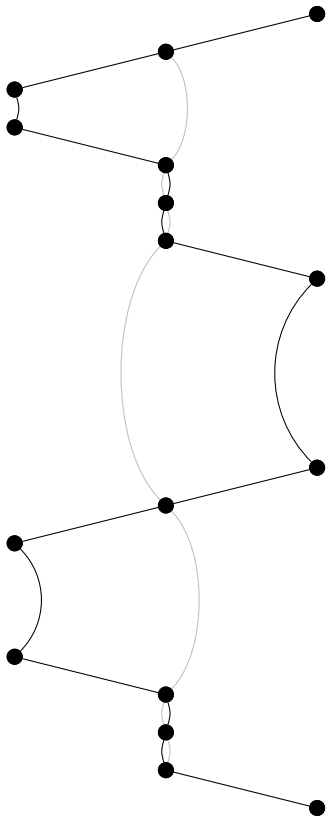


Figure 65: A composition diagram for the $[\Psi, S, T^*\Phi]$-thread and $[S_*\Psi, T, \Phi]$-thread from Figure 64. Notice that the composite (highlighted in black) is the same as the $[\Psi, S\|T, \Phi]$-thread shown in Figure 62.

4.3 Further categorical properties

4.3.1 Aside: discrete fibrations

Fibrations in category theory are a generalised way to express indexed families. An excellent exposition of this with motivation from indexed families of sets can be found in [Pho92]. While some terminology differs from place to place, the following definitions for *cartesian*, *fibration*, *cleavage* and *fibre* are standard.

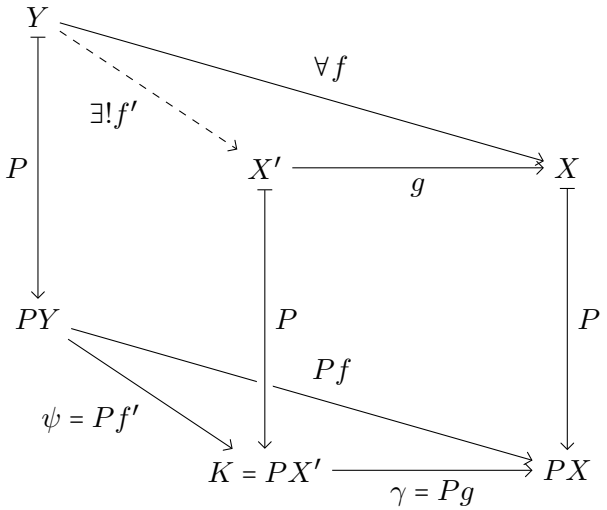
Definition 184 ([BW99], pp. 327–328.)**.** Let $P : \mathscr{E} \to \mathscr{B}$ be a functor between small categories, let X be an object of $\mathscr{E}$ and let $\gamma : K \to PX$ be an arrow in $\mathscr{B}$. Then an arrow $g : X' \to X$ of $\mathscr{E}$ is **cartesian** for γ and X if

(C1) $Pg = \gamma$ (so $K = PX'$).

(C2) For any arrow $f : Y \to X$ of $\mathscr{E}$ and any arrow $\psi : PY \to PX'$ with $Pf = \gamma \circ \psi$, there is a unique arrow $f' : Y \to X'$ in $\mathscr{E}$ so that $f = g \circ f'$ and $Pf' = \psi$.

In this case we call g a **lifting** of γ.

In other words [Pho92], a map $g : X' \to X$ in $\mathscr{E}$ is cartesian if, given any $f : Y \to X$, each factorisation of Pf through Pg uniquely determines a factorisation of f through g



Definition 185 ([Pho92], Definition 2.2.2.)**.** $P : \mathscr{E} \to \mathscr{B}$ is a **fibration** if, for all objects X in $\mathscr{E}$ and arrows $\gamma : K \to PX$ in $\mathscr{B}$, there is an object X' of $\mathscr{E}$ such that $PX' = K$ and there is a cartesian lifting $g : X' \to X$ of γ. $\mathscr{B}$ may be called the **base** category.

Definition 186 ([Shu08], Definition 3.1.)**.** A **cleavage** for a fibration $P : \mathscr{E} \to \mathscr{B}$ is a choice $\tilde{\gamma}$ of cartesian lifting for each $X \in \mathscr{E}$ and arrow $\gamma : K \to PX$ of $\mathscr{B}$. A cleavage is **normal** if $\widetilde{\mathrm{id}_{PX}} = \mathrm{id}_X$ for each $X \in \mathscr{E}$. A cleavage is **split** (or is a **splitting**) if $\widetilde{g \circ f} = \tilde{g} \circ \tilde{f}$ for all composible g and f in $\mathscr{B}$.

Definition 187. Given a fibration $P : \mathscr{E} \to \mathscr{B}$, the **fibre** over an object $I \in \mathscr{B}$ is the subcategory $P^{-1}(I)$.

If $P : \mathscr{E} \to \mathscr{B}$ is a fibration, and I is an object of $\mathscr{B}$, we can think of the fibre of I to be I-indexed (or I-parameteriesd) families of objects of $\mathscr{E}$. Then an arrow $J \to I$ is lifted to a "reindexing" or "change-of-basis" arrow. Of particular interest is when $\mathscr{B} = \mathcal{Set}$.

Definition 188 ([BW99], p. 334.)**.** A fibration is **discrete** when the fibres over each object of the base category are discrete categories (all arrows are identities).

Definition 189 ([MM92], p. 41.)**.** Let $\mathscr{C}$ be a small category and let $F : \mathscr{C}^{\mathrm{op}} \to \mathcal{Set}$ be a contravarient functor. The **category of elements** for F is denoted $\int_{\mathscr{C}} F$. Its objects are pairs (X, x) where $X \in \mathscr{C}$ and $x \in FX$. An arrow $(X, x) \to (Y, y)$ is an arrow $f : X \to Y$ in $\mathscr{C}$ for which $(Ff)(y) = x$.

Remark 190. The category of elements goes by many names across various sources [MM92, BW99, Shu08], including the *category of elements for a presheaf*, the *category of coelements* and the *Grothendieck construction* (where it is often written $G_0(\mathscr{C}, F)$), though the Grothendieck construction often refers to a more general case where $\mathcal{Set}$ is replaced by some other category.

Given a functor $F : \mathscr{C}^{\mathrm{op}} \to \mathcal{Set}$, there is a projection functor

$$U : \textstyle\int_{\mathscr{C}} F \longrightarrow \mathscr{C}$$

$$(X, x) \longmapsto X$$

$$(f : (X, x) \to (Y, y)) \longmapsto (f : X \to Y)$$

Proposition 191. *Given a functor $F : \mathscr{C}^{\mathrm{op}} \to \mathcal{Set}$, the projection functor $U : \int_C F \to \mathscr{C}$ is a discrete fibration.*

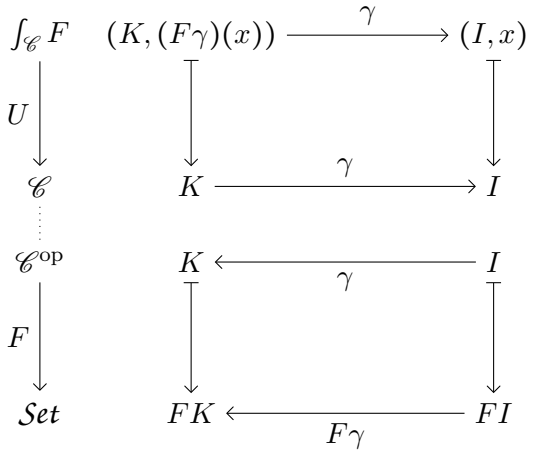
Proof. Chose an object (I, x) of $\int_{\mathscr{C}} F$ and a morphism $\gamma : K \to I = U(I, x)$ of $\mathscr{C}$. We have the following picture:

$$\begin{array}{ccc} \int_{\mathscr{C}} F & & (I,x) \\ {\scriptstyle U}\downarrow & & \downarrow \\ \mathscr{C} & & I \\ & K \xrightarrow{\gamma} & \end{array}$$

We want an object $(I, x)'$ and an arrow $\tilde{\gamma} : (I, x)' \to (I, x)$ such that

- $U((I, x)') = K$.
- $\tilde{\gamma}$ is a cartesian lift of γ.

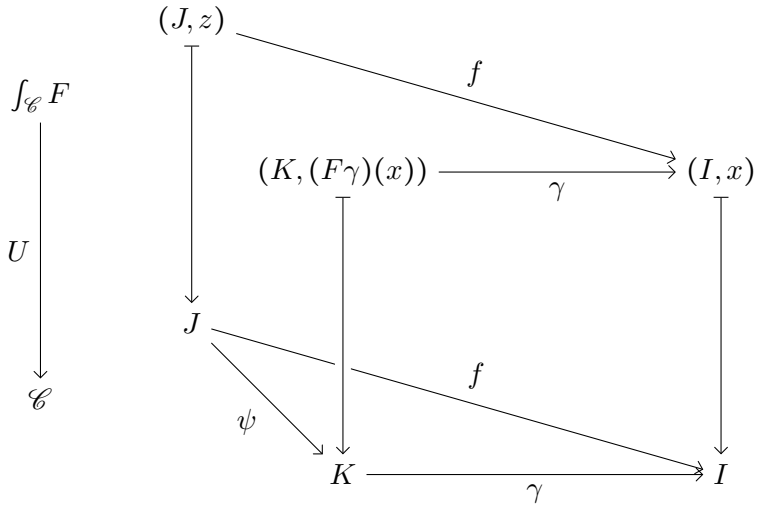
$(I,x)' \in \int_{\mathscr{C}} F$ and we wish it to lie above K, so we may write it $(I,x)' = (K,y)$. For any arrow $g : (K,y) \to (I,x)$ in $\int_{\mathscr{C}} F$ we have that $(Fg)(x) = y$, and if g lies above γ, then it is in fact γ. So we have the following picture:



Now suppose we have another arrow $f : (J,z) \to (I,x)$ of $\int_{\mathscr{C}} F$ such that

$$f = \gamma \circ \psi \quad \text{in } \mathscr{C} \tag{4.3}$$

In pictures:

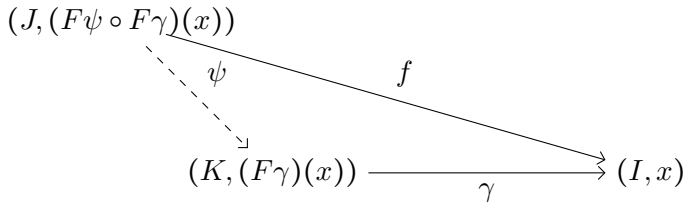


We want some lifted arrow $\tilde{\psi} : (J,z) \to (K,(F\gamma)(x))$ so that $f = \gamma \circ \psi$ in $\int_{\mathscr{C}} F$.

So, by (4.3) and since f is an arrow in $\int_{\mathscr{C}} F$, we have

$$(J,z) = (J,(Ff)(x)) = (J,(F(\gamma \circ \psi))(x)) = (J,(F\psi \circ F\gamma)(x))$$

and so $\psi : (J, (F\psi \circ F\gamma)(x)) \to (K, (F\gamma)(x))$ is the unique arrow in $\int_{\mathscr{C}} F$ making



commute. Therefore U is a fibration.

Notice that the fibres are discrete categories:

$$(U^{-1}I)_0 = \{(I, x) \mid x \in FI\} \cong FI$$

and if $f : (I, x) \to (J, y)$ is such that $Uf = \mathrm{id}$ then $f = \mathrm{id}$. □

4.3.2 O-heaps as discrete fibrations

We may now see how the categories of heaps and schedules we have previously described fit in.

Proposition 192. *Oheap* $\cong \int_{\mathcal{Sched}} \mathrm{Ohp}$.

Proof. By Theorem 172, Ohp is a functor $\mathcal{Sched}^{\mathrm{op}} \to \mathcal{Set}$. Therefore we may construct $\int_{\mathcal{Sched}} \mathrm{Ohp}$. $\int_{\mathcal{Sched}} \mathrm{Ohp}$ has objects which are pairs (Φ, n) with

$$\Phi = \Phi_n \in \mathrm{Ohp}(n) = \{\text{O-heaps on } [n]\}$$

and since n can be recovered from Φ_n this may be considered an object of *Oheap*.

A morphism $(\Psi_m, m) \to (\Phi_n, n)$ in $\int_{\mathcal{Sched}} \mathrm{Ohp}$ is a morphism $m \to n$ of $\mathcal{Sched}$ — i.e. a $\multimap$-schedule $S_{m,n} : m \to n$ — such that

$$\mathrm{Ohp}(S)(\Phi) = S^*\Phi = \Psi$$

which is exactly a morphism $\Psi \to \Phi$ in *Oheap*. □

The projection functor from $\int_{\mathcal{Sched}} \mathrm{Ohp}$ is

$$\begin{aligned} U : \mathit{Oheap} &\longrightarrow \mathcal{Sched} \\ \Phi_n &\longmapsto n \\ S : \Psi \to \Phi &\longmapsto (S : m \to n) \end{aligned} \tag{4.4}$$

Propositions 191 and 192 therefore lets us make the following classification:

Theorem 193. *The functor* (4.4) *is a discrete fibration. The fibre over* $n \in \mathcal{Sched}$ *is the set* $\{$*O-heap graphs* $\Phi_n\}$.

Proposition 192 and Theorem 193 do not buy us anything in what follows in Section 4.4, but seem worth documenting nonetheless. Further investigation into the categorical structures of $\mathcal{Oheap}$ and $\mathcal{Sched}$ would surely be interesting.

4.4 Exponentials

[HHM07] describes a functor ! on their category $\mathscr{G}$ of games and strategies which grants permission for O to "backtrack" to an earlier move and replay it. ! is shown to be a monoidal comonad which satisfies the conditions required for a *linear exponential* comonad (in the terminology of [HS99]).

Together with the functor ? providing a monad structure on $\mathscr{G}$, the exponential ! is central to Harmer et al.'s novel characterisation of the category of games and innocent strategies. After giving a *distributive law* λ of ! over ?, they exhibit the *biKleisli category* $\mathrm{Kl}(\lambda)$ as a category of games an innocent strategies.

In this section we give functors $!, ? : \mathcal{Game} \to \mathcal{Game}$ constructed from the graphical representations of pointer structures, and use this formulation to give a category $\mathcal{Innocent}$ of games and innocent strategies.

4.4.1 The exponential functor !

Definition 194 (As in [HHM07], Definition 10.)**.** Given a game A, the game $!A$ is given by the diagram

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

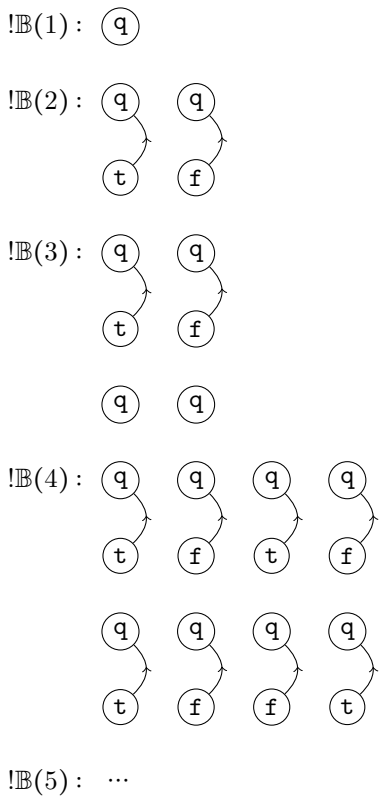
where moves in $(!A)(k)$ are pairs $(\Phi_k, \overrightarrow{a})$, where $\overrightarrow{a} \in A^k$ is a k-tuple of moves of A and where Φ_k is an O-heap graph with nodes labelled with $\overrightarrow{a}$ such that each Φ_k-thread is a play of A. The parent function $\pi_{!A}$ is given by truncation:

$$\pi_{!A} : (\Phi_k, \overrightarrow{a}) \mapsto (\Phi_k \restriction_{k-1}, \overrightarrow{a} \restriction_{k-1})$$

with $\overrightarrow{a} \restriction_{k-1}$ the first $k-1$ elements of $\overrightarrow{a}$.

The intuition with $!G$ is that O can always play by "backtracking" — opening a new copy of G at some previous O-move back in time.

Example 195. Recall the game $\mathbb{B}$ from Example 65. The game $!\mathbb{B}$ is given by its sets $!\mathbb{B}(k)$,



with $\pi_{!\mathbb{B}}$ given by truncation of labelled heap graphs.

In this way $!\mathbb{B}$ can viewed as $\mathbb{N}$ copies of the game $\mathbb{B}$, but where a new copy may only be opened by O, and only when the previous copy has been opened.

Though we are used to thinking of finite games A in terms of their finite game trees, in more complicated cases it may be harder to think about the game $!A$ in these terms, as its tree necessarily has infinite depth.

We will show that the assignment $!$ extends to a functor $\mathcal{G}\mathit{ame} \to \mathcal{G}\mathit{ame}$. First, let us examine a game $!A \multimap !B$.

Remark 196. Moves of $!A$ are O-heaps Φ, labelled with moves in A, such that every labelled Φ-thread is a play of A. The parent function $\pi_{!A}$ is given by truncation of these O-heaps. Likewise for $!B$. Moves of $!A \multimap !B$ are thus triples

$$\begin{aligned}(!A \multimap !B)(k) = \{(S_{m,n}, (\Phi_m, \overrightarrow{a}), (\Psi_n, \overrightarrow{b})) \mid\ & \\ & m + n = k, \\ & \Phi_m\text{-threads are plays in } A, \\ & \Psi_n\text{-threads are plays in } B\}\end{aligned}$$

with $\pi_{!A \multimap !B}$ given by truncation of the $\multimap$-schedule S. Technically, the nodes of $S : U_m \to V_n$ are labelled with labelled O-heaps, but since in U_m the labels are entirely determined

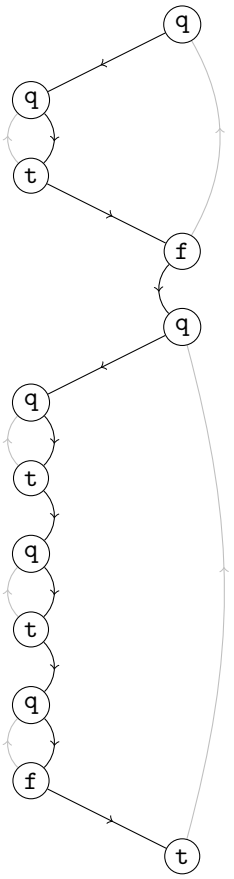


Figure 66: A position $(S_{8,4}, (\phi_8, \overrightarrow{a}), (\psi_4, \overrightarrow{b}))$ of the game $!\mathbb{B} \multimap !\mathbb{B}$. The black lines give the a $\multimap$-schedule $S_{8,4}$ and the gray lines give the O-heaps ϕ_8 and ψ_4. Labels come from $\overrightarrow{a} \in \mathbb{B}^8$ and $\overrightarrow{b} \in \mathbb{B}^4$. Note that each ψ-thread ad ϕ-thread is a play in $\mathbb{B}$.

by $(\phi_m, \overrightarrow{a})$ and $\pi_{!A}$ (truncation), we can view the complicated labels on U_m simply as attaching the heap graph Φ_m to the nodes of U_m with labels in A. Similarly, we can view the labels of V_n by attaching the graph Ψ_n. The parent function $\pi_{!A \multimap !B}$ now can be seen as truncating S (and automatically truncating Φ_m and Ψ_n) as necessary.

Example 197. Figure 66 shows a position of the game $!\mathbb{B} \multimap !\mathbb{B}$. The play of $!\mathbb{B} \multimap !\mathbb{B}$ which ends with this move is given by the sequence of truncation of the graph.

Remark 198. Figure 66 is not a progressive graph map as we have directed cycles. Rather, it is a superimposition of one downwardly progressive graph and one upwardly progressive graph.

Definition 199. Let $\sigma : A \multimap B$ be a strategy and hence a morphism $A \to B$ in $\mathcal{Game}$. The strategy $!\sigma : !A \multimap !B$ consists of positions $(S, (\Phi, \overrightarrow{a}), (\Psi, \overrightarrow{b}))$ satisfying:

(E1) $\Phi = S^* \Psi$

(E2) $[\Pi, S, \Psi]$-threads are deformable into positions of σ without moving nodes.

Theorem 200. *If $\sigma : A \multimap B$ is a strategy then $!\sigma : !A \multimap !B$ is a strategy.*

See Appendix A.4.2 for an example of a strategy $!\sigma$.

Theorem 201. $!$ *is a functor* $\mathcal{G}ame \to \mathcal{G}ame$.

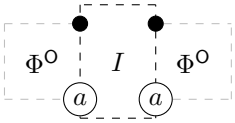
Proof. We are required to prove

$$!\kappa_A = \kappa_{!A} \tag{4.5}$$

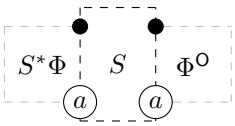
and, for strategies $\sigma : A \multimap B$ and $\tau : B \multimap C$,

$$!\sigma \| !\tau = !(\sigma \| \tau) \tag{4.6}$$

Identities. From the definitions, the strategy $\kappa_{!A}$ is the set of all positions

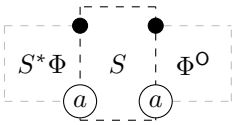


and the strategy $!\kappa_A$ is given by all positions



subject to the condition that $[\Pi, S, \Phi]$-threads are plays of κ_A (when taken as schedules). So it remains for us to prove that $S \sim I$ (and hence that $S^*\Phi = \Phi$, by Lemma 169).

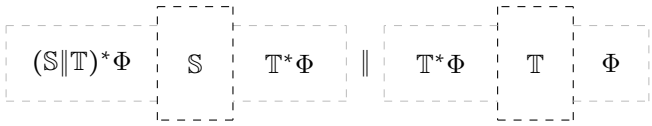
Consider some P-move (even-parity) on the right-hand side of the graph



The thread of this move in $[\Pi, S, \Phi]$ starts with an edge of S, and also must be a copycat (as it plays κ_A), so it must be an edge *across* S from right to left. Likewise the thread of any P-move on the left-hand side of $[\Pi, S, \Phi]$ starts with a thread across S from left to right. Therefore S is a copycat schedule and (4.5) holds.

Composites. We want to show that $!\sigma \| !\tau = !(\sigma \| \tau)$.

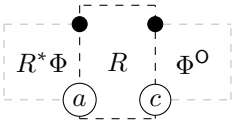
By Lemma 170, elements of $!\sigma \| !\tau$ are composites



(4.7)

so that threads of the left-hand component are schedules S playing σ, and threads of the right-hand component are schedules T playing τ. A thread of the composite (4.7) is a composite of a thread of $[(\mathbb{S}\|\mathbb{T})^*\Phi, \mathbb{S}, \mathbb{T}^*\Phi]$ and a thread of $[\mathbb{T}^*\Phi, \mathbb{T}, \Phi]$, by Proposition 180. Each thread plays $\sigma\|\tau$ by definition.

Likewise, elements of $!(\sigma\|\tau)$ are



(4.8)

such that threads are plays of $\sigma\|\tau$, and are thus schedules $S\|T$. We need to show that since threads of (4.8) are of the form $S\|T$, therefore $R = \mathbb{S}\|\mathbb{T}$ with $\mathbb{T}$ having threads playing τ and $\mathbb{S}$ having threads playing σ.

Since the threads of (4.8) are composites, they can be drawn as in their composition diagrams. Let us draw the threads in this way so that their nodes lie on the boundary of the vertical strip $[u, v] \times \mathbb{R}$, and so that for some $u' \in (u, v)$, the thread intersects $\{u'\} \times \mathbb{R}$ at the internal nodes of its composition diagram.

The heap graph Φ dictates how these threads are assembled together into the diagram $[\Phi^* R, R, \Phi]$, and from this diagram we can find a schedule in $[u', v] \times \mathbb{R}$ built from the fragments of R which lie in that strip. This schedule is by definition a schedule $\mathbb{T}$ playing $!\tau$ (one whose threads with Φ play τ). From Φ and this T, we can calculate $\mathbb{T}^*\Phi$, which allows us to likewise combine the fragments of R in $[u, u'] \times \mathbb{R}$ into a schedule $\mathbb{S}$, whose threads with $\mathbb{T}^*\Phi$ play σ.

Therefore $R = \mathbb{S}\|\mathbb{T}$, and (4.7) and (4.8) represent the same elements (up to deformation). □

The intuition with $!\sigma$ is as follows: $\sigma : A \multimap B$ provides a response for P to any possible O-move, including permission for P to swap between game components A and B. When in A, P plays "as O". One can think of σ as giving O complete freedom while restricting the actions of P. In $!A \multimap !B$, O can backtrack in B and P must respond to this new thread according to σ. If σ requires P to respond in A, $!\sigma$ requires P to backtrack in A.

4.4.2 Backtracking backtracks, ! as a comonad on *Game*

To examine the comonadic structure of !, we need to first consider games of the form $!!A$. Using the construction of ! from Definition 194 for a game G literally, the game $!!G$

is given by the diagram

$$(!!G)(1) \xleftarrow{\pi_{!!G}} (!!G)(2) \xleftarrow{\pi_{!!G}} \cdots$$

where

$$(!!G)(k) = \{(\Psi_k, \overrightarrow{(\Phi, \overrightarrow{g})}) \mid \begin{array}{l} \Psi_k \text{ is an O-heap,} \\ \Psi_k\text{-threads are plays in } !G\} \end{array} \tag{4.9}$$

so that $\overrightarrow{(\Phi, \overrightarrow{g})}$ is a sequence of moves from $!G$.

We can give an isomorphic representation of $!!G$ which is more intuitive. This is illustrated by example in Appendix A.4.3.

From (4.9), Ψ_k is an O-heap graph with nodes labelled by O-heaps Φ whose nodes are in-turn labelled by moves of G in such a way that a Ψ-thread gives a sequence of moves that is a play of $!G$. In other words, a sequence

$$(\Phi_k \restriction_{k-i})_{i=k-1}^{0}$$

of truncations of an O-heap graph Φ_k such that Φ_k-threads are plays of G.

For a node of Ψ (the "outer" heap), all information about the nodes of its label are given by the final such node, as the rest are given by Φ's action of truncation. So $(\Psi_k, \overrightarrow{(\Phi, \overrightarrow{g})})$ is isomorphic to a sequence of moves of G, together with two O-heap structures on it:

- Ψ acts by truncation on Φ.
- Φ gives "internal" pointer of the last node.

We must have that $\Psi \succcurlyeq \Phi$ as $\phi(i)$ must point to a truncation of i, which is somewhere reachable from i by Φ.

Conversely, a k-length sequence of moves of G together with two O-heap structures $\Psi_k \succcurlyeq \Phi_k$ on it such that Φ-threads are plays of G uniquely determines a labelling of Ψ by O-heaps Φ (ψ truncates) labelled by moves of G such that each Φ-thread is a play of G.

So we can represent the sets of $!!G$ as

$$(!!G)(k) = \{(\Psi_k, \Phi_k, \overrightarrow{g}) \mid \begin{array}{l} \Psi_k \succcurlyeq \Phi_k \text{ are O-heaps,} \\ \Phi_k\text{-threads are plays in } G\} \end{array}$$

Further, this can be extended to games such as $!!!G$, whose sets are

$$(!!!G)(k) = \{(\Theta_k, \Psi_k, \Phi_k, \overrightarrow{g}) \mid \begin{array}{l} \Theta_k \succcurlyeq \Psi_k \succcurlyeq \Phi_k \text{ are O-heaps,} \\ \Phi_k\text{-threads are plays in } G\} \end{array}$$

and so on.

See Appendix A.4.3 for an example of a game $!!G$.

An intuition with $!!G$ (and further applications of !) is as follows: In $!G$, O can always play by replaying a previous O-move to which P must respond. The record of which previous move O is playing is represented by a heap so that each thread of the heap is a complete play in G. In $!!G$, O has two "timelines" in which to backtrack: to a previous O-move in G (which we record by a heap Φ) or a previous O-move in $!G$ (which we record by a heap $\Psi \succcurlyeq \Phi$). Where as Φ can be though of as "undoing" moves in G, Ψ can be thought of as "undoing" moves in $!G$, including the backtracks. That way, Ψ acts as truncation on Φ, hence "undoing" Φ's backtracks, whereas Φ gives an "internal" pointer to the most recent move in G. Similarly, in $!!!G$ we have a third level of "undoing" represented by a heap $\Theta \succcurlyeq \Psi \succcurlyeq \Phi$, and so on.

We must also understand $!!\sigma : !!A \multimap !!B$.

Using the above understanding of games of the form $!!G$ allows us to unpack the definition of $!!\sigma$ to see that it consists of moves

$$(S, (\Psi, \Phi, \overrightarrow{a}), (\Psi', \Phi', \overrightarrow{b}))$$

subject to

$$\Psi \succcurlyeq \Phi \qquad \Psi' \succcurlyeq \Phi'$$
$$\Phi = S^* \Phi' \qquad \Psi = S^* \Psi'$$

and that $[\Pi, S, \Phi']$-threads are schedules in σ.

We can understand this by extending the intuition for $!\sigma$. In $!!\sigma : !!A \multimap !!B$, O can backtrack in B along either "timeline" Ψ' or Φ'. P must respond according to σ in the current thread, backtracking according to Ψ or Φ if appropriate.

Now we give the *comonad* structure of ! on $\mathcal{G}ame$. This definition is standard in the literature.

Definition 202. A **comonad** on a category $\mathscr{C}$ consists of a functor $U : \mathscr{C} \to \mathscr{C}$ and natural transformations

$$\varepsilon : U \Rightarrow \mathrm{id}_{\mathscr{C}} \qquad \delta : U \Rightarrow U^2$$

such that the conditions

$$\delta_U \circ \delta = U\delta \circ \delta \tag{Co1}$$
$$\varepsilon_U \circ \delta = \mathrm{id}_U = U\varepsilon \circ \delta \tag{Co2}$$

both hold.

We call ε the **counit** and δ the **comultiplication**.

For ! to have this structure on $\mathcal{G}ame$, we need $\varepsilon_A : !A \multimap A$ and $\delta_A : !A \multimap !!A$.

$\varepsilon_A : !A \multimap A$ is given by all positions of the form $(I, (\Pi, \overrightarrow{a}), a)$, with I a copycat schedule, Π a graph of the maximal heap π and $\overrightarrow{a}$ a sequence of moves in A with final move a.

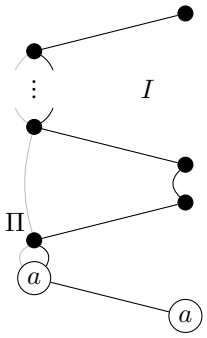
$\delta_A : !A \multimap !!A$ is given by all positions of the form $(I, (\Psi, \overrightarrow{a}), (\Phi, \Psi, \overrightarrow{a}))$, with $\Phi \succcurlyeq \Psi$, I a copycat schedule, and $\overrightarrow{a}$ a sequence of moves of A.

Lemma 203. *ε_A is natural in A.*

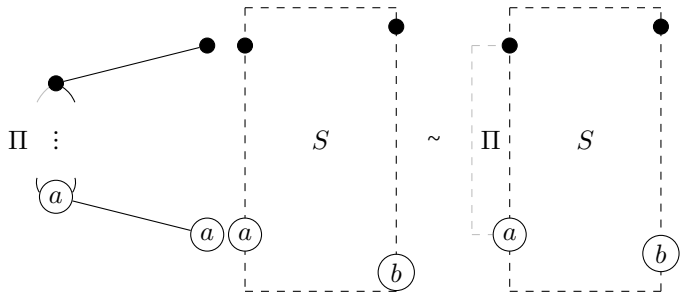
Proof. Consider a strategy $\sigma : A \multimap B$. We must show that

$$!\sigma \| \varepsilon_B = \varepsilon_A \| \sigma \tag{4.10}$$

As a strategy, $\varepsilon_A :!A \multimap A$ consists of all plays of the form $(I, (\Pi, \overrightarrow{a}), a)$, i.e.



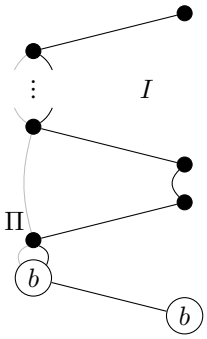
Therefore, composing on the right with a play (S, a, b) in $\sigma : A \multimap B$ gives



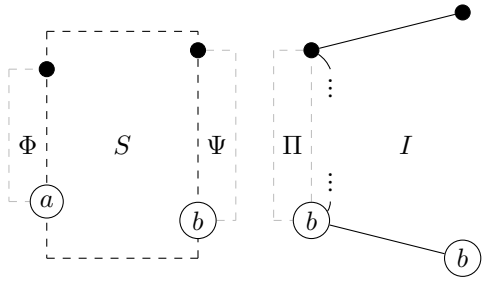
a play $(S, (\Pi, \overrightarrow{a}), b)$.

So $\varepsilon_A \| \sigma :!A \multimap B$ is the set of all plays $(S, (\Pi, \overrightarrow{a}), b)$ so that $(S, a, b) \in \sigma$, and the previously mentioned conditions are satisfied.

Similarly, $\varepsilon_B :!B \multimap B$ consists of all plays resembling



And the strategy $!\sigma : !A \multimap !B$ contains plays $(S, (\Phi, \overrightarrow{a}), (\Psi, \overrightarrow{b}))$ where $\Phi = S^*\Psi$. Plays in $!\sigma \| \varepsilon_B$ consist of compositions of plays $(S, (\Phi, \overrightarrow{a}), (\Psi, \overrightarrow{b})) \| (I, (\Pi, \overrightarrow{b}), b)$;



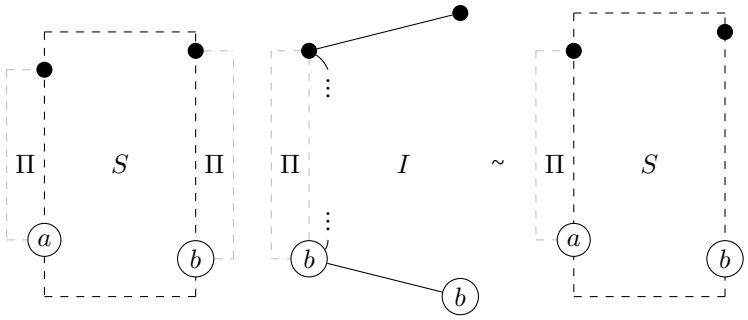
For these to be composable, we require

$$\Psi \sim \Pi$$

but then from (E1) and Lemma 163 we have

$$\Phi = S^*\Pi = \Pi$$

So $!\sigma \| \varepsilon_B$ consists of all plays



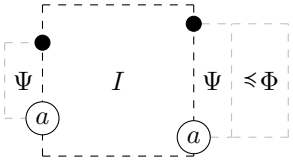
and (4.10) holds, as required. □

Lemma 204. *δ_A is natural in A.*

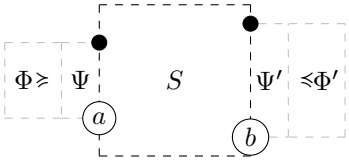
Proof. Consider a strategy $\sigma : A \multimap B$. We must show that

$$\delta_A \| !!\sigma = !\sigma \| \delta_B \tag{4.11}$$

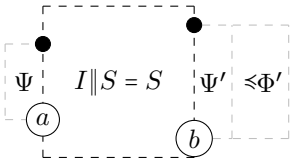
The component $\delta_A : !A \multimap !!A$ is a strategy consisting of plays



and $!!\sigma : !!A \multimap !!B$ consists of plays



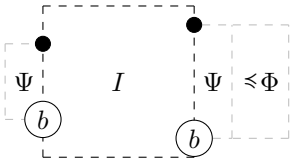
In $\delta_A \| !!\sigma : !A \multimap !!B$, these plays are composed to give



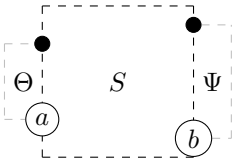
such that $\Psi = S^*\Psi'$. So

$$\delta_A \| !!\sigma = \{(S, (\Psi, \overrightarrow{a}), (\Psi', \Phi', \overrightarrow{b})) \mid (S, a, b) \in \Sigma\}$$

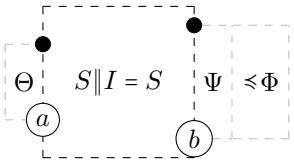
Similarly, $\delta_B : !B \multimap !!B$ consists of all plays



and $!\sigma : !A \multimap !B$ is all plays



such that $S^*\Psi = \Theta$. Then plays in $!\sigma \| \delta_B$ are composites



such that $S^*\Psi = \Theta$, so (4.11) holds, as required. □

Lemma 205. *ε and δ satisfy comonad axioms* (Co1) *and* (Co2).

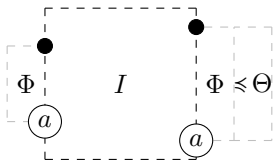
Proof. We are required to prove:

$$\delta_A \| \delta_{!A} = \delta_A \| !\delta_A \tag{4.12}$$

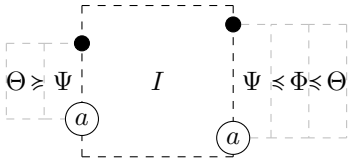
$$\delta_A \| \varepsilon_{!A} = \mathrm{id}_{!A} \tag{4.13}$$

$$\delta_A \| !\varepsilon_A = \mathrm{id}_{!A} \tag{4.14}$$

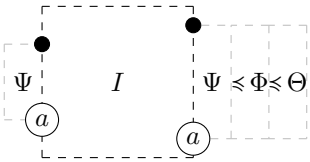
The strategy $\delta_A : !A \multimap !!A$ consists of plays



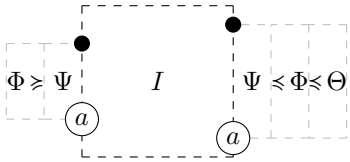
so the strategy $\delta_{!A} : !!A \multimap !!!A$ consists of plays



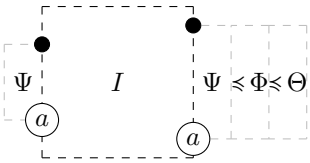
So that the composition $\delta_A \| \delta_{!A} : !A \multimap !!!A$ contains



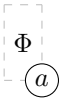
Similarly, since $!\delta_A : !!A \multimap !!!A$ has plays



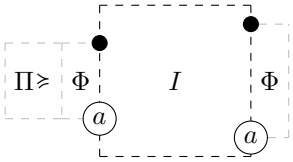
the composite $\delta_A \| !\delta_{!A}$ contains



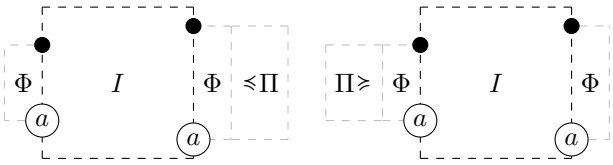
and (4.12) holds, as required. Recall that the game $!A$ has plays



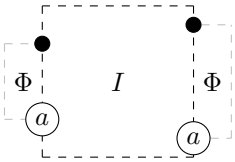
so that the strategy $\varepsilon_{!A} : !!A \multimap !A$ consists of positions



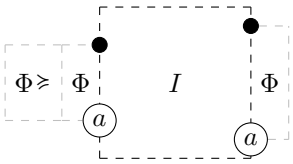
so that $\delta_A \| \varepsilon_{!A} : !A \multimap !A$ consists of compositions



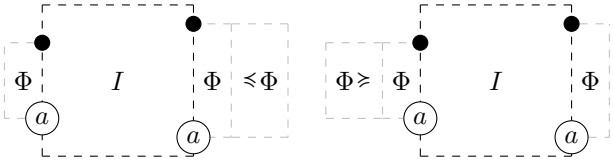
i.e.



which is exactly a play of $\mathrm{id}_{!A} : !A \multimap !A$. So (4.13) holds.
Similarly the strategy $!\varepsilon_A : !!A \multimap !A$ consists of positions



so that the composition $\delta_A | !\varepsilon_A : !A \multimap !A$ is



and (4.14) holds, as required. □

Theorem 206. $(!, \varepsilon, \delta)$ *is a comonad on* $\mathcal{G}ame$.

Proof. This follows by definition from Lemmata 203, 204 and 205. □

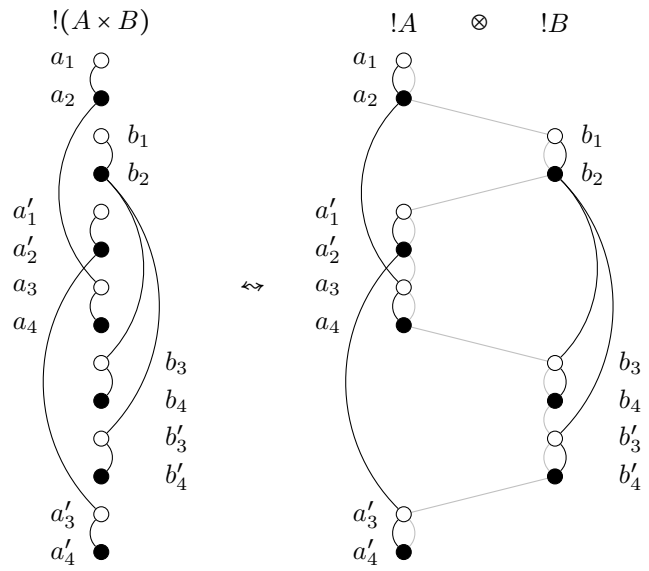


Figure 67: An illustration of the isomorphism between an example of a game $!(A \times B)$ and $!A \otimes !B$.

For games A and B, the product game $A \times B$ has moves $A \cup B$, but where O chooses to play in *either* A *or* B, with no possibility of switching for O or P afterwards. The game 1 which is the identity of $\times$ is the game with no elements.

In [HHM07], Harmer et al. give ! as the *linear exponential comonad* (terminology of [HS99, Sch04], cf. [Mel]). The Seely isomorphisms [See87, Bie95]

$$!1 \cong I \tag{Se1}$$

$$!(A \times B) \cong !A \otimes !B \tag{Se2}$$

follow from the structure of the games. I is the empty game; the game with no moves. Therefore the game $A \multimap I$ has no moves for any game A. As $A \multimap I$ has no moves, there is only one strategy (the empty strategy) on it, and since such strategies provide morphisms $A \to I$ in $\mathcal{G}ame$, I is a terminal object. By definition of !, the game !1 is given by heaps graphs with nodes labelled with moves of 1, but since 1 has no moves, the game !1 doesn't either. Hence (Se1) holds.

The isomorphism of games (Se2) comes as backtracking in both $!(A \times B)$ and $!A \otimes !B$ is permitted only within the same component. In the game $A \times B$, once O has chosen A or B to play in, neither O or P can switch, so the positions of $!(A \times B)$ are given by those O-heaps with at least two disjoint threads, where each thread plays in A or B exclusively. On the other hand, a position in $!A \otimes !B$ is given by a mediating $\otimes$-schedule with heaps on the left- and right-hand side whose threads are plays in A or B exclusively. This witnesses the isomorphism (Se2) via a diagrammatic manoeuvre, illustrated by example in Figure 67.

Furthermore the component of the monoidal comonad morphism family ϕ at games A

and B is given by

$$\begin{array}{ccccc} !A \otimes !B & & \xrightarrow{\quad\phi_{A,B}\quad} & & !(A \otimes B) \\ \wr\| & & & & \| \\ !(A \times B) & \xrightarrow{\delta_{A\times B}} & !!(A \times B) \cong & !(!A \otimes !B) & \xrightarrow{!(\varepsilon_A \otimes \varepsilon_B)} !(A \otimes B) \end{array}$$

(See Definition 235.)

Theorem 207 ([HHM07], Theorem 4.5.)**.** $(!, \varepsilon, \delta)$ *is a linear exponential comonad on* $\mathcal{G}ame$*.*

4.4.3 Backtracking for P: the functor ?

Just as ! gives O permission to backtrack, there is also a similar structure allowing P to backtrack. The situation is very similar to that with !, so we will proceed through the following analogous definitions and theorems quickly.

Definition 208 (Similar to [HHM07], Definition 12.)**.** Given a game A, the game $?A$ is given by the diagram

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

where positions in $(?A)(k)$ are pairs $(\Psi_k, \overrightarrow{a})$, where $\overrightarrow{a} \in A^k$ is a k-tuple of positions of A and where Ψ_k is a P-heap graph with nodes labelled with $\overrightarrow{a}$ such that each Ψ_k-thread is a play of A. The parent function $\pi_{?A}$ is given by truncation:

$$\pi_{?A} : (\Psi_k, \overrightarrow{a}) \mapsto (\Psi_k \restriction_{k-1}, \overrightarrow{a} \restriction_{k-1})$$

Definition 209. Given a strategy $\sigma : A \multimap B$, the strategy $?\sigma : ?A \multimap ?B$ has positions $(S, (\Phi, \overrightarrow{a}), (\Psi, \overrightarrow{b}))$ satisfying

(Q1) $S_* \Phi = \Psi$

(Q2) $[\Phi, S, \Pi]$-threads play in σ.

In analogy to Theorems 200 and 201, we get:

Theorem 210. *If* $\sigma : A \multimap B$ *is a strategy then* $?\sigma : ?A \multimap ?B$ *is a strategy.*

Theorem 211. $?$ *is a functor* $\mathcal{G}ame \to \mathcal{G}ame$*.*

Remark 212. Again, repeated application of the ? endofunctor produces games which may be given in a simplified (yet isomorphic) form:

$$(??A)(k) = \{(\Psi^{\mathsf{P}}, \Phi^{\mathsf{P}}, \overrightarrow{a}) \mid \Phi\text{-threads are plays in } A, \ \Psi \succcurlyeq \Phi\}$$

$$(???A)(k) = \{(\Theta^{\mathsf{P}}, \Psi^{\mathsf{P}}, \Phi^{\mathsf{P}}, \overrightarrow{a}) \mid \Phi\text{-threads are plays in } A, \ \Theta \succcurlyeq \Psi \succcurlyeq \Phi\}$$

(etc.)

where the $^{\mathsf{P}}$ decoration indicates that the heaps are P-heaps.

The following definition is standard in the literature.

Definition 213. A **monad** on a category $\mathscr{C}$ is a comonad on $\mathscr{C}^{\mathrm{op}}$. In other words, it is a functor $T : \mathscr{C} \to \mathscr{C}$ and natural transformations

$$\eta : \mathrm{id}_{\mathscr{C}} \implies T$$
$$\mu : T^2 \implies T$$

called the **unit** and **multiplication** respectively, which satisfy

$$\mu \circ \mu_T = \mu \circ T\mu \tag{Mo1}$$
$$\mu \circ \eta_T = \mathrm{id} = \mu \circ T\eta \tag{Mo2}$$

Definition 214. For the functor $? : \mathcal{G}ame \to \mathcal{G}ame$ and for a game A, we define:

$$\eta_A = \{(I, a, (\Pi, \overrightarrow{a}))\} : A \multimap ?A$$

$$\mu_A = \{(I, (\Psi, \Phi, \overrightarrow{a}), (\Psi, \overrightarrow{a})) \mid \Psi \succcurlyeq \Phi\} : ??A \multimap ?A$$

Again, the naturality of this μ_A and η_A, and the monad laws (Mo1) and (Mo2), follow in a similar manner to Theorem 206.

Theorem 215 ([HHM07], Theorem 4.7.)**.** $(?, \eta, \mu)$ *is a monoidal monad on* $\mathcal{G}ame$*.*

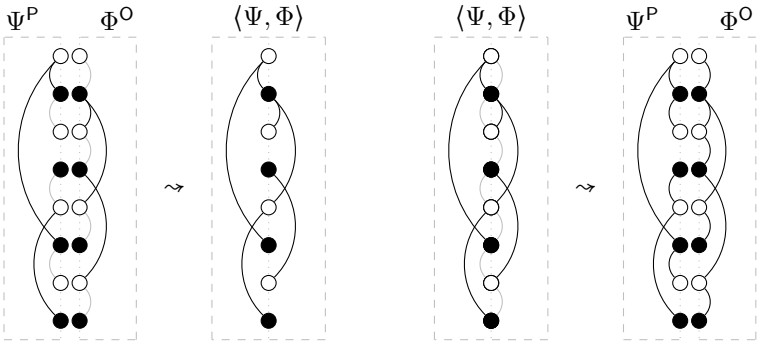
4.4.4 Games constructed with both ? and !

We are interested in games formed by application of both ? and !. To consider this in detail, however, we first observe that any parity heap graph Ω with n nodes may be uniquely given by a pair of an O-heap graph and a P-heap graph, each with n nodes. The O-heap graph determines Ω's outward edges from nodes of odd parity (O-positions) and the P-heap graph determines Ω on nodes of even parity (P-positions). Furthermore given an O-heap and a P-heap, together they specify a unique parity heap in the same way.

Definition 216. Given an O-heap graph Φ and a P-heap Ψ, both on $[n]$, the heap pair $\langle \Psi, \Phi \rangle$ is a heap graph on $[n]$ with:

- The outward edge from an odd-parity node taken from Φ
- The outward edge from an even-parity node taken from Ψ

Remark 217. A similar construction appears in [HHM07], but, as with the $[\Psi, S, \Phi]$ construction, we write the symbols in the reverse order here, and for similar reasons.



(a) Construction of a heap pair $\langle \Psi, \Phi \rangle$ from heaps Ψ^{P} and Φ^{O}.

(b) Heaps Ψ^{P} and Φ^{O} may be recovered from any parity heap Ω.

Figure 68: Every parity heap is made of an O-heap and a P-heap.

Remark 218. Equivalently, we can define $\langle \Psi, \Phi \rangle$ to be constructed by placing Ψ in standard configuration with its edges on the left of its nodes, and Φ in standard configuration with its edges to the right of its nodes, so that Ψ's and Φ's corresponding nodes coincide, and then removing the familiar "extended $\}$" shape. This shape is comprised of exactly those edges of Ψ and Φ which are forced to exist by the definitions of P- and O-heaps, but are not particular to Ψ and Φ.

Conversely, from a parity heap graph Ω, we can recover a P-heap Ψ and an O-heap Φ on the same nodes by adding in the "extended $\}$" edges.

Both of these can be seen in Figure 68.

Observe that Φ is an O-heap if and only if $\Phi = \langle \Pi, \Phi \rangle$, and that Φ is a P-heap if and only if $\Phi = \langle \Phi, \Pi \rangle$.

In terms of our current definitions of ! and ?, a game $!?A$ would be given by

$$(!?A)(k) = (!(?A))(k) = \{(\Phi, \overrightarrow{(\Psi, \overrightarrow{a})}) \mid \Phi\text{-threads are plays of } ?A\}$$

I.e., Φ is an O-heap whose nodes are labelled by P-heaps such that each Φ-thread is labelled by successive truncations of the same P-heap.

However, in a similar fashion to the discussion of $!!A$ in Section 4.4.2, we can give the following isomorphic representation:

$$\begin{aligned}(!?A)(k) = \{(\Phi, \Omega, \overrightarrow{a}) \mid\ & \Phi \text{ is an O-heap,} \\ & \Phi \succcurlyeq \Omega, \\ & \Omega\text{'s O-moves are given by } \Phi, \\ & \Omega\text{-threads are plays of } A\}\end{aligned}$$

Or, equivalently,

$$(!?A)(k) = \{(\Phi, \langle \Psi, \Phi \rangle, \overrightarrow{a}) \mid \begin{array}{l} \Phi \text{ is an O-heap,} \\ \Psi \text{ is a P-heap,} \\ \Phi \succcurlyeq \langle \Psi, \Phi \rangle, \\ \langle \Psi, \Phi \rangle\text{-threads are plays of } A\} \end{array}$$

In general, we can view ? as providing an additional timeline in which P may backtrack, and vice versa with ! for O. Thus, viewing an O-heap Φ as a pair $\langle \Pi, \Phi \rangle$ and a P-heap Ψ as a pair $\langle \Psi, \Pi \rangle$; we can easily and arbitrarily construct games with a sequence of ?s and !s by altering the P-heap or O-heap components of the "smaller" pairs accordingly, always maintaining the necessary heap order. For example (with a slight abuse of notation and omitted conditions for the sake of brevity):

$$\begin{array}{rl} A = \{ & a \quad \} \\ !A = \{(& \langle \Pi, \Phi \rangle, \overrightarrow{a}) \quad \} \\ !!A = \{(& \langle \Pi, \Psi \rangle \succcurlyeq \langle \Pi, \Phi \rangle, \overrightarrow{a}) \quad \} \\ ?!!A = \{(& \langle \Theta, \Pi \rangle \succcurlyeq \langle \Theta, \Psi \rangle \succcurlyeq \langle \Theta, \Phi \rangle, \overrightarrow{a}) \quad \} \\ !?!!A = \{(& \langle \Pi, \mathrm{X} \rangle \succcurlyeq \langle \Theta, \mathrm{X} \rangle \succcurlyeq \langle \Theta, \Psi \rangle \succcurlyeq \langle \Theta, \Phi \rangle, \overrightarrow{a}) \quad \} \\ ?!?!!A = \{(& \langle \mathrm{P}, \Pi \rangle \succcurlyeq \langle \mathrm{P}, \mathrm{X} \rangle \succcurlyeq \langle \Theta, \mathrm{X} \rangle \succcurlyeq \langle \Theta, \Psi \rangle \succcurlyeq \langle \Theta, \Phi \rangle, \overrightarrow{a}) \quad \} \\ \text{(etc.)} & \end{array}$$

4.4.5 A distributive law of ! over ?

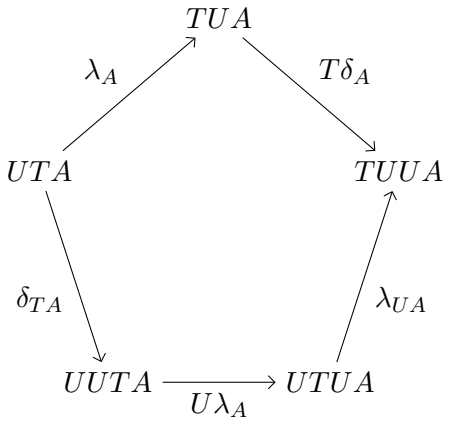
Now that we have an understanding of games formed by ? and ! together, we will give, as Harmer et al. did, a *distributive law* of ? over !.

Definition 219 ([HHM07], Definition 1; cf. [PW02].)**.** Given a category $\mathscr{C}$, and a monad (T, η, μ) and comonad (U, ε, δ) on $\mathscr{C}$, the **distributive law of** U **over** T is a natural transformation

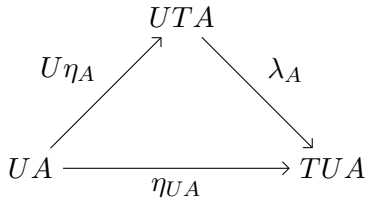
$$\lambda : UT \implies TU$$

so that the following four diagrams commute as diagrams of natural transformations (shown at component object A):

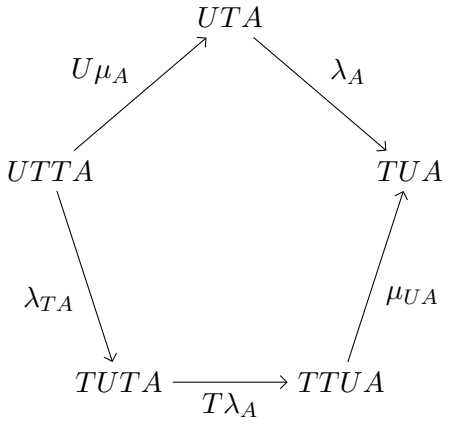
$$\begin{array}{ccc} & TUA & \\ {\scriptstyle \lambda_A}\nearrow & & \searrow{\scriptstyle T\varepsilon_A} \\ UTA & \xrightarrow[\varepsilon_{TA}]{} & TA \end{array} \tag{D1}$$



(D2)



(D3)



(D4)

In the case of *Game*, with the comonad ! and the monad ?, we define the following family of morphisms which we claim is a distributive law.

Definition 220 (As in, [HHM07], Definition 14.)**.** For a game A, the strategy $\lambda_A : !?A \multimap ?!A$ consists of all positions of the form

$$(I, (\Phi^{\mathsf{O}}, \langle \Psi, \Phi \rangle, \overrightarrow{a}), (\Psi^{\mathsf{P}}, \langle \Psi, \Phi \rangle), \overrightarrow{a})$$

To show that λ_A is a distributive law, we will need to consider strategies of the form $!?\sigma$ and $?!\sigma$. To give a shorthand isomorphic representation of these, as we have before, we will need to make use of the following lemma, which is stated in [HHM07].

Lemma 221. *Let $S : U_m \to V_n$ be a $\multimap$-schedule, and let Ψ be a P-heap graph with nodes U_m and Φ be an O-heap graph with nodes V_n. Then the following are true:*

(i) $S^*\Phi \gtrsim \langle \Psi, S^*\Phi \rangle \implies \Phi \gtrsim \langle S_*\Psi, \Phi \rangle$.

(ii) $S_*\Psi \gtrsim \langle S_*\Psi, \Phi \rangle \implies \Psi \gtrsim \langle \Psi, S^*\Phi \rangle$.

Proof. We will prove (i); (ii) follows by a similar argument.

Suppose that the antecedent of (i) holds, so that

$$S^*\Phi = \langle \Pi, S^*\Phi \rangle \gtrsim \langle \Psi, S^*\Phi \rangle \tag{4.15}$$

We will show that the consequent of (i) holds, i.e. that

$$\Phi = \langle \Pi, \Phi \rangle \gtrsim \langle S_*\Psi, \Phi \rangle \tag{4.16}$$

By considering the nodes of V_n in reverse order, as in Proposition 144, a $\dot{\gtrsim}$-sequence of graphs may be constructed to yield (4.16). We will argue for an arbitrary node of V_n.

Both heaps in (4.16) have (corresponding) nodes in V_n. As they are P-heaps, all edges from odd-parity nodes are equal (they are edges from Φ). So $\langle \Pi, \Phi \rangle$ and $\langle S_*\Psi, \Phi \rangle$ may only differ in edges outward from even-parity nodes. Therefore, consider an even-parity node $y \in V_n$. (Of course, y is really a corresponding pair of nodes in the two heap graphs $\langle \Pi, \Phi \rangle$ and $\langle S_*\Psi, \Phi \rangle$.)

The outward edge from y in is from $\langle S_*\Psi, \Phi \rangle$ is an edge from S, and so leads to an even-parity node $x \in U_m$. We want to know where the path which continues from this edge ends up when it returns to V_n.

This node x is also a node of $\langle \Psi, S^*\Phi \rangle$, and since it is of even parity, in these graphs its outward edge is taken from Π or Ψ. Its path then continues in a sequence of edges before (perhaps) returning to V_n via an edge of S. Since (4.15) holds, for each of the edges in the sequence, one of the following must be true:

(a) The edge is from an even-parity node and is the same in $\langle \Psi, S^*\Phi \rangle$ and $\langle \Pi, S^*\Phi \rangle$ (so is an edge of Π).

(b) The edge is from an even-parity node and is missing only in $\langle \Psi, S^*\Phi \rangle$.

(c) The edge is from an even-parity node, and in $\langle \Psi, S^*\Phi \rangle$ corresponds to a path in $\langle \Pi, S^*\Phi \rangle$.

(d) The edge is from an odd-parity node, so is an edge of S which lands to U_m.

(e) The edge is from an odd-parity node, so is an edge of S which lands in V_n.

If the sequence of edges in $\langle \Psi, S^*\Phi \rangle$ does not contain an edge for which (b) or (c), is true, then it must return to the node of V_n immediately above y. In this case, the edge from y in $\langle S_*\Psi, \Phi \rangle$ is equal to that in $\langle \Pi, \Phi \rangle$.

If the sequence of edges contains an edges for which (b) holds, then the path is broken and the edge from y is missing only in $\langle S_*\Psi, \Phi \rangle$.

If the sequence of edges contains an edge for which (c) holds, then the path will return to V_n at a node above y in the path of $\langle \Pi, \Phi \rangle$.

In any event, by altering only this edge from y, we get the next heap in our $\dot{\succcurlyeq}$-sequence. $\square$

On the basis of Lemma 221, we give the following denotations of strategies of the form $!?\sigma$ and $?!\sigma$:

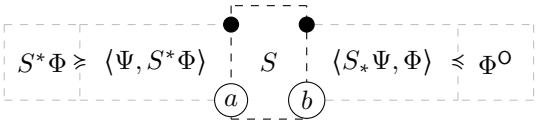
$$!?\sigma = \{(S, (S^*\Phi \succcurlyeq \langle \Psi, S^*\Phi \rangle, \overrightarrow{a}), (\Phi^{\mathsf{O}} \succcurlyeq \langle S_*\Psi, \Phi \rangle, \overrightarrow{b})) \mid [\Psi, S, \Phi]\text{-threads play } \sigma\} \tag{4.17}$$

$$?!\sigma = \{(S, (\Psi^{\mathsf{P}} \succcurlyeq \langle \Psi, S^*\Phi \rangle, \overrightarrow{a}), (S_*\Psi \succcurlyeq \langle S_*\Psi, \Phi \rangle, \overrightarrow{b})) \mid [\Psi, S, \Phi]\text{-threads play } \sigma\} \tag{4.18}$$

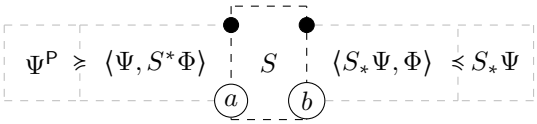
Theorem 222 ([HHM07], Theorem 4.9.)**.** *λ_A gives a distributive law $\lambda : !? \Longrightarrow ?!$.*

Proof. We are required to prove that λ_A is natural in A, and that axioms (D1)–(D4) hold. Here we will show naturality and (D2), with the other distributive axioms following along very similar lines.

Naturality. From (4.17) and (4.18), we have that $!?\sigma$ and $?!\sigma$ consist of all positions

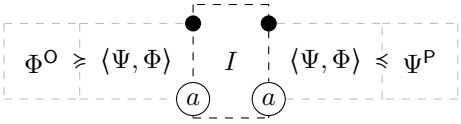


and

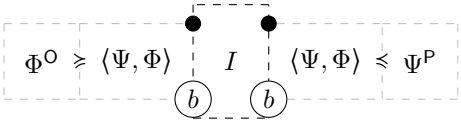


respectively, so that $[\Psi, S, \Phi]$-threads are plays of σ.

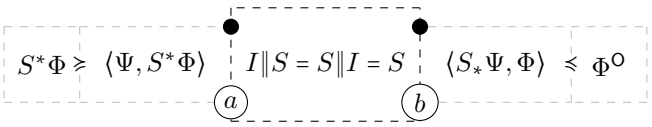
We also have that λ_A is the set of all positions



and that λ_B is the set of all positions

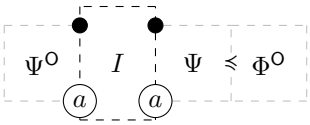


Therefore, since I is the identity of $\multimap$-schedule composition, both $!?\sigma \| \lambda_B$ and $\lambda_A \| ?!\sigma$ are strategies on $!?A \multimap ?!B$ which consist of all plays

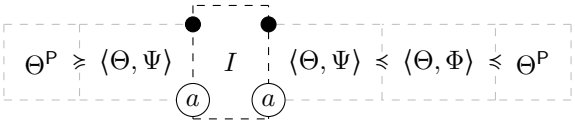


such that $[\Psi, S, \Phi]$-threads are plays of σ.

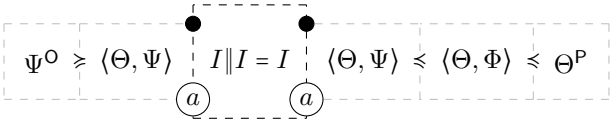
(D2) **holds.** Recall that $\delta_A : !A \multimap !!A$ is the strategy consisting of all positions



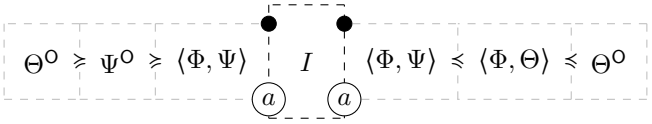
So $?\delta_A : ?!A \multimap ?!!A$ consists of all positions



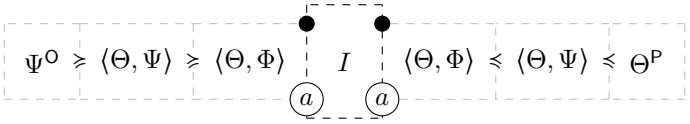
So the composite $\lambda_A \| ?\delta_A : !?A \multimap ?!!A$ consists of all positions



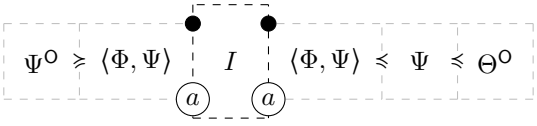
On the other hand, $!\lambda_A : !!?A \multimap !?!A$ is the set of positions



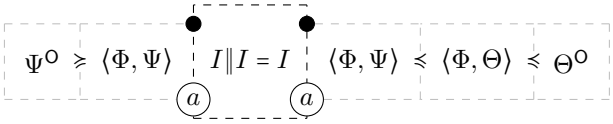
and $\lambda_{!A} : !?!A \multimap ?!!A$ has positions



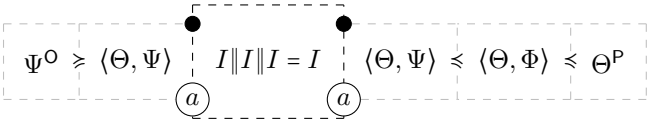
Finally, $\delta_{?A} : !?A \multimap !!?A$ has positions



so the composite $\delta_{?A} \| !\lambda_A : !?A \multimap !?!A$ is all



and the composite $\delta_{?A} \| !\lambda_A \| \lambda_{!A} : !?A \multimap ?!!A$ is the set of all positions



and so we have

$$\lambda_A \| ?\delta_A = \delta_{?A} \| !\lambda_A \| \lambda_{!A} : !?A \multimap ?!!A$$

Similarly routine arguments show (D1), (D3) and (D4). □

4.4.6 The category *Innocent* of games and innocent strategies

Harmer et al. use the distributive law λ to construct a *biKleisli category* $\mathrm{Kl}(\lambda)$.

Definition 223 ([BG91], Definition 2.2.)**.** Given a comonad (U, ε, δ) on a category $\mathscr{C}$, the **coKleisli category** for the comonad, $\mathrm{Kl}(U)$, is given by:

- Objects of $\mathrm{Kl}(U)$ are objects of $\mathscr{C}$.
- A morphism $A \to B$ in $\mathrm{Kl}(U)$ a morphism $UA \to B$ in $\mathscr{C}$.
- The identity morphism on the object A is ε_A.
- The composite of the morphism $f : A \to B$ and $g : B \to C$ in $\mathrm{Kl}(U)$ is given by the composite
$$UA \xrightarrow{\delta_A} UUA \xrightarrow{Uf} UB \xrightarrow{g} B$$
in $\mathscr{C}$.

Definition 224 ([BW85], p. 88.)**.** Given a monad (T, η, μ) on a category $\mathscr{C}$, the **Kleisli category** for the monad, $\mathrm{Kl}(T)$, is given by:

- Objects of $\mathrm{Kl}(T)$ are objects of $\mathscr{C}$.
- A morphism $A \to B$ in $\mathrm{Kl}(T)$ is a morphism $A \to TB$ in $\mathscr{C}$.
- The identity morphism on the object A is η_A.
- The composite of the morphism $f : A \to B$ and $g : B \to C$ in $\mathrm{Kl}(T)$ is given by the composite
$$A \xrightarrow{f} TB \xrightarrow{Tg} TTC \xrightarrow{\mu_C} TC$$
in $\mathscr{C}$.

In $\mathrm{Kl}(U)$ and $\mathrm{Kl}(T)$, the comonad and monad axioms provide the associativity and unit axioms of the categories.

Definition 225 ([PW02, HHM07].)**.** Given a distributive law λ of a comonad (U, ε, δ) over a monad (T, η, μ) on a category $\mathscr{C}$, the **biKleisli category** for the distributive law, $\mathrm{Kl}(\lambda)$, is given by:

- The objects of $\mathrm{Kl}(\lambda)$ are objects of $\mathscr{C}$.
- A morphism $A \to B$ of $\mathrm{Kl}(\lambda)$ is a morphism $UA \to TB$ in $\mathscr{C}$.
- The identity morphism on A is the composite

  $$UA \xrightarrow{\varepsilon_A} A \xrightarrow{\eta_A} TA$$

  in $\mathscr{C}$.
- The composite of the morphism $f : A \to B$ and $g : B \to C$ in $\mathrm{Kl}(\lambda)$ is given by the composite

  $$UA \xrightarrow{\delta_A} UUA \xrightarrow{Uf} UTB \xrightarrow{\lambda_B} TUB \xrightarrow{Tg} TTC \xrightarrow{\mu_C} TC$$

  in $\mathscr{C}$.

In terms of our $\lambda :!? \Longrightarrow ?!$, we have:

Definition 226 (As in [HHM07], Definition 2.)**.** Recall the distributive law $\lambda :!? \Longrightarrow ?!$ on *Game*. The **category of games an innocent strategies**, *Innocent*, is defined to be $\mathrm{Kl}(\lambda)$.

Chapter 5

Further directions

5.1 This thesis and future work

In this thesis we have given a detailed and formal account of a graphical foundation for some methods in game semantics. We have seen this specifically applied to give a new angle of understanding to a categorical account of games and innocent strategies.

As previously detailed, we have chosen to work in the graphical setting of Joyal and Street's *progressive plane graphs*, as progressive graphs in the plane are sufficient to capture a great deal of the structure otherwise recorded in several combinatorial devices.

The setting of progressive plane graphs has been used for many formal accounts of graphical methods in algebra. This is likely due in part to the abundance of planes in a researcher's environment on which notation may be drawn. But the use of planes also may not be entirely coincidental. More general investigations into what game semantic structures may be represented in the progressive planar setting would certainly be interesting.

The facts about the categorical structures of $\mathcal{O}\mathit{heap}$ and $\mathcal{S}\mathit{ched}$ established in Section 4.3 are interesting but have not been thoroughly investigated here.

The game semantic setting described in this thesis is deterministic. Extensions into some nondeterministic settings may be possible, though it is not yet clear whether the progressive plane setting would be sufficient for this.

5.2 Appraisal

The broad intent of this work can be summarised into two themes. First, to lend mathematical foundations and legitimacy to the informal arguments (such as associativity of composition of $-\circ$-schedules) and representations (such as interleaving graphs) which are perhaps already familiar to game semantics researchers. Second, to explore these formulations of game semantics and provide examples of how clear arguments can be

used without too many elided details, as is perhaps more common in the combinatorial setting.

There are several examples presented here which seem to make especially successful use of the graphical setting. The proof of associativity of composition of $\multimap$-schedules and the "unfolding" isomorphism which facilitates a straight forward proof of the symmetric monoidal structure of *Game* in particular make heavy use of the plane to avoid repeated reindexing. As previously discussed, much of the plane's strength in this regard comes from automatic facts such as "left of left is left" and the intermediate value theorem, whose analogues in the combinatorial setting are perhaps less immediate.

There are other cases — such as the construction of the distributive law $\lambda :!? \Longrightarrow ?!$ and the biKleisli category *Innocent* — where the graphical setting buys us less, though it is at least no hindrance.

The aim of using a planar graphical calculus is to aid humans doing mathematics, as often graphical arguments and manipulations are more immediately obvious to observers than their combinatorial analogues. There is also, of course, much research done with mechanical aids and much interest in techniques which lend themselves well to automation and mechanisation (for example, the work of Ghica et al. mentioned below). There is interest in developing graphical calculi which are amenable to automatic reasoning (see e.g. [DDK10, Kis12]), though these tend to require a greatly restricted class of diagrams, with those diagrams researchers draw with a pen being further removed from the definitions. While the formulation we have chosen is less suitable for automation, it benefits from being highly representative of what is already drawn.

It is not our intention that every game semanticist immediately adopt our definitions, or begin to phrase all their work in terms of Hausdorff graphs. Rather it is our intention to demonstrate that the reasoning methods already employed *can be made precise*, and that the arguments used are formalisable. One hope is that this reassurance will permit greater experimentation with graphical reasoning techniques, based on a concrete foundation.

5.3 Some related work

This work should be regarded in the context of a great deal of highly relevant related work in game semantics. A brief overview of some of this wider context is given in Chapter 1. In this section we wish to draw specific attention to a few other things in the game semantics literature.

Of course, the graphical methods we have described and employed here work in parallel to the highly successful combinatorial approach in [HHM07].

Other work relating to topological understandings of game semantics include that of Melliès's work on game semantics in string diagrams [FY89, Mel12b] and dialogue categories for tensorial logic [Mel12a]. Similar discussion can also be found in [HS02].

One aim of this work has been to give a new understanding of the pointer structures. Another approach to the same area has come from *nominal* game semantics, in the work

of Ghica, Tzevelekos, Gabbay and others [Tze08, GG12]. One advantage to the nominal setting, as is amongst its aims, is mechanisability of the methods (something to which the graphical methods of this thesis are perhaps not well suited).

The use of graphical calculi as a tool to reduce syntactic clutter and facilitate intuitive reasoning in more diverse settings is also present across the literature in mathematics, computer science and logic. This has ranged from graphical languages for monoidal categories [JS88, JS91, JS92, JS93, Shu94, JSV96, Mel06, Sel11] to applications in physics and other natural sciences [Pen71, DL04, CD08, CSC10, DDK10, BS11, Lam11, Kis12], logic and proof theory [Gir89, AJ92, Gir96, GG08].

Bibliography

[Abr96] Samson Abramsky. Semantics of interaction. In Hélène Kirchner, editor, *Trees in Algebra and Programming — CAAP*, volume 1059 of *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg, 1996.

[Abr08] Samson Abramsky. Temperley-Lieb algebra: From knot theory to logic and computation via quantum mechanics. In G. Chen, L. Kauffman, and S. Lomonaco, editors, *Mathematics of Quantum Computing and Technology*, pages 415–458. 2008.

[AHM98] Samson Abramsky, K. Honda, and Guy McCusker. A fully abstract game semantics for general references. *Thirteenth annual IEEE symposium on Logic in Computer science*, pages 334–344, 1998.

[AJ92] Samson Abramsky and Radha Jagadeesan. New foundations for the geometry of interaction. In *Logic in Computer Science, 1992. LICS'92., Proceedings of the Seventh Annual IEEE Symposium on*, pages 211–222. IEEE, 1992.

[AJ94] Samson Abramsky and Radha Jagadeesan. Games and full completeness for multiplicative linear logic. In Rudrapatna Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 652 of *Lecture Notes in Computer Science*, pages 291–301. Springer Berlin/Heidelberg, 1994.

[AJM00] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000.

[AM97] Samson Abramsky and Guy McCusker. Linearity, sharing and state: a fully abstract game semantics for Idealised Algol with active expressions. *Algol-like languages, Progress in theoretical computer science*, 2:297–330, 1997.

[AM99] Samson Abramsky and Guy McCusker. Game semantics. In Ulrich Berger and Helmut Schwichtenberg, editors, *Computational Logic*, volume 165 of *NATO ASI Series*, pages 1–55. Springer Berlin/Heidelberg, 1999.

[Arm83] Mark Anthony Armstrong. *Basic Topology*. Springer-Verlang, 1983.

[Awo06] Steve Awodey. *Category Theory*, volume 49. Oxford University Press, USA, 2006.

[BG91] Stephen Brookes and Shai Geva. Computational comonads and intensional semantics. In M. P. Fourman, P. T. Johnstone, and A. M. Pitts, editors, *Proceedings of the Durham conference on Categories in Computer Science*, Applications of categories in computer science. 1991.

[Bie95] Gavin M. Bierman. What is a categorical model of intuitionistic linear logic? In Mariangiola Dezani-Ciancaglini and Gordon Plotkin, editors, *Second international conference on typed lambda calculi and applications*, volume 902 of *Lecture Notes in Computer science*, pages 78–93. Springer, 1995.

[BS11] John Baez and Mike Stay. Physics, topology, logic and computation: a Rosetta stone. In Bob Coecke, editor, *New Structures in physics*, Lecture Notes in Physics, pages 95–172. 2011.

[BW85] Michael Barr and Charles Wells. Toposes, triples, and theories. *Grundelhren der math. Wissenscheften*, (278), 1985.

[BW99] Michael Barr and Charles Wells. *Category Theory for Computing Science (3rd edition)*. Les Publications CRM, 1999. TAC preprint available online at `ftp://ftp.math.mcgill.ca/barr/ctcs.pdf`.

[CD08] Bob Coecke and Ross Duncan. Interacting quantum observables. In *Automata, Languages and Programming*, pages 298–310. Springer, 2008.

[CH10] Pierre Clairambault and Russ Harmer. Totality in arena games. *Annals of pure and applied logic*, 5(161):673–689, 2010.

[Chu11] Martin David Churchill. *Imperative programs as proofs via game semantics*. PhD thesis, 2011.

[CM10] Ana C. Calderon and Guy McCusker. Understanding game semantics through coherence spaces. *Electronic Notes in Theoretical Computer Science*, 265:231 – 244, 2010. Proceedings of the 26th Conference on the Mathematical Foundations of Programming Semantics (MFPS 2010).

[CP11] B. Coecke and É.O. Paquette. Categories for the practising physicist. In Bob Coecke, editor, *New Structures for Physics*, Lecture Notes in Physics, pages 173–286. Springer, 2011.

[Cro04] Peter R. Cromwell. *Knots and links*. Cambridge University Press, 2004.

[CSC10] Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*, 2010.

[Cur06] Pierre-Louis Curien. Notes on game semantics. *Unpublished notes, viewed February 2013, available at author's website: `http://www.pps.univ-paris-diderot.fr/~curien/Game-semantics.pdf`*, 2006.

[DDK10] Lucas Dixon, Ross Duncan, and Aleks Kissinger. Open graphs and computational reasoning. *arXiv preprint arXiv:1007.3794*, 2010.

[DH01] Vincent Danos and Russell Harmer. The anatomy of innocence. In Laurent Fribourg, editor, *Computer Science Logic*, volume 2142 of *Lecture Notes in Computer Science*, pages 188–202. Springer Berlin Heidelberg, 2001.

[DL04] Vincent Danos and Cosimo Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[EGNO09] P. Etingof, S. Gelaki, D. Nikshych, and V. Ostrik. Tensor categories. *Lecture notes for MIT course 18.769 "Tensor categories"*, 2009. Available online at `http://ocw.mit.edu/courses/mathematics`.

[FY89] Peter J. Freyd and David N. Yetter. Braided compact closed categories with applications to low dimensional topology. *Advances in mathematics*, 77(2):156–182, 1989.

[GG08] Alessio Guglielmi and Tom Gundersen. Normalisation control in deep inference via atomic flows. *Logical methods in computer science*, 2008.

[GG12] Murdoch Gabbay and Dan Ghica. Game semantics in the nominal model. *Electronic Notes in Theoretical Computer Science*, 286:173–189, 2012.

[Ghi09] Dan R. Ghica. Applications of game semantics: From program analysis to hardware synthesis. *Logic in Computer Science, Symposium on*, pages 17–26, 2009.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 1987.

[Gir89] Jean-Yves Girard. Geometry of interaction 1: Interpretation of system F. *Studies in Logic and the Foundations of Mathematics*, 127:221–260, 1989.

[Gir96] Jean-Yves Girard. Proof-nets: the parallel syntax for proof-theory. *Lecture Notes in Pure and Applied Mathematics*, pages 97–124, 1996.

[Gol74] Deborah Louise Goldsmith. Homotopy of braids — in answer to a question of E. Artin. In Raymond F. Dickman Jr. and Peter Fletcher, editors, *Topology Conference*, volume 375 of *Lecture notes in mathematics*, pages 91–96. 1974.

[Hal07] Thomas C. Hales. Jordan's proof of the Jordan curve theorem. *Studies in Logic, Grammar and Rhetoric*, 10(23):45–60, 2007.

[Har00] R. Harmer. *Games and full abstraction for non-deterministic languages*. PhD thesis, University of London, 2000.

[Har04] Russ Harmer. Innocent game semantics. Lecture Notes. Available on author's website: `http://pps.jussieu.fr/~russ/GS.pdf`, 2004.

[Hat02] Allen Hatcher. *Algebraic topology*. Cambridge University Press, 2002.

[HHM07] Russ Harmer, Martin Hyland, and Paul-André Melliès. Categorical combinatorics for innocent strategies. *Logic in Computer Science. LICS 2007. 22nd Annual IEEE Symposium on*, pages 379–388, 2007.

[HM99] Russ Harmer and Guy McCusker. A fully abstract game semantics for finite non-determinism. *Proceedings of the Fourteenth Annual Symposium in Logic in Computer science*, pages 422–430, 1999.

[HO00] Martin Hyland and Luke Ong. On full abstraction for PCF: I, II, and III. *Information and computation*, 163(2):285–408, 2000.

[HS99] Martin Hyland and Andrea Schalk. Abstract games for linear logic extended abstract. *Electronic Notes in Theoretical Computer Science*, 29:127–150, 1999.

[HS02] Martin Hyland and Andrea Schalk. Games on graphs and sequentially realizable functionals. extended abstract. *Proceedings of Seventeenth Annual IEEE Symposium on Logic in Computer Science*, pages 257–264, 2002.

[Hyl97] Martin Hyland. Game semantics. In A. Pitts and P. Dybjer, editors, *Semantics of Logics and Computation*, pages 131–184. Publications of the Newton Institute, Cambridge University, 1997.

[Hyl07] Martin Hyland. Combinatorics of proofs. `http://dpmms.cam.ac.uk/~martin/Research/Slides/licsasl07.pdf`, 2007.

[JS88] André Joyal and Ross Street. Planar diagrams and tensor algebra. *Unpublished manuscript*, 1988.

[JS91] André Joyal and Ross Street. The geometry of tensor calculus I. *Advances in Mathematics*, 88(1):55–112, 1991.

[JS92] A. Joyal and R. Street. The geometry of tensor calculus II. *Unpublished draft, available from Ross Street's website* `http://maths.mq.edu.au/~street/GTCII.pdf`, 1992.

[JS93] André Joyal and Ross Street. Braided tensor categories. *Advances in Mathematics*, 102(1):20–78, 1993.

[JSV96] André Joyal, Ross Street, and Dominic Verity. Traced monoidal categories. *Mathematical proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.

[Kel64] G. M. Kelly. On MacLane's conditions for coherence for natural associativities, commutitivites, etc. *Journal of Algebra*, 1964.

[Kis07] Aleks Kissinger. TikZiT 0.8. Software download `http://sourceforge.net/projects/tikzit/`, 2007. Open source software accessed March 2012.

[Kis12] Aleks Kissinger. *Pictures of Processes: Automated Graph Rewriting for Monoidal Categories and Applications to Quantum Computing.* PhD thesis, 2012.

[Lai01] Jim Laird. A fully abstract game semantics for local exceptions. *Proceedings of the 16th annual IEEE symposium on logic in computer science*, pages 105–114, 2001.

[Lam95] François Lamarche. Games semantics for full propositional linear logic. In *Logic in Computer Science, 1995. LICS'95. Proceedings., Tenth Annual IEEE Symposium on*, pages 464–473, 1995.

[Lam11] J Lambek. Compact monoidal categories from linguistics to physics. In *New Structures for Physics*, Lecture notes in physics, pages 467–487. Springer, 2011.

[Lau04] Olivier Laurent. Polarized games. *Annals of Pure and Applied Logic*, 130(1–3):79–123, 2004.

[Mac63] Sunders MacLane. Natural associativity and commutativity. *The Rice University Studies*, 49(4):28–46, 1963.

[Mac97] Saunders MacLane. *Categories for the Working Mathematician. Second Edition.* Springer, 1997.

[Mel] Paul-André Melliès. Categorical models for linear logic revisited. *To appear, Theoretical Computer Science.*

[Mel06] Paul-André Melliès. Functorial boxes in string diagrams. In Zolátn Ésik, editor, *Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*. Springer BerlinHeidelberg, 2006.

[Mel12a] Paul-André Melliès. Braided notions of dialogue categories. *Submitted manuscript, available on the web page of the author* `http://www.pps.univ-paris-diderot.fr/~mellies/tensorial-logic/6-braided-notions-of-dialogue-categories.pdf`, 2012.

[Mel12b] Paul-André Melliès. Game semantics in string diagrams. In *Proceedings of the 2012 27th Annual IEEE/ACM Symposium on Logic in Computer Science*, pages 481–490. IEEE Computer Society, 2012.

[Mim09] Samuel Mimram. The structure of first-order causality. *24th Annual IEEE symposium on Logics in Computer science*, pages 212–221, 2009.

[MM92] Saunders MacLane and Ieke Moerdijk. *Sheaves in geometry and logic: A first introduction to topos theory.* Springer-Verlang New York, 1992.

[MPW] Guy McCusker, John Power, and Cai Wingfield. A graphical foundation for schedules. *Submitted, Journal of pure and applied algebra.*

[MPW12] Guy McCusker, John Power, and Cai Wingfield. A graphical foundation for schedules. *Electronic Notes in Theoretical Computer Science*, 286:273–289, 2012. Proceedings of the 28th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXVIII).

[Nic94] Hanno Nickau. Hereditarily sequential functionals. In Anil Nerode and Yu. V. Matiyasevich, editors, *Proceedings, Logical Foundations of Computer Science: Logic at St. Petersburg*, Lecture notes in computer science, pages 253–264. 1994.

[Pen71] Roger Penrose. Applications of negative dimensional tensors. In D. J. A. Welsh, editor, *Combinatorial mathematics and its applications*. Academic Press, 1971.

[Pho92] Wesley Phoa. An introduction to fibrations, topos theory, the effective topos and modest sets. LFCS report series ECS-LFCS-92-208, University of Edinburgh, April 1992.

[Pow90] A. John Power. A 2-categorical pasting theorem. *Journal of Algebra*, 129(2):439–445, 1990.

[PW02] A. John Power and Hiroshi Watanabe. Combining a monad and a comonad. *Theoretical Computer Science*, (280):137–162, 2002.

[Rei27] Kurt Reidemeister. Elementare begründung der knotentheorie. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 5, pages 24–32. Springer, 1927.

[Rol76] Dale Rolfsen. *Knots and links*. American Mathematical Society, 1976.

[Sch01] Andrea 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`, 2001.

[Sch04] Andrea Schalk. What is a categorical model of linear logic. *Manuscript, available from* `http://www.cs.man.ac.uk/~schalk/work.html`, 2004.

[See87] R. A. G. Seely. Linear logic, *-autonomous categories and cofree algebras. *Proceedings, AMS conference on categories in computer science and logic*, 1987.

[Sel11] Peter Selinger. A survey of graphical languages for monoidal categories. Lecture Notes in Physics, pages 289–355. Springer Berlin/Heidelberg, 2011.

[Shu94] Mei Chee Shum. Tortile tensor categories. *Journal of Pure and Applied Algebra*, 93(1):57–110, 1994.

[Shu08] Michael Shulman. Framed bicategories and monoidal fibrations. *Theory and Applications of Categories*, 20(18):650–738, 2008.

[SS78] Lynn Steen and J. Arthur Seebach. *Counterexamples in topology, second edition*. Springer-Verlag (New York), 1978.

[SS86] Stephen Schanuel and Ross Street. The free adjunction. *Cahiers de topologie et géométrie différentielle catégoriques*, 27(1):81–83, 1986.

[Str76] Ross Street. Limits indexed by category-valued 2-functors. *Journal of Pure and Applied Algebra*, 1976.

[Tze08] Nikos Tzevelekos. *Nominal game semantics*. PhD thesis, 2008.

[Veb05] Oswald Veblen. Theory on plane curves in non-metrical analysis situs. *Transactions of the American Mathematical Society*, 6(1):83–98, 1905.

Appendices

A.1 Geometric methods

We take the following standard definitions and facts of topology from [Arm83].

Definition 227. An **open cover** of a topological space X is a collection of open sets $\{U_i \mid i \in I\}$ with $X \subseteq \bigcup_{i \in I} U_i$.

Definition 228. A topological space is said to be **compact** if every open cover has a finite sub-cover.

Lemma 229. *Any closed subset of a compact space is compact.*

Lemma 230. *The continuous image of a compact space is compact.*

Definition 231. For manifolds S, A and embeddings g, h of S in A, an **ambient isotopy** is a continuous map $F : A \times [0,1] \to A$ such that $F(-,0) = \mathrm{id}_A$, $F(-,t)$ is a homeomorphism $A \to A$ for each $t \in [0,1]$, and $F(-,1) \circ g = h$.

In the case where $A = \mathbb{R}^2$, we call F a **planar isotopy**.

A.2 Monoidal categories

We assume basic knowledge of category theory, such as definitions of *categories*, *functors* and *natural transformations*. Detailed discussion of these can be found in many sources, such as Mac Lane's textbook [Mac97] or Awodey's more modern textbook [Awo06]. We refer the reader to such sources rather than reproducing this elementary material here. Monoidal categories are also dealt with in [Mac97] and extensively in the literature, but it seems worth presenting in some detail here, since Chapter 2 relies on it so heavily.

As seems to be both conventional and clarifying, when discussing categories we identify objects X and their identity morphisms id_X.

A.2.1 Monoidal categories, monoidal functors and $\mathcal{MonCat}$

The following definitions are standard and can be found in, for example, [JS91, Mac97, Sel11].

Definition 232. A **monoidal category** or **tensor category** is a category $\mathscr{V}$ together with the following additional data:

- A functor $\otimes : \mathscr{V} \times \mathscr{V} \to \mathscr{V}$ (written infix) called **tensor**.
- A distinguished object $I \in \mathscr{V}$ called the **unit object**.
- Natural isomorphisms

$$a : (- \otimes -) \otimes - \xRightarrow{\sim} - \otimes (- \otimes -)$$

$$a_{X,Y,Z} : (X \otimes Y) \otimes Z \xrightarrow{\sim} X \otimes (Y \otimes Z)$$

  called the **associator**,

$$l : I \otimes - \xRightarrow{\sim} \mathrm{id}_{\mathscr{V}}$$

$$r : - \otimes I \xRightarrow{\sim} \mathrm{id}_{\mathscr{V}}$$

  called the **left** and **right unit** respectively.

These must satisfy the following coherence axioms:

$$\begin{array}{ccc}
 & (W \otimes X) \otimes (Y \otimes Z) & \\
{\scriptstyle a_{W\otimes X,Y,Z}}\nearrow & & \searrow{\scriptstyle a_{W,X,Y\otimes Z}} \\
((W \otimes X) \otimes Y) \otimes Z & & W \otimes (X \otimes (Y \otimes Z)) \\
\downarrow{\scriptstyle a_{W,X,Y} \otimes Z} & & \uparrow{\scriptstyle W \otimes a_{X,Y,Z}} \\
(W \otimes (X \otimes Y)) \otimes Z & \xrightarrow[a_{W,X\otimes Y,Z}]{} & W \otimes ((X \otimes Y) \otimes Z)
\end{array} \tag{MC1}$$

$$\begin{array}{ccc}
(X \otimes I) \otimes Y & \xrightarrow{a} & X \otimes (I \otimes Y) \\
 & {\scriptstyle r \otimes Y}\searrow \quad \swarrow{\scriptstyle X \otimes l} & \\
 & X \otimes Y &
\end{array} \tag{MC2}$$

(known as the "pentagon axiom" and "triangle axiom" respectively) commute as diagrams of natural transformations.

In the case where a, l and r are identity natural transformations, we call $\mathscr{V}$ a **strict monoidal category**.

It should be noted that there are many possible axiomatisations of monoidal categories, and that the one given here (taken from [JS93]) is just one such. For another possible

equivalent definition, see [EGNO09]. The original definition in [Mac97] has $l_I = r_I$ as an axiom, but this is actually redundant, being derivable from the other axioms, as is shown in [JS93].

Example 233.

- $(\mathcal{Set}, \times, \{*\})$ is a non-strict monoidal category. Its non-strictness is manifest in the fact that, for example, $\{1,2,3\} \neq \{(1,*),(2,*),(3,*)\}$, which we get from application of the right unit.
- Let $\mathscr{C}$ be a category with finite products and terminal object 1. $(\mathscr{C}, \times, 1)$ is a monoidal category called a **cartesian monoidal category**. Note that the categorical product is different from the tensor product: a category either has products or it doesn't, whereas $\otimes$ is something which we additionally impose.
- If $\mathscr{C}$ is a category then $([\mathscr{C},\mathscr{C}], \circ, \mathrm{id}_{\mathscr{C}})$ is a strict monoidal category. Its strictness comes from the fact that functors compose strictly associatively, and the identity functor is a strict identity.
- The category $\mathcal{Rel}$ of sets and relations has sets as objects and relations as morphisms; a morphism $X \to Y$ is a relation $R \subseteq X \times Y$. There is a natural monoidal structure on $\mathcal{Rel}$ which is the ordinary product of relations — $X \otimes Y$ is $X \times Y$ and for $R : X \to Y$ and $S : Z \to W$, $R \otimes S : X \otimes Z \to Y \otimes W$ is the relation so that

  $$(x,z)(R \otimes S)(y,w) \iff xRy \text{ and } zSw$$

  The tensor unit is the singleton $\{*\}$. It is non-strict since the Cartesian product of sets is not strict.

The following theorem of Mac Lane, which we prove later, demonstrates that the coherence axioms MC1 and MC2 are sufficient to ensure that $\otimes$ behaves like the product structure we intend to capture.

Theorem 234 ([Mac97])**.** *Let $\mathscr{V}$ be a monoidal category. Let $\mathbf{X}$ and $\mathbf{X}'$ be two identically ordered but arbitrarily bracketed tensor products of objects from $\mathscr{V}$ with arbitrarily inserted Is. For example,*

$$\mathbf{X} = ((X_1 \otimes X_2) \otimes ((I \otimes (X_3 \otimes I)) \otimes I) \otimes I) \otimes (X_4 \otimes X_5)$$

$$\mathbf{X}' = (((I \otimes ((I \otimes X_1) \otimes (I \otimes X_2))) \otimes X_3) \otimes X_4) \otimes I) \otimes I$$

Let f and g be two isomorphisms

$$\mathbf{X} \underset{g}{\overset{f}{\rightrightarrows}} \mathbf{X}'$$

formed by composing components of a, l and r and tensoring with identity arrows. Then $f = g$.

This famous theorem is often summarised as "in a monoidal category, all diagrams commute that should".

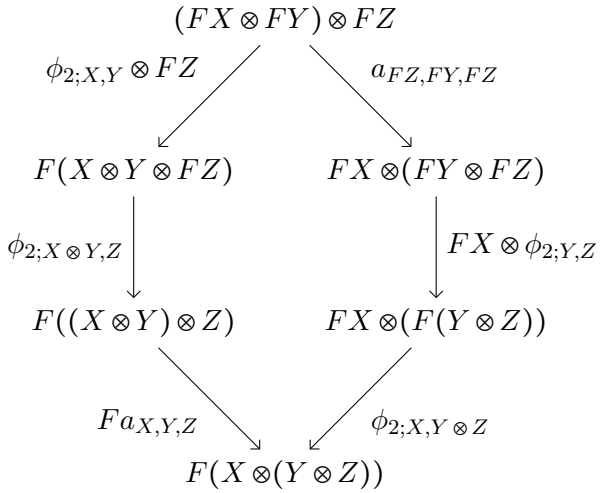
Definition 235. For monoidal categories $\mathscr{V}$ and $\mathscr{W}$, a **monoidal functor** is a functor $F : \mathscr{V} \to \mathscr{W}$, together with natural transformation

$$\phi_{2;X,Y} : FX \otimes FY \to F(X \otimes Y)$$

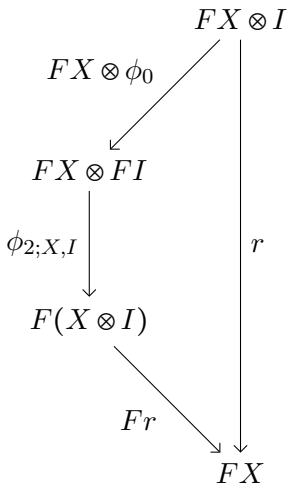
and distinguished morphism

$$\phi_0 : I \to FI$$

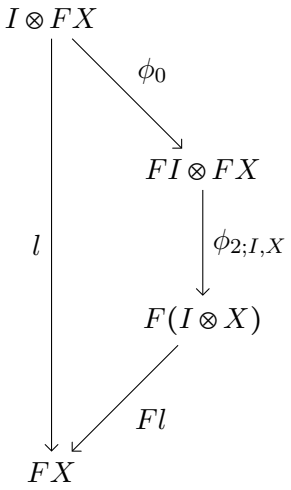
such that the following diagrams of natural transformations commute:



(MF1)



(MF2)



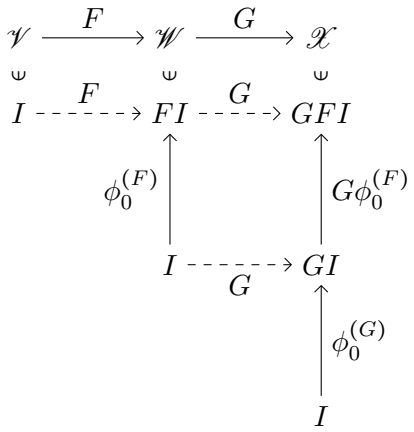
(MF3)

If ϕ_0 and components of ϕ_2 are isomorphisms then F is called a **strong monoidal functor**. If furthermore ϕ_0 and ϕ_2 are identities then F is called a **strict monoidal functor**.

Definition 236. Monoidal functors $F : \mathscr{V} \to \mathscr{W}$ together with $\phi_0^{(F)}$ and $\phi_2^{(F)}$, and $G : \mathscr{W} \to \mathscr{X}$ together with $\phi_0^{(G)}$ and $\phi_2^{(G)}$ may be **composed** to give a monoidal functor $G \circ F : \mathscr{V} \to \mathscr{X}$ (also written GF) together with $\phi_0^{(GF)}$ defined by

$$\phi_0^{(GF)} = G\phi_0^{(F)} \circ \phi_0^{(G)}$$

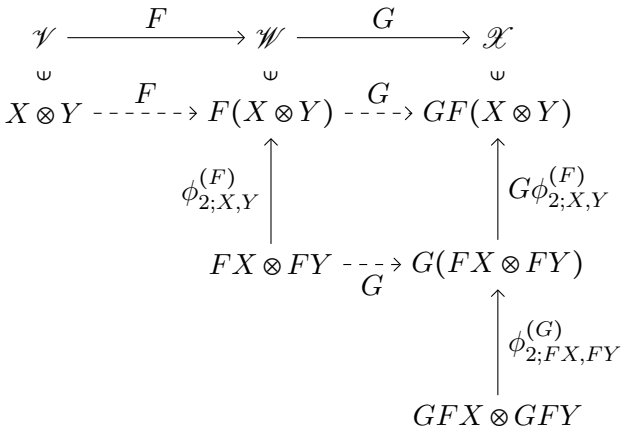
as in



and with $\phi_2^{(GF)}$ defined by

$$\phi_{2;X,Y}^{(GF)} = G\phi_{2;X,Y}^{(F)} \circ \phi_{2;FX,FY}^{(G)}$$

as in



Observe that the composition of monoidal functors is associative and that the monoidal functor

$$\begin{aligned} \mathrm{id}_{\mathscr{V}} &: \mathscr{V} \to \mathscr{V} \\ \phi_0^{(\mathrm{id}_{\mathscr{V}})} &= \mathrm{id}_I \\ \phi_{2;X,Y}^{(\mathrm{id}_{\mathscr{V}})} &= \mathrm{id}_{X \otimes Y} \end{aligned}$$

is the identity of monoidal functor composition.

Definition 237. The category of monoidal categories and monoidal functors is called $\mathcal{MonCat}$.

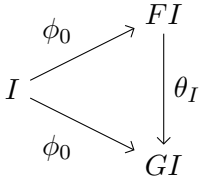
A.2.2 Monoidal natural transformations and monoidal functor categories

There are also monoidal analogues to natural transformations.

Definition 238. Given monoidal functors $F, G : \mathscr{V} \to \mathscr{W}$, a **monoidal natural transformation** is a natural transformation $\theta : F \Longrightarrow G : \mathscr{V} \longrightarrow \mathscr{W}$ of the underlying functors, such that

$$\begin{array}{ccc} FX \otimes FY & \xrightarrow{\phi_2} & F(X \otimes Y) \\ \Big\downarrow{\scriptstyle \theta \otimes \theta} & & \Big\downarrow{\scriptstyle \theta} \\ GX \otimes GY & \xrightarrow[\phi_2]{} & G(X \otimes Y) \end{array} \tag{MN1}$$

commutes as a diagram of natural transformations and such that



(MN2)

also commutes.

Definition 239. Given monoidal categories $\mathscr{V}$ and $\mathscr{W}$, we can for the category of strong monoidal functors $\mathscr{V} \to \mathscr{W}$. Its objects are strong monoidal functors $F : \mathscr{V} \to \mathscr{W}$, and a morphism $F \to G$ is a monoidal natural transformation $\theta : F \Rightarrow \mathscr{G}$. We call this category $[\mathscr{V}, \mathscr{W}]_{\mathcal{S}t}$.

A.2.3 A generalisation of Cayley's Theorem for monoids

The proof of Theorem 234 will rely on the following property of monoidal categories.

Proposition 240. *Every monoidal category is equivalent to a strict monoidal category.*

We may better understand this statement and its proof if we notice that it is a categorical generalisation of the more familiar Cayley's Theorem for monoids.

Theorem 241 (Cayley's theorem for monoids)**.** *Let $(M, \cdot, e)$ be a monoid. There is an injective monoid homomorphism*

$$\begin{aligned} (M, \cdot, e) &\longrightarrow (M^M, \circ, \mathrm{id}_M) \\ m &\longmapsto (f_m : n \mapsto mn) \\ e &\longmapsto \mathrm{id}_M \end{aligned}$$

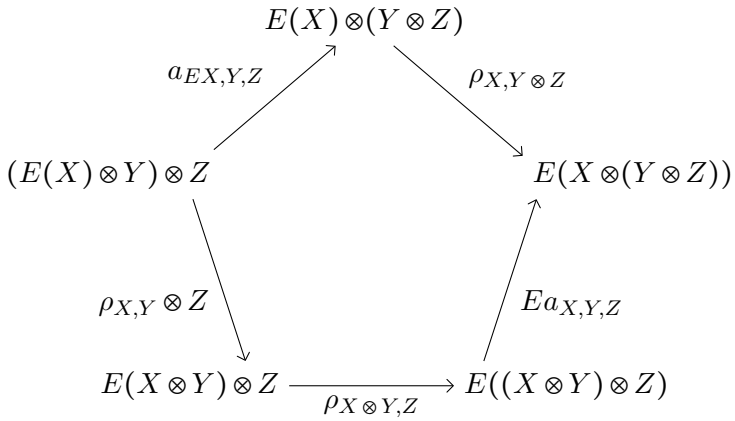
Keeping this in mind, we can proceed with the proof of Proposition 240. This proof follows the proof in [JS93], with further details fleshed-out as in [EGNO09], and follows a procedure similar to that in a proof of Cayley's theorem.

Proof of Proposition 240. Let $\mathscr{V}$ be a monoidal category. We'll construct a strict monoidal category, $e(\mathscr{V})$, which we'll then demonstrate to be equivalent to $\mathscr{V}$.

An object of $e(\mathscr{V})$ is a pair (E, ρ), with $E : \mathscr{V} \to \mathscr{V}$ a functor and

$$\rho_{X,Y} : (EX) \otimes Y \tilde{\to} E(X \otimes Y)$$

a natural isomorphism satisfying the following coherence axiom



(A.1)

This diagram can be read as "E commutes with $\otimes$ on the right via ρ".

An arrow $\phi : (E, \rho) \to (F, \sigma)$ is a natural transformation $\phi : E \Rightarrow F$ such that the following commutes as a diagram of natural transformations

$$\begin{array}{ccc} (EX)\otimes Y & \xrightarrow{\rho} & E(X\otimes Y) \\ \downarrow{\scriptstyle \phi\otimes Y} & & \downarrow{\scriptstyle \phi} \\ (FX)\otimes Y & \xrightarrow[\rho]{} & F(X\otimes Y) \end{array}$$

Composition of arrows is vertical composition of natural transformations.

The tensor product is defined to be

$$(E, \rho)\otimes(F, \sigma) = (EF, E\sigma_{X,Y} \circ \rho_{FX,Y})$$

where $E\sigma_{X,Y} \circ \rho_{FX,Y}$ is the composition

$$(EFX)\otimes Y \xrightarrow{\rho_{FX,Y}} E((FX)\otimes Y) \xrightarrow{E\sigma_{X,Y}} EF(X\otimes Y)$$

with the unit object being the pair $(\mathrm{id}_{\mathscr{V}}, =)$.

By observing the appropriate natural isomorphisms we can see that $e(\mathscr{V})$ is strict, since compositions of functors and natural transformations is strict. For example, with

associativity:

$$\begin{aligned}\big[(E,\rho)\otimes(F,\sigma)\big]\otimes(G,\tau) =& \big(EF, E\sigma_{X,Y}\circ\rho_{FX,Y}\big)\otimes(G,\tau)\\ =& \big((EF)G, EF\tau_{X,Y}\circ(E\sigma_{GX,Y}\circ\rho_{FGX,Y})\big)\\ =& \big((EF)G, (EF\tau_{X,Y}\circ E\sigma_{GX,Y})\circ\rho_{FGX,Y}\big)\\ =& \big(E(FG), E(F\tau_{X,Y}\circ\sigma_{GX,Y})\circ\rho_{FGX,Y}\big)\\ =& (E,\rho)\otimes\big(FG, F\tau_{X,Y}\circ\sigma_{GX,Y}\big)\\ =& (E,\rho)\otimes\big[(F,\sigma)\otimes(G,\tau)\big]\end{aligned}$$

The left and right unit isomorphisms are similarly identities.

Now we'll describe an equivalence $e(\mathscr{V})\cong\mathscr{V}$. Construct a monoidal functor

$$\begin{aligned} L:\mathscr{V} &\longrightarrow e(\mathscr{V})\\ X &\longmapsto (X\otimes -, a_{X,-,-})\\ (f:X\to Y) &\longmapsto (f\otimes -: (X\otimes -, a_{X,-,-})\to(Y\otimes -, a_{Y,-,-})\end{aligned}$$

The monoidal structure of L is given by

$$\begin{array}{ccc} \phi_0^{(L)} : LI^{(\mathscr{V})} & \longrightarrow & (\mathrm{id}_{\mathscr{V}}, =)\\ \| \quad \| & & \|\\ l : (I^{(\mathscr{V})}\otimes -, a_{I,-,-}) & \longrightarrow & I^{(e(\mathscr{V}))}\end{array}$$

and

$$\begin{array}{ccc} \phi_2^{(L)} : LX\otimes LY & \longrightarrow & L(X\otimes Y)\\ \| \quad \| & & \|\\ a_{X,Y,-} : \big(X\otimes(Y\otimes -), (X\otimes a_{X,-,-})\circ a_{X,Y\otimes -,-}\big) & \longrightarrow & \big((X\otimes Y)\otimes -, a_{X\otimes Y,-,-}\big)\end{array}$$

(with superscripts to disambiguate).

We'll show that this functor is full, faithful and essentially surjective and thus (by a theorem of category theory) an equivalence.

We first prove the claim that L is essentially surjective; that is, every object (E,ρ) in $e(\mathscr{V})$ is isomorphic to one of the form LX for some $X\in\mathscr{V}$. Let (E,ρ) be an object in

$e(\mathscr{V})$. There is an isomorphism

$$LEI \xrightarrow{El \circ \rho_{I,-}} (E, \rho)$$

$$(EI) \otimes - \xRightarrow[\rho_{I,-}]{} E(I \otimes -) \xRightarrow[El]{} E$$

in $e(\mathscr{V})$. We can now use the coherence (A.1) to show that E, ρ and a commute appropriately.

Next we show that L is full; that is, surjective on hom sets. Let $\theta : LX \to LY$ be an arrow in $e(\mathscr{V})$. We define an arrow $f : X \to Y$ as

$$f : X \xrightarrow{r_X} X \otimes I \xrightarrow{\theta_X} Y \otimes I \xrightarrow{r_Y} Y$$

Now we show $\theta = f \otimes - = Lf$. Consider the following diagram of morphisms

$$\begin{array}{ccccccc} X \otimes Z & \xrightarrow{r_X \otimes Z} & (X \otimes I) \otimes Z & \xrightarrow{a} & X \otimes (I \otimes Z) & \xrightarrow{X \otimes l_Z} & X \otimes Z \\ \downarrow{\scriptstyle f \otimes Z} & (\mathrm{A}) & \downarrow{\scriptstyle \theta_I \otimes Z} & (\mathrm{B}) & \downarrow{\scriptstyle \theta_{I \otimes Z}} & (\mathrm{C}) & \downarrow{\scriptstyle \theta_Z} \\ Y \otimes Z & \xrightarrow[r_Y \otimes Z]{} & (Y \otimes I) \otimes Z & \xrightarrow[a]{} & Y \otimes (I \otimes Z) & \xrightarrow[Y \otimes l_Z]{} & Y \otimes Z \end{array} \tag{A.2}$$

- (A) commutes by the definition of f because r is an isomorphism.
- (B) commutes by the definition of θ being an arrow in $e(\mathscr{V})$.
- (C) commutes by the naturality of θ.

Therefore the perimeter of (A.2) commutes. The top and bottom long edges commute because they are secretly triangle axioms, so therefore the left and right edges are equal. In other words, $f \otimes Z = \theta_Z$ for arbitrary $Z \in \mathscr{V}$ and so $\theta = f \otimes - = Lf$.

Finally that L is faithful; that is, injective on hom-sets. Suppose that arrows f and g in $\mathscr{V}$ are such that $Lf = Lg$. Then

$$\begin{aligned} & Lf = Lg \\ \implies & f \otimes - = g \otimes - \\ \implies & f \otimes I = g \otimes I \\ \implies & f = g \end{aligned}$$

Hence L is full and faithful and essentially surjective and so is an equivalence of categories $\mathscr{V} \cong e(\mathscr{V})$. □

Now we can give the following sketch of a proof for Theorem 234.

Sketch proof for Theorem 234. Draw a diagram of morphisms in $\mathscr{V}$ with f and g as the two legs and with all composed components separate. Apply L (as defined in Proposition 240) to get a diagram in $e(\mathscr{V})$.

The monoidal structure of L gives us rectangles as in (MF1), (MF2) and (MF3) which we may attach to the edges La, Ll and Lr, so that we reach a "prism" with our diagram in $e(V)$ as one face and an identity diagram as the other face, and commutative diagrams round the edges. Therefore the final face commutes, so our original diagram commutes. □

A.3 Schedules and interleaving graphs

A.3.1 An example of suitability

Figure 69 works through the procedure of demonstrating the suitability of a 5-interleaving graph for a binary $(\otimes, \multimap)$-word of length 5.

A.3.2 An example of unfolding

Figure 70 takes a 3-interleaving graph suitable for $A \multimap (X \otimes C)$ whose X-nodes are labelled by (truncations of) a $\multimap$ schedule labelled in $X_1 \multimap X_2$, and works through the procedure of unfolding the X-nodes to leave a 4-interleaving graph suitable for $A \multimap (X_1 \multimap X_2) \otimes C)$.

A.3.3 Bifuctoriality of $\otimes$

Lemma 242. $\otimes$ *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

$$(\sigma \otimes \sigma') \| (\tau \otimes \tau') = (\sigma \| \tau) \otimes (\sigma' \| \tau') \tag{A.3}$$

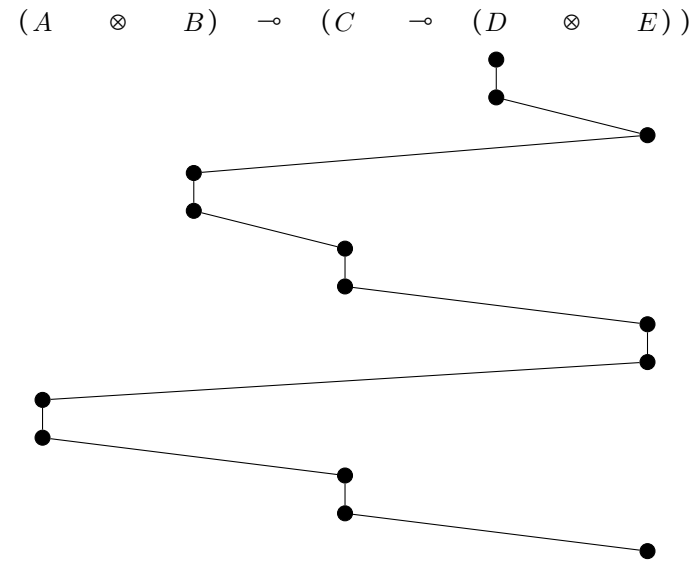
We will illustrate the argument with a running example in Figure 71.

Since both sides of (A.3) are sets of plays, we will proceed by showing set inclusions in both directions.

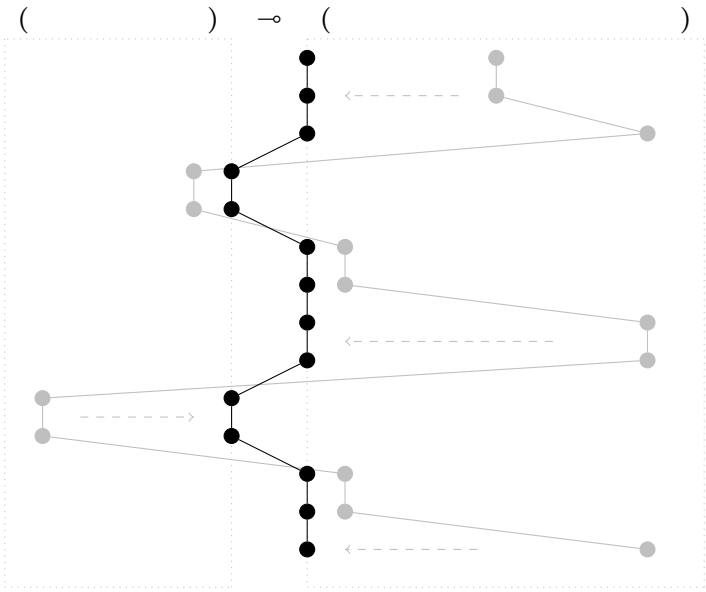
($\subseteq$): Consider

$$(\sigma \otimes \sigma') \| (\tau \otimes \tau') : (A \otimes A') \multimap (C \otimes C') \tag{A.4}$$

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.4) are given

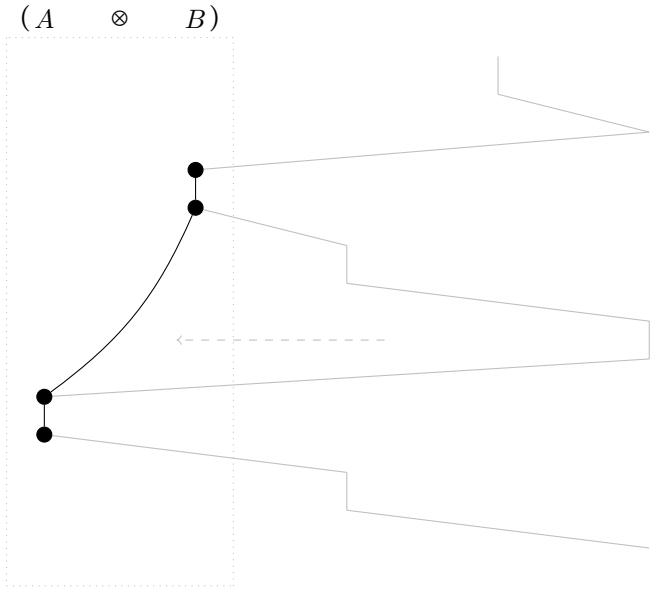


(a) The 5-interleaving graph from Figure 47.

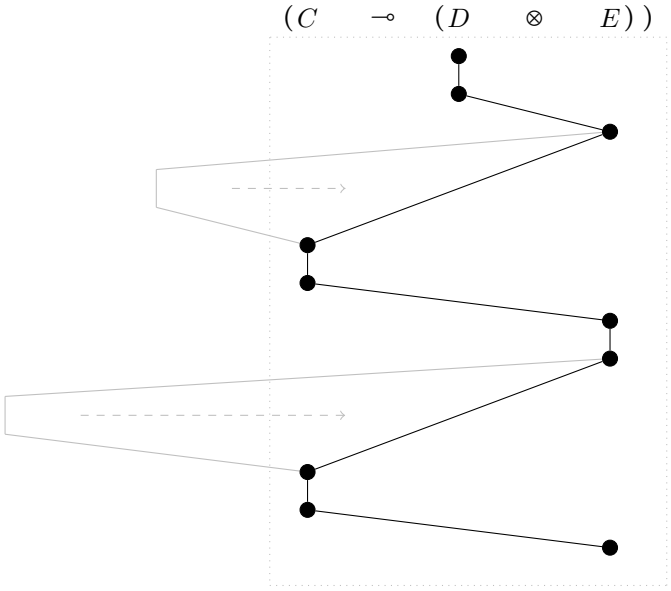


(b) Segregation over the main connective yields a schedule for that connective.

Figure 69: Demonstrating the suitability of the 5-interleaving graph of Figure 47 for the word $(A \otimes B) \multimap (C \multimap (D \otimes E))$.

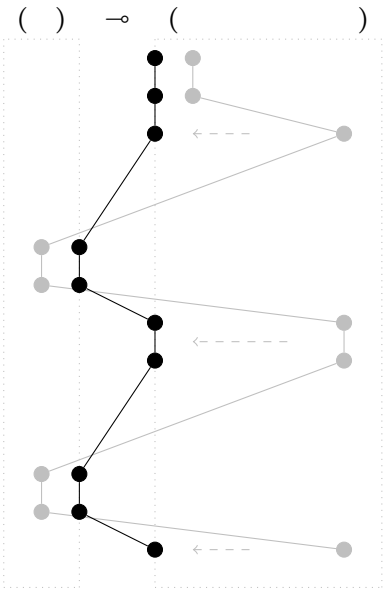


(c) Restriction to a bracketed subword yields an interleaving graph suitable for that connective. In this case, a ⊗-schedule.

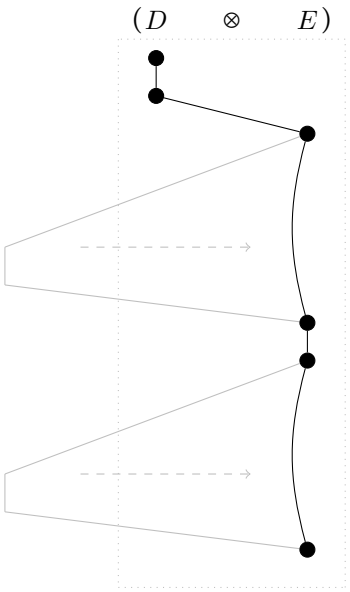


(d) Restriction to a bracketed subword to recursively check for suitability.

Figure 69: (Continued.)



(e) Segregation over the main connective yields a schedule for that connective.



(f) Restriction to a bracketed subword yields an interleaving graph suitable for that connective.

Figure 69: (Continued.)

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 71(a), and Z and $\check{Z}$are shown in Figure 71(b).

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

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

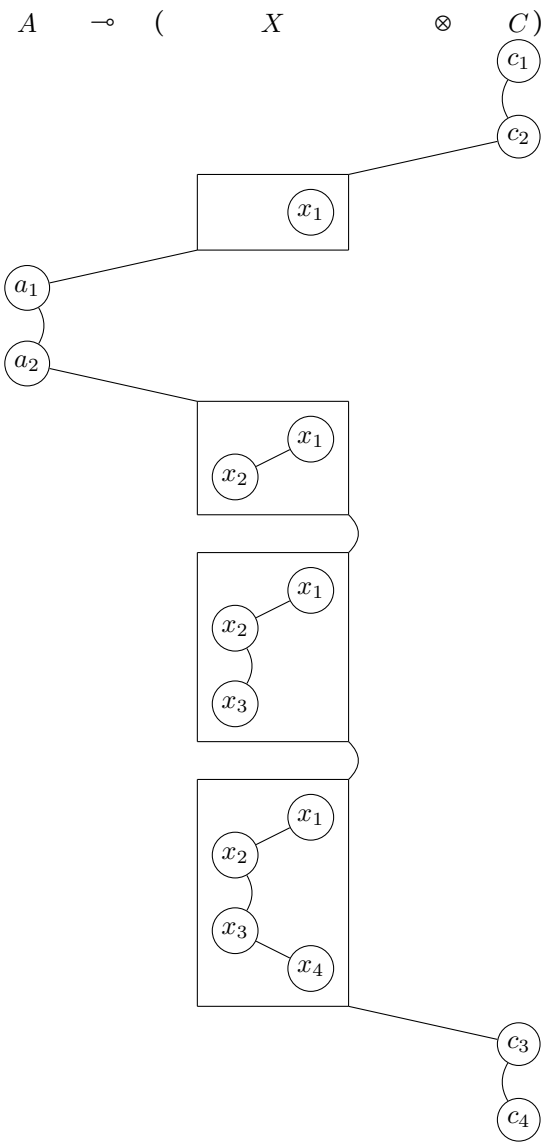
So a play in (A.4) is given by a 4-interleaving graph

$$(S \otimes_{\check{Z}} S') \| (T \otimes_Z T') \tag{A.5}$$

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

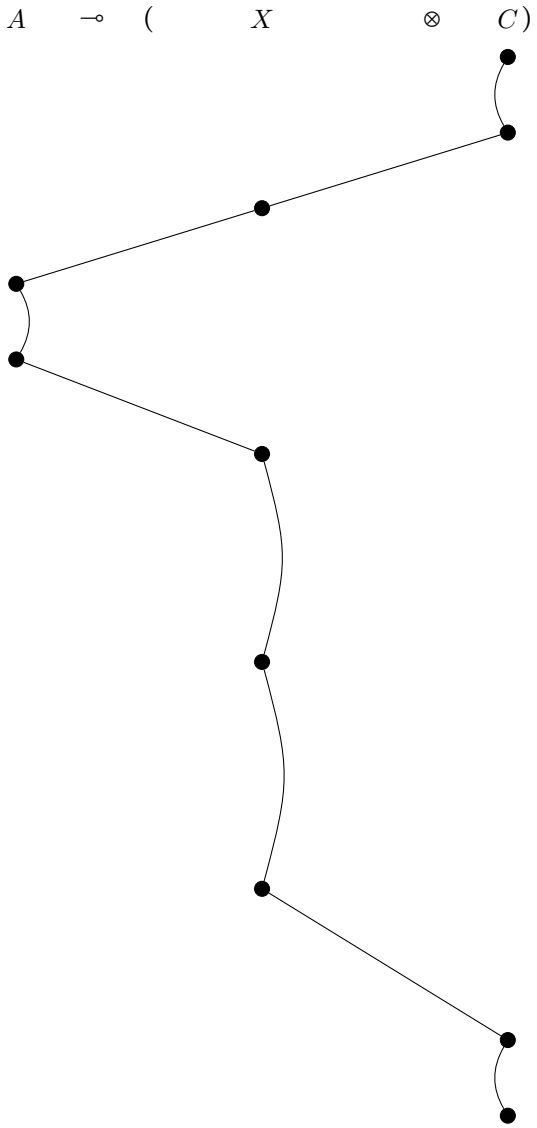
$T \otimes_Z T'$ and $S \otimes_{\check{Z}} S'$ are shown by example in Figures 71(c) and 71(d).

Composing (A.5) hides activity in $B \otimes B'$, with play continuing in whichever (left or right) component we are in:

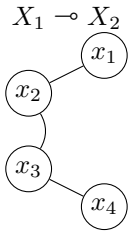


(a) A 3-interleaving graph labelled in a game $A \multimap (X \otimes C)$. X is a game of the form $X_1 \multimap X_2$ and as such, the X-nodes are labelled with $\multimap$-schedules labelled in $X_1 \multimap X_2$.

Figure 70: Unfolding a 3-interleaving graph labelled in $A \multimap (X \otimes C)$ with $X = X_1 \multimap X_2$ to get a 4-interleaving graph labelled in $A \multimap (X_1 \multimap X_2) \otimes C)$.

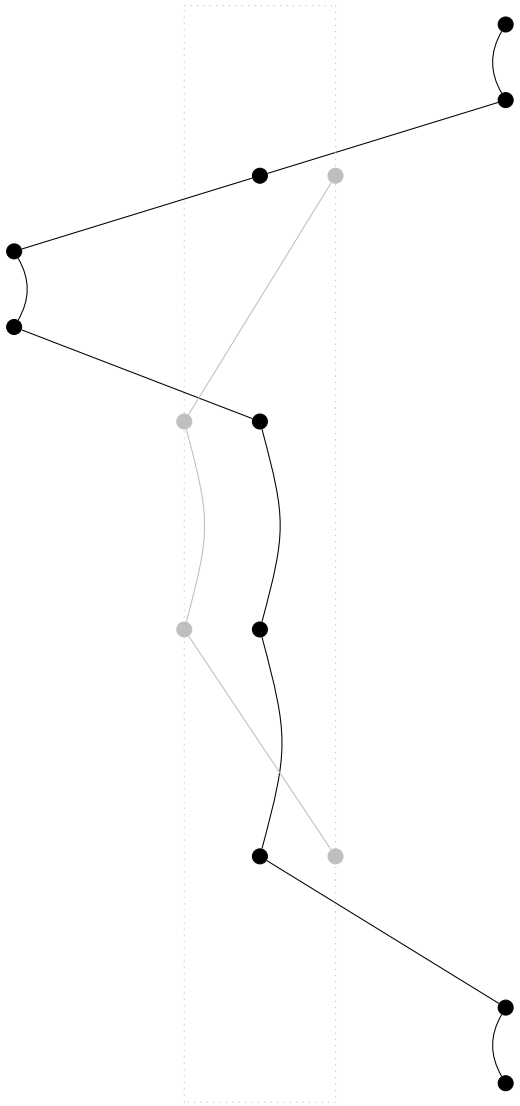


(b) The underlying 3-interleaving graph of the one shown in Figure 70(a).



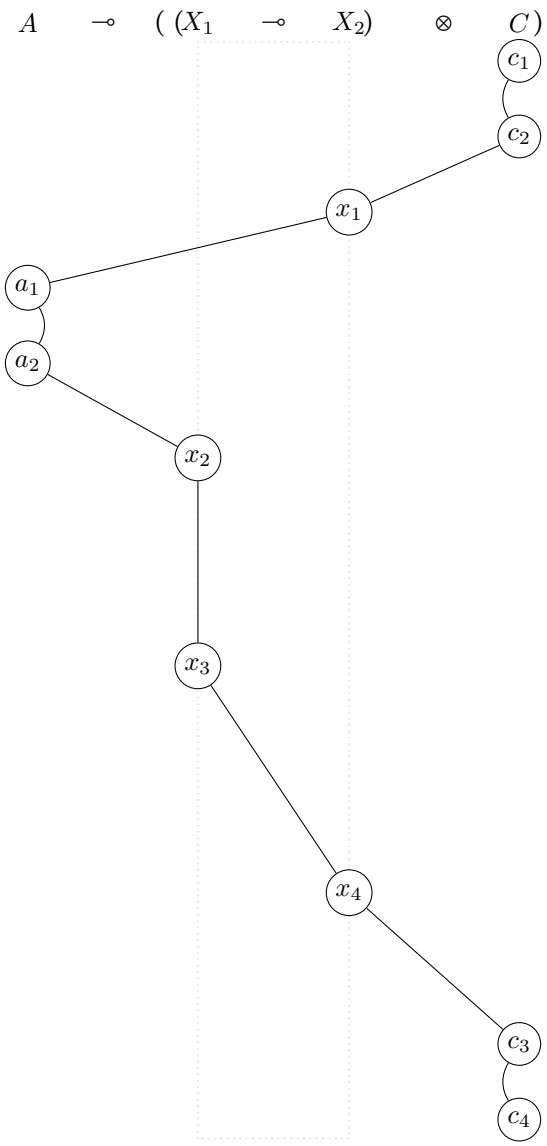
(c) The label on the final X-node of the 3-interleaving graph in Figure 70(a). It determines the labels on all previous X-nodes by truncation.

Figure 70: (Continued.)



(d) The $\multimap$-schedule from Figure 70(c) superimposed over the 3-interleaving graph from Figure 70(b), deformed so that the corresponding nodes have the same vertical positions.

Figure 70: (Continued.)



(e) The 3-interleaving graph from Figure 70(b), now deformed (as a progressive plane graph) so it's X-nodes coincide with the corresponding nodes of the $\multimap$-schedule from Figure 70(d), and with labels taken from the original 3-interleaving graph and the $\multimap$-schedule from Figure 70(c).

Figure 70: (Continued.)

- 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

$$(\sigma\otimes\sigma')\|(\tau\otimes\tau')\subseteq(\sigma\|\tau)\otimes(\sigma'\|\tau') \tag{A.6}$$

In our example, the composition diagram (A.5) is shown in Figure 71(e) with the final composite shown in Figure 71(f).

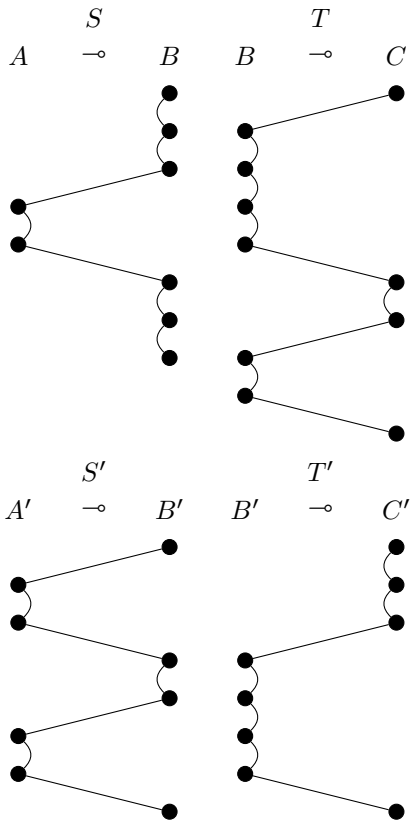
($\supseteq$): A play in $(\sigma\|\tau)\otimes(\sigma'\|\tau')$ is given by a 4-interleaving graph $(S\|T)\otimes_Z(S'\|T')$ for some appropriate $\multimap$-schedules $S\|T$ and $S'\|T'$, and $\otimes$-schedule Z. Examples are given in Figure 71(g).

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 49, and can be seen by example in Figure 71(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 71(i). Finally there's a uniquely determined $\otimes$-schedule interleaving positions in $B\otimes B'$ and it is $\check{Z}$ induced by $T\otimes_Z T'$.

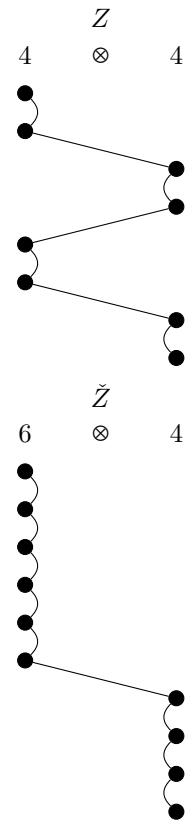
Hence

$$(\sigma\otimes\sigma')\|(\tau\otimes\tau')\supseteq(\sigma\|\tau)\otimes(\sigma'\|\tau') \tag{A.7}$$

And (A.6) and (A.7) together give (A.3), as required. □

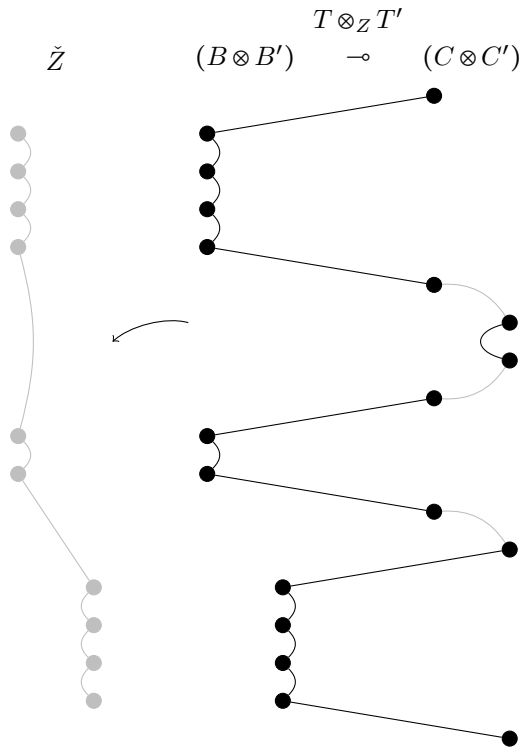


(a) $\multimap$-schedules S, T, S' and T'.



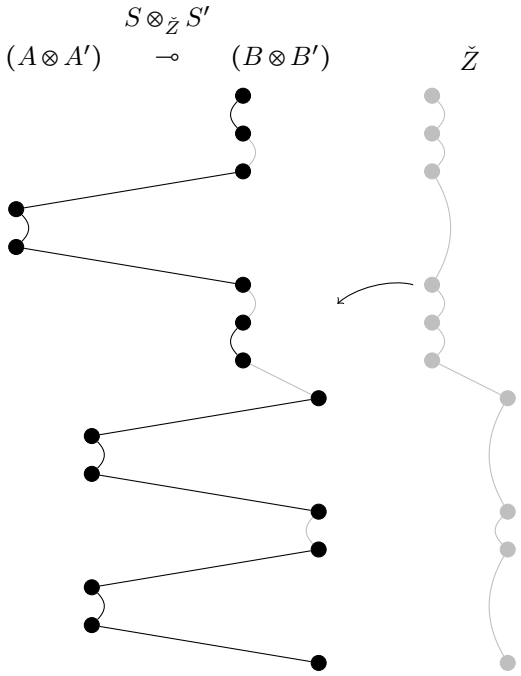
(b) The $\otimes$-schedules Z and the $\otimes$-schedule $\check{Z}$ induced by $T \otimes_Z T'$.

Figure 71: Bifunctoriality.



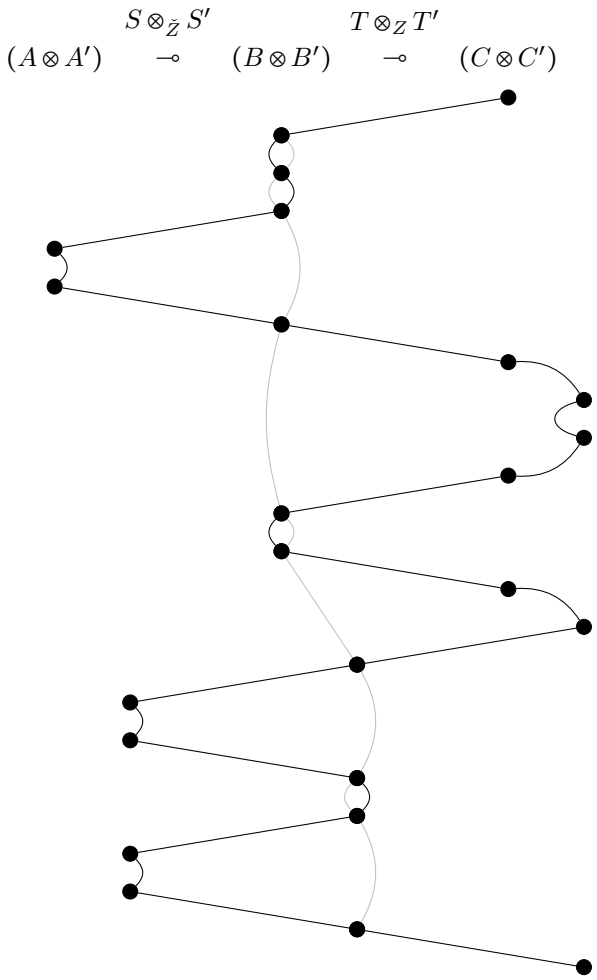
(c) The construction $T \otimes_Z T'$ and the induced $\check{Z}$.

Figure 71: (Continued.)



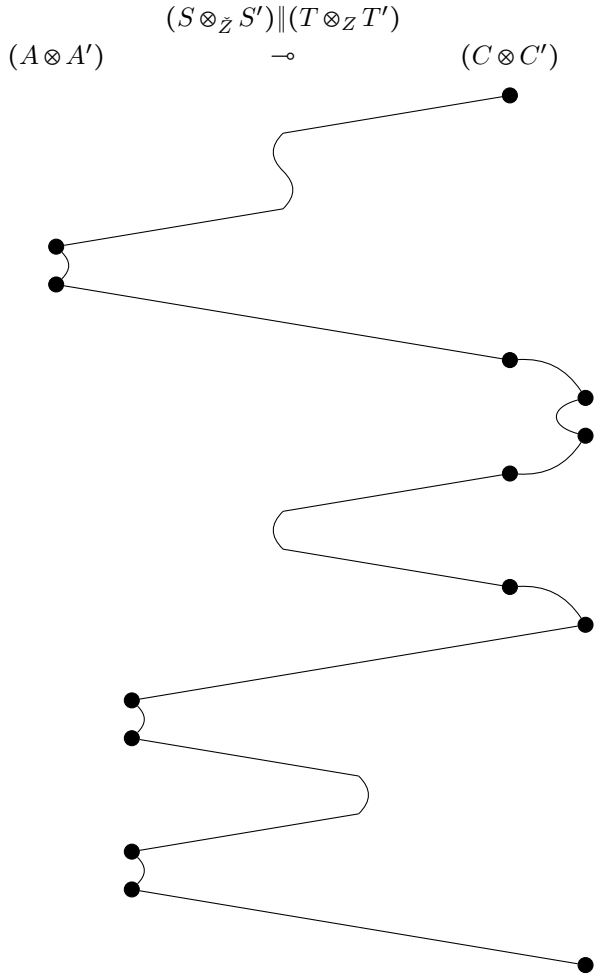
(d) The construction $S \otimes_{\check{Z}} S'$.

Figure 71: (Continued.)

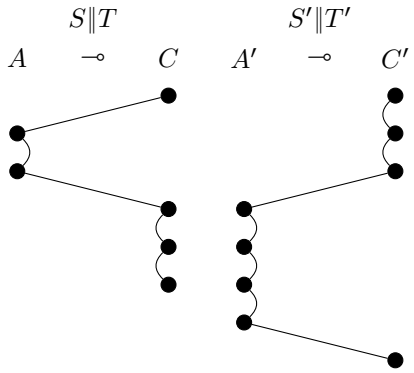


(e) Composition diagram for $(S \otimes_{\tilde{Z}} S') \| (T \otimes_Z T')$.

Figure 71: (Continued.)

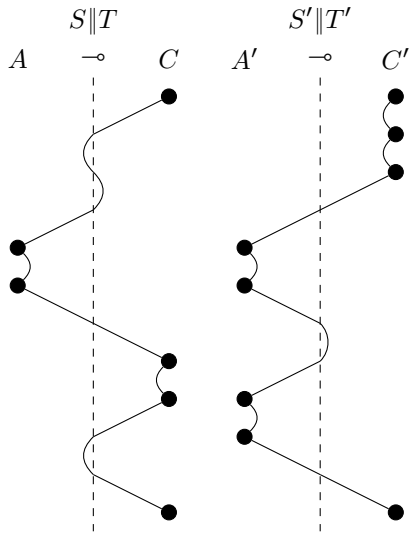


(f) 4-interleaving graph $(S \otimes_{\tilde{Z}} S') \| (T \otimes_Z T')$.



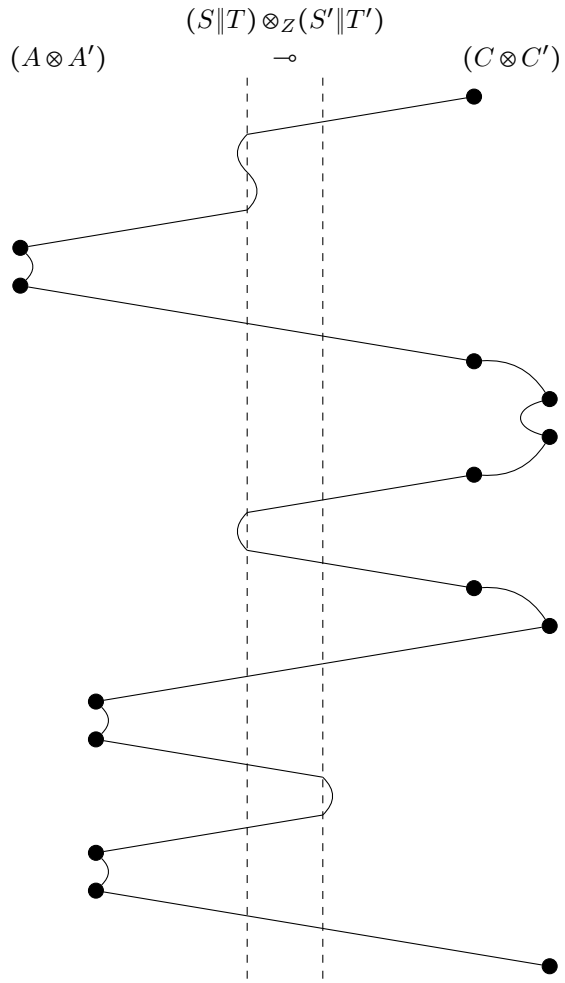
(g) Composites $S \| T$ and $S' \| T'$.

Figure 71: (Continued.)



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

Figure 71: (Continued.)



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

Figure 71: (Continued.)

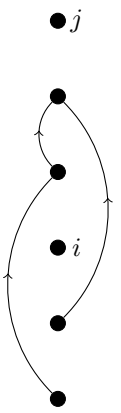


Figure 72: It is not immediately clear how to add an upwardly progressive edge $i \to j$ in this heap graph without edge crossing.

Remark 243. In the proof of Lemma 242, 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 71(e) and 71(f).

A.4 Pointers

A.4.1 Heap graphs in standard configuration

It may appear that being in standard configuration would prevent the image of a heap grpah in the plane being an embedding. Foe example, it may not be immediately apparent how there could be an edge $i \to j$ in the heap graph in Figure 72 without a crossing of edges.

However, there is always a deformation of a heap graph which uncrosses all edges while leaving the graph in standard configuration.

Proposition 244. *For each heap ϕ_n there is a progressive image $\iota(\Phi)$ of a heap graph of ϕ_n in the plane so that ι is an embedding and the image is in standard configuration.*

Proof. We prove by induction that there is a construction of a heap graph

$$\Phi_n = (G^\phi, \{g_1, \ldots, g_n\})$$

for ϕ_n in the plane with no crossings.

Without loss of generality we will assume that we place the nodes g_i of our heap graphs at integer coordinates $(0, -i) \in \mathbb{R}^2$.

Suppose there is an embedding of a graph $\Phi \restriction_{k-1}$ of the restricted heap $\phi \restriction_{k-1}$ in the plane in standard configuration. Then a heap graph $\Phi \restriction_k$ of the restricted heap $\phi \restriction_k$

may be formed from $\Phi\restriction_{k-1}$ by adding an edge $e_k : g_k \to g_{\phi(k)}$ (assuming $\phi(k)\downarrow$). It is sufficient to show that this edge made be added in such a way that it does not cross any edge of $\Phi\restriction_{k-1}$.

Consider a horizontal line H through the point $g_{\phi(k)}$ in $\Phi\restriction_{k-1}$. H will intersect some edges of $\Phi\restriction_{k-1}$. We label these edges with L if their intersection with H is left of $g_{\phi(k)}$ and R if their intersection is right. Furthermore, any paths in $\Phi\restriction_{k-1}$ below H which contain a labelled edge have all their other edges labelled similarly. Any edges below H which are not labelled in this way are given the label R.

Every edge below H has exactly one label since no edge can intersect H more than once as $\Phi\restriction_{k-1}$ is progressive in the plane and if any two paths containing a single edge have a common extension as $\Phi\restriction_{k-1}$ is a forest.

Since $\Phi\restriction_{k-1}$ is the continuous image of a compact space, we may draw an edge $e_k : g_k \to g_{\phi(k)}$ so that every R-labelled edge is to the right of e_k and every L-labelled edge is to the left of e_k. Therefore, e_k crosses no other edge of $\Phi\restriction_k$. □

Example 245. Following the induction step of the proof of Proposition 244, consider the example graph shown in Figure 73. $\Phi\restriction_{k-1}$ is shown in Figure 73(a) along with the point g_k. The line H and the labellings of the edges of $\Phi\restriction_{k-1}$ below H are shown in Figure 73(b). The edge e_k and hence the graph $\Phi\restriction_k$ is shown in Figure 73(c).

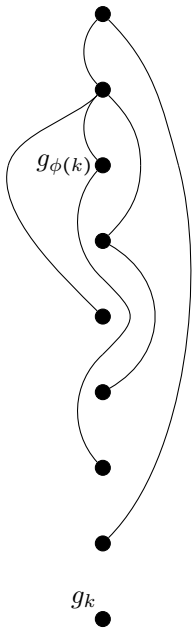
Example 246. The proof of Proposition 244 also gives us a procedure for transforming a heap graph Φ with crossing edges into a heap graph $\tilde{\Phi}$ with no crossings. The graph Φ gives us the heap ϕ for which it is a graph, and $\tilde{Phi}$ is then built "top-down" using the inductive method of the proof. For example, to add an edge $i \to j$ in Figure 72, we first naïvely add the edge, allowing it to cross, and then rebuild the graph from the induced heap structure on the nodes. This can be seen in Figure 74.

Though we have proved that, even in standard configuration, all heap graphs may be embedded in the plane, we will frequently consider examples where we allow lines to cross, and so must in general consider heap graphs as images of progressive maps. For example, when we affix heap graphs to other diagrams in the plane in Section 4.1.4 it is extremely convenient for us to consider cases where our diagrams have no progressive planar embedding. Consider, for example, Figure 75. To keep the images of both heaps in standard configuration and to keep the $\multimap$-schedule progressive, there can be no arrangement of edges which doesn't have a crossing.

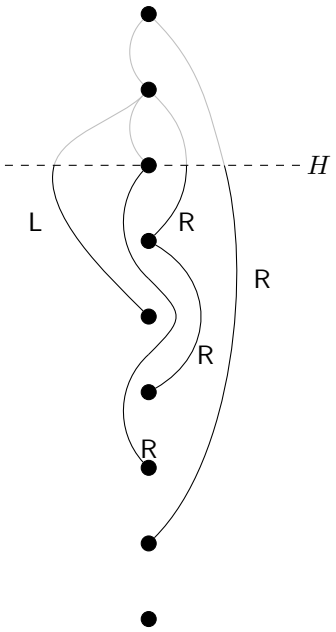
A.4.2 The strategy $!\neg : !\mathbb{B} \multimap !\mathbb{B}$

Recall from Example 76 the strategy for $\neg : \mathbb{B} \multimap \mathbb{B}$. Let us consider the strategy $!\neg : !\mathbb{B} \multimap !\mathbb{B}$. $!\neg$ will consist of labelled $\multimap$-schedules with attached O-heap graphs. We cannot explicitly write down every such graph as there are an infinite number of them, but we present one example in Figure 76.

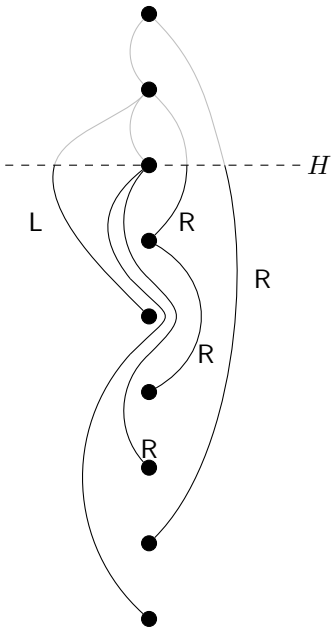
Notice that not every $\multimap$-schedule is part of a play of $!\neg$. For example, the graph in Figure 77 shows a triple $(S, (\phi, \overrightarrow{a}), (\psi, \overrightarrow{b}))$ which is a play of $!\mathbb{B} \multimap !\mathbb{B}$ but is not a play of $!\neg$ as it does not satisfy $\psi = S^*\phi$.



(a) The graph $\Phi\restriction_{k-1}$.

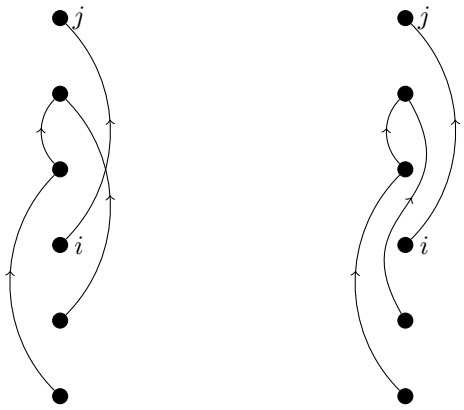


(b) Labelling of the edges of $\Phi\restriction_{k-1}$ below H.



(c) The graph $\Phi\restriction_k$.

Figure 73: Adding an edge without crossing.



(a) Add edge $i \to j$ to the graph from Figure 72, with crossing.

(b) Rebuild graph top-down without crossing in the manner of the proof of Proposition 244.

Figure 74: Adding an edge $i \to j$ without crossings.

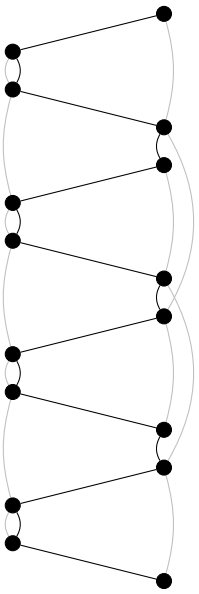
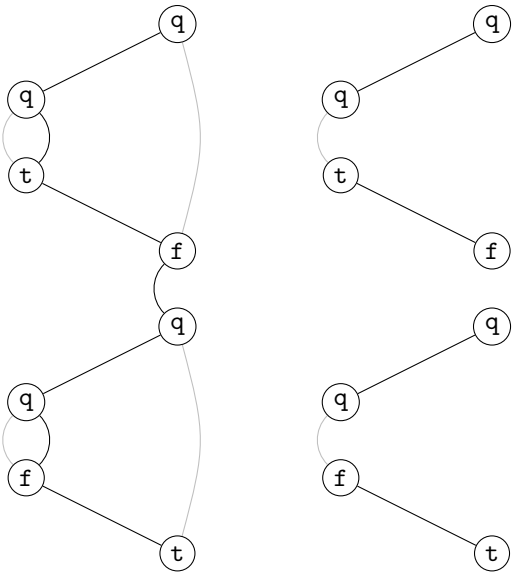


Figure 75: A $-\circ$-schedule S and two heaps Π and Φ, as in the construction of $[\Pi, S, \Phi]$.



(a) A play of the strategy $!\neg$. Notice that $\Phi = S^*\Psi$.

(b) The $[\Pi, S \| T, \Psi]$-threads of this play. Notice they are all plays of $\neg : \mathbb{B} \multimap \mathbb{B}$.

Figure 76: One play of the strategy $!\neg : !\mathbb{B} \multimap !\mathbb{B}$.

A.4.3 Example of a game $!!G$

Consider the game G given by the diagram in Figure 78. The first few set of $!G$ are then as in Figure 79, with $\pi_{!G}$ given by truncation of labelled heap graphs.

Now consider the position of $(!!G)(7)$ given in Figure 80(a), with the "outer" heap Ψ_7 and inner heaps $\Phi^{(i)}$. Since Ψ acts as truncation on the "inner" O-heaps, this uniquely determines the following double O-heap structure on the sequence of final labels of nodes of labels of Φ as show in Figure 80(b), where Ψ is as before, $\Psi \succcurlyeq \Phi$ and Φ sends g_i to the node whose label corresponds to $\Phi^{(i)}(g_i)$.

Conversely, consider the double structure shown in Figure 81(a), which satisfies $\Psi \succcurlyeq \Phi$ and Φ-threads are plays of G. This uniquely determines the following position in $(!!G)(7)$ shown in Figure 81(b), where $\overrightarrow{g}^{(i)}$ are given by Ψ-threads and $\Phi^{(i)}$ sends $g_j^{(i)}$ to the node whose label is $\Phi(g_j)$ when truncated by Ψ.

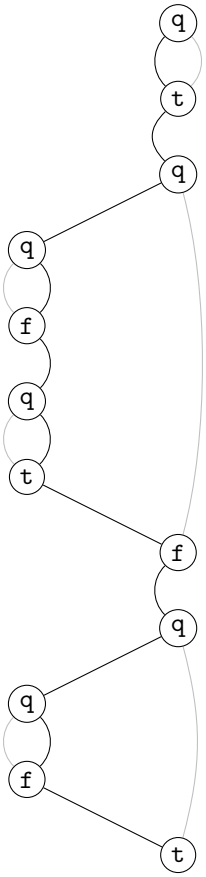


Figure 77: Not a play of !¬.

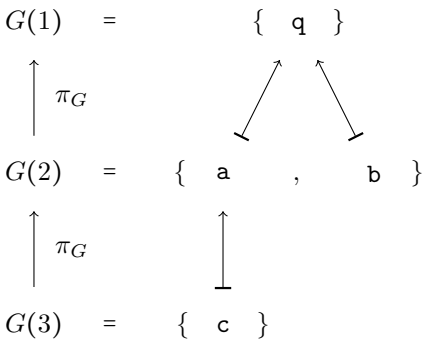


Figure 78: An example game G.

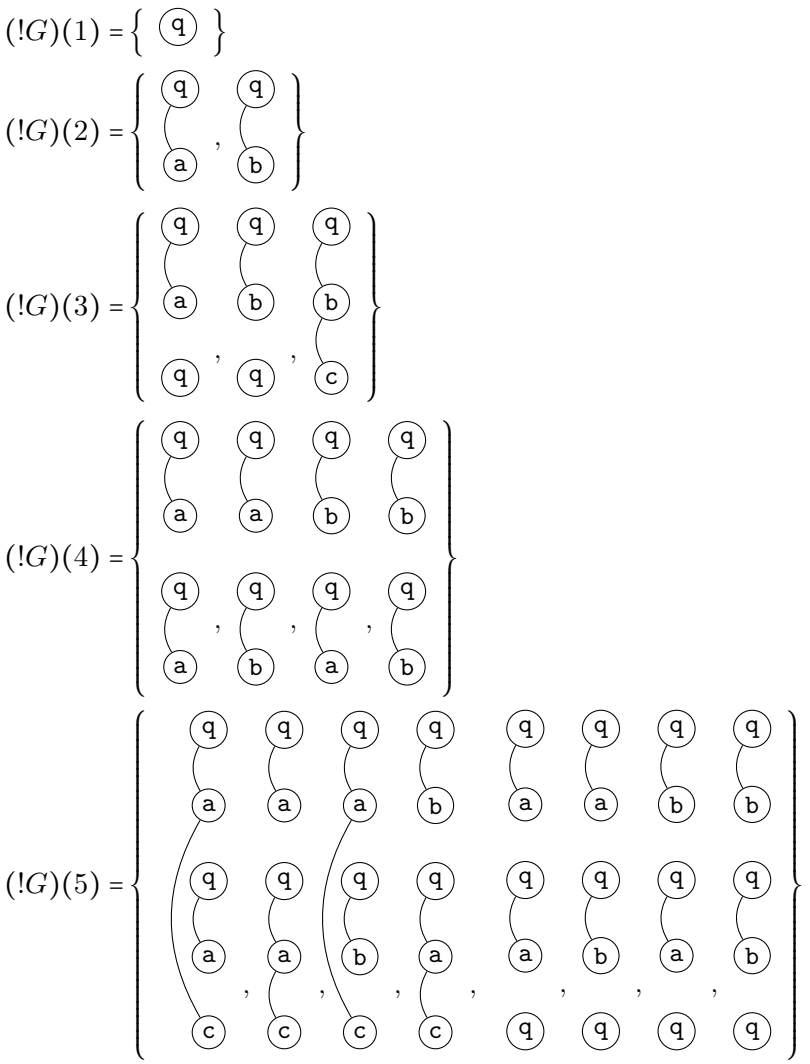
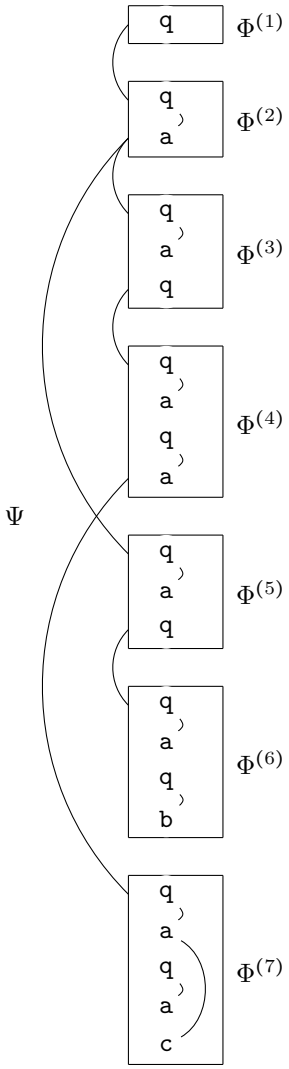
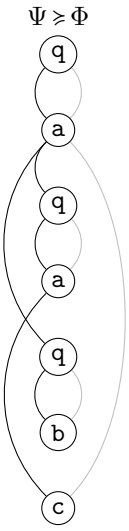


Figure 79: Some positions of !G, where G is given in Figure 78.

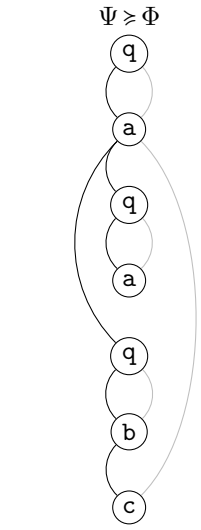


(a) A position of the game !!G, where G is as in Figure 78.

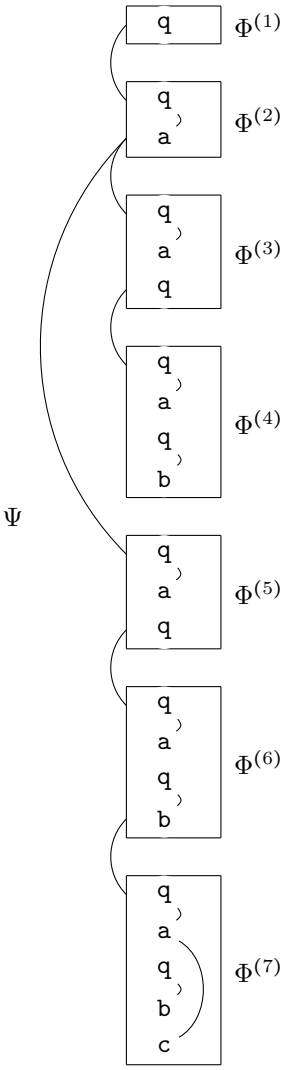


(b) The same structure, indicated with two heap graphs sharing the same nodes.

Figure 80: Different representations of nested heaps.



(a) An example of a pair of heap graphs sharing the same set of nodes.



(b) The uniquely determined position of !!G.

Figure 81: Reconstructing nested heaps.